Gordon Day

University of Strathclyde

MSc Energy Systems and the Environment

Thesis

Development of Linear Analysis Tools to Aid Design of Large Scale Wind Turbines

Contents

Acknowledgements

None of this work would have been possible without the assistance of Prof W. Leithead and Sergio Dominguez of Strathclyde University.

Abstract

The aim of this thesis is to evaluate methods of predicting fatigue damage to wind turbines caused by random stress loading over long time periods. Traditionally, such damage is calculated in the time domain using stress cycle counting methods such as the Rainflow counting algorithm which is used here as a control method. Since the stress loadings are random, extrapolation of a time series evaluation from data lasting a few minutes to give an estimate for a period of 20 years gives rise to a likelihood of an inaccurate prediction. Using much longer time series data would undoubtedly produce more accurate results but processing time taken to perform the calculations would also increase, ultimately to the point where it becomes unfeasible. It would therefore be more beneficial if a reasonably accurate fatigue prediction could be calculated from data which is presented in a form that permits much quicker calculations. Data that has been transferred from the time domain to the frequency domain and is presented in the form of a power spectral density (PSD) should, in theory, present the solution to this problem. The Dirlik method of stress cycle analysis in the frequency domain is the method that is tested here against the Rainflow method. This report will show that the Dirlik method does indeed give very good fatigue estimation results when compared with the Rainflow counting method and, subsequently could be recommended for use when calculating long term fatigue damage caused by random stress cycle loading.

Introduction

The very nature of a wind turbine and its function of deriving energy from the wind means that it has to operate in environmentally hostile conditions. Land-based wind turbines are exposed to a great deal of stresses and strains caused by the forces of fluctuating wind and off-shore turbines also have to deal with additional forces produced by sea currents and waves. These stresses and strains are varied in magnitude and direction and over time they will eventually cause fatigue which weakens the structure beyond serviceable limits. One of the most important challenges facing modern turbine designers is to find ways of increasing the lifespan of wind turbines by minimising structural failure due to fatigue[4][5]. In order to do this, fatigue needs to be reasonably accurately predicted for the projected lifecycle of the turbine. Current wind turbines have an approximate life expectancy of around 20 - 25 years. Accurate computer modelling of structural fatigue enables designers to alter physical parameters of the turbine and examine any effect these changes have on the structure. If a change has a positive effect on the modelled structure then, theoretically, the same change in the physical design of the turbine should prolong its life-cycle.

There are some difficulties associated with this type of prediction though. Firstly, it can be difficult to determine the accuracy of a newly created computer simulation/prediction model. It is not feasible to compare the results of a simulation with real physical results since the time-span involved is so great so other methods must be used to validate results. One possible method could be to use two or more different prediction models and compare the results. A particular model which has been previously proven to be reasonably reliable can be used as a control model in the development of new models.

Another difficulty involves the nature of the damaging forces. Wind is essentially random and intermittent. Every wind farm site will have been subjected to a thorough feasibility study to ensure there is a sufficient wind resource available but it is ultimately impossible to accurately predict wind patterns over a 20 year period. The subsequent problem for the designer is how to model the expected damaging effects of wind over the life of the turbine. This problem may be tackled by examining the loads on the turbine in a different manner. Traditional methods of fatigue prediction using data presented in the time domain involve calculations that can become very time intensive. If the design process involves creating a model then analysing changes in fatigue caused by subtle changes to component parameters, then the issue of calculation time becomes very important. Instead of analysing the loads in the time domain, which plots signal amplitude against time, the loads can be analysed in the frequency domain, which plots signal amplitude against frequency. The signal can be transformed from time domain to frequency domain using a Fourier Transform. In the frequency domain, calculations can be performed much more quickly at the acceptable cost of a slight reduction in accuracy. Another significant advantage of this is that the fatigue prediction process can be directly incorporated into control design. This approach is often the best where random signals span lengthy time series.

Although wind patterns cannot be accurately predicted in the long term, this does not mean that they are a completely unknown quantity. Clever use of probability density functions and other statistical methods can build a picture of wind patterns which is accurate enough for the purposes of the prediction model. A probability density function is usually displayed on a graph as a bell shaped curve. The x-axis contains a numerical variation of possibilities and the y-axis indicates the probability of each possibility. It is known that the probability of wind speed occurrences follows a particular probability density function called a Weibull Distribution. The exact shape of the Weibull Distribution for a particular site depends on the average wind speed measured during the feasibility study on that site.

The aim of this project is to compare two different methods of fatigue prediction. These are the Rainflow method and the Dirlik method. The Rainflow method has been widely used in fatigue analysis since the 1970's and is regarded as the control method here. It is used to analyse data in the time domain and is a simple but effective way of identifying stress cycles. The Rainflow method is renowned for its accuracy and the ease with which it can be programmed. The Dirlik method has been around since the mid 1980's and is the method that is to be tested against the Rainflow method. The Dirlik method analyses data which has been transferred into the frequency domain and is a series of calculations based on four moments of area of a Power Spectral Density function. Since the Dirlik method is used in the frequency domain then theoretically it should produce reasonably accurate fatigue prediction results more quickly than the Rainflow counting algorithm for random loading data over a lengthy time series.

Background

The need to derive energy from non-finite sources is on the increase. Advances in marine technology[7] indicate that wave and tidal power may soon become viable options. Solar energy is a technology that has been commercially recognised for more than 3 decades and is now in use globally although the vast majority of its applications are small scale[8].

For national renewable energy production, wind and hydro power are the only viable energy sources that current technology allows to be exploited on a large enough scale and since the UK has the largest share of wind resource in Europe[6], it makes sense to exploit this resource to its maximum potential. Maximising this exploitation ultimately means building bigger wind turbines and consequently technology needs to be pushed further to achieve this. Onshore wind turbines are ultimately limited in size due to their visual impact on the environment. The largest wind turbines currently being installed onshore in the UK have a rated power of approximately 2MW. Offshore wind turbines have no such size restriction as they are considered to have little or no detrimental visual impact on the environment. Offshore wind turbines are, therefore, restricted in size only by the available technology. The largest offshore wind turbines currently being installed globally have the capacity to produce 4.5MW of power and 5MW turbines are currently being tested[10].

As turbines increase in size, the stresses they have to endure also increase in size since they are being exposed to more of the damaging effects of the wind. Without advances in fatigue prediction techniques, there would ultimately be a reduction in lifespan of wind turbines as their size increases. This would make them commercially unattractive as the cost of installing large offshore wind turbines is considerably greater than that of installing onshore wind turbines to create the same amount of power. In fact, figures supplied by the Danish Wind Industry Association[11] suggest that offshore wind turbines cost approximately 70% more per MW installed than onshore turbines. It is therefore very important that wind farm operators are able to purchase turbines that will last as long as possible in order to maximise their potential.

The consequence of all this is that the ability to predict fatigue with a reasonable amount of accuracy is a fundamentally important part of the turbine design process. If this prediction process can also be incorporated into the control design stage then the effects that small alterations to the control parameters have on the fatigue levels of the turbine can instantly be seen. Traditional methods of fatigue estimation analyse data which is presented in the time domain and while these methods are sufficiently accurate, they are computationally time intensive and are not particularly suited to incorporation into control analysis. This thesis will investigate the viability of a fatigue prediction technique which analyses data presented in the frequency domain where computations are much less time intensive and the ability to incorporate the fatigue analysis into control analysis should be a natural progression.

Fatigue

The vast majority of structural failures are due to some sort of fatigue in the component material caused by stress and/or strain loading experienced over a period of time. Very few modern structures experience purely static loading with some form of cyclic or repeated loading being the most commonly observed. These cyclic or repeated loads often lead eventually to structural failure caused by fatigue. This is particularly true of wind turbines which are deliberately placed in positions where they are exposed to somewhat hostile elements.

Fatigue is defined as a failure which is caused by a varying or repeated load. This load is never at a sufficiently high level to cause structural failure in a single application. There are two fundamental forms of cyclic stress called low-cycle fatigue and high-cycle fatigue and their characteristics are very different. Subsequently the physical conditions caused by these cycles which lead to structural failure also differ.

Low-cycle fatigue normally leads to structural failure in less than 100,000 cycles. The cycles are usually large and result in a relatively short life. Low-cycle fatigue is typically associated with significant amounts of plastic deformation.

High-cycle fatigue normally leads to structural failure in greater than 100,000 cycles. The cycles are low load and high-cycle fatigue is normally associated with long component life. The applied stress resulting from high-cycle fatigue is normally confined to the elastic region.

In addition to this, the stress cycles can either be periodic, continuous or random. Periodic stress cycles need not be continuous but they can more or less be treated as such since the same set of stress cycles is repeated so even if there is a rest gap between the sets of cycles it makes very little difference to the analysis process. Continuous stress cycles are the easiest to analyse with a degree of accuracy since the same stress cycles are repeated without a break. Random loading stress cycles are by their nature very difficult to analyse for the purpose of fatigue prediction. The stress cycles experienced by wind turbines are random and vary greatly in magnitude making it very difficult to predict turbine fatigue over an expected lifespan of around 20 years.

There are many different methods that can be employed to predict fatigue and subsequently estimate the expected lifespan of a component or structure. The method employed will be largely dependent on the type of stress experienced during the lifespan. The aim of any fatigue prediction tool is essentially to estimate how many stress cycles will lead to failure of the component or structure in question. To this end, the most basic form of fatigue prediction uses the Palmgren-Miner rule. The Palmgren-Miner rule is commonly used and asserts that terminal fatigue is caused by a combination of stress cycles of differing magnitude (s_1, s_2, s_3, s_4,……,s_x, for x levels of stress). The number of cycles to failure for each stress level is called N and can be obtained from a suitable S-N Wohler curve for the material in question. The number of stress cycles actually experienced by the component or structure is called n and is obtained from a method of cycle counting. The fatigue level is deemed to be critical when $\frac{n_1}{N_1} + \frac{n_2}{N_2} + \frac{n_3}{N_3} + \ldots + \frac{n_x}{N_x} > 1$.

The Palmgren-Miner rule is a very basic concept and it has been found to have limitations in certain types of application. Nevertheless, this rule and many variations of it are extensively used in the process of fatigue prediction.

Fourier Transform

There are two different ways to view a signal and these are in the time domain and the frequency domain. The most common method of viewing a signal is in the time domain which plots the signal amplitude as the dependent variable against time as the independent variable. An example of a signal plotted in the time domain can be seen in figure 1.

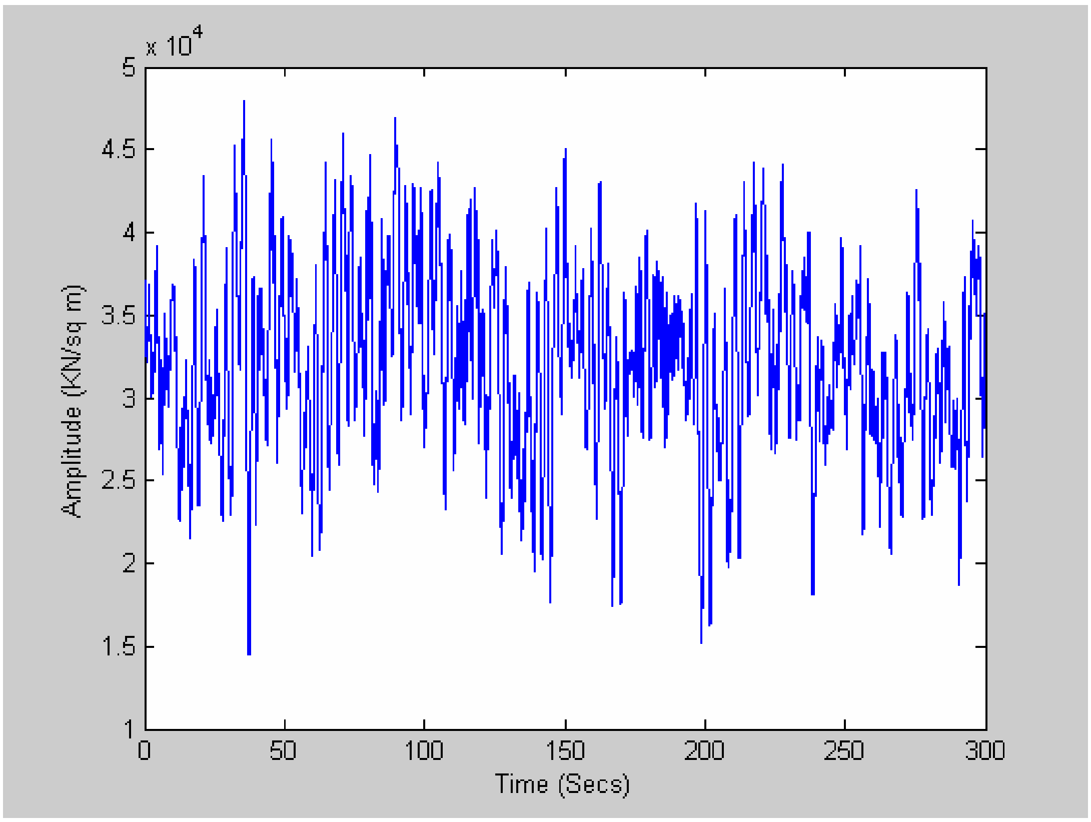


Figure 1. Time Series Signal of Stress Loading on a Wind Turbine

For most signal analysis this simple method is sufficient and it is also relatively easy to understand. There are some signal types that make this form of signal viewing somewhat impractical however, such as signals that are discontinuous or random over a large time period. These signals are best viewed in the frequency domain where the signal amplitude as the dependent variable is plotted against signal frequency as the independent variable. An example of this can be seen in figure 2 where the signal from figure 1 is represented in the frequency domain and is shown as a log-log plot.

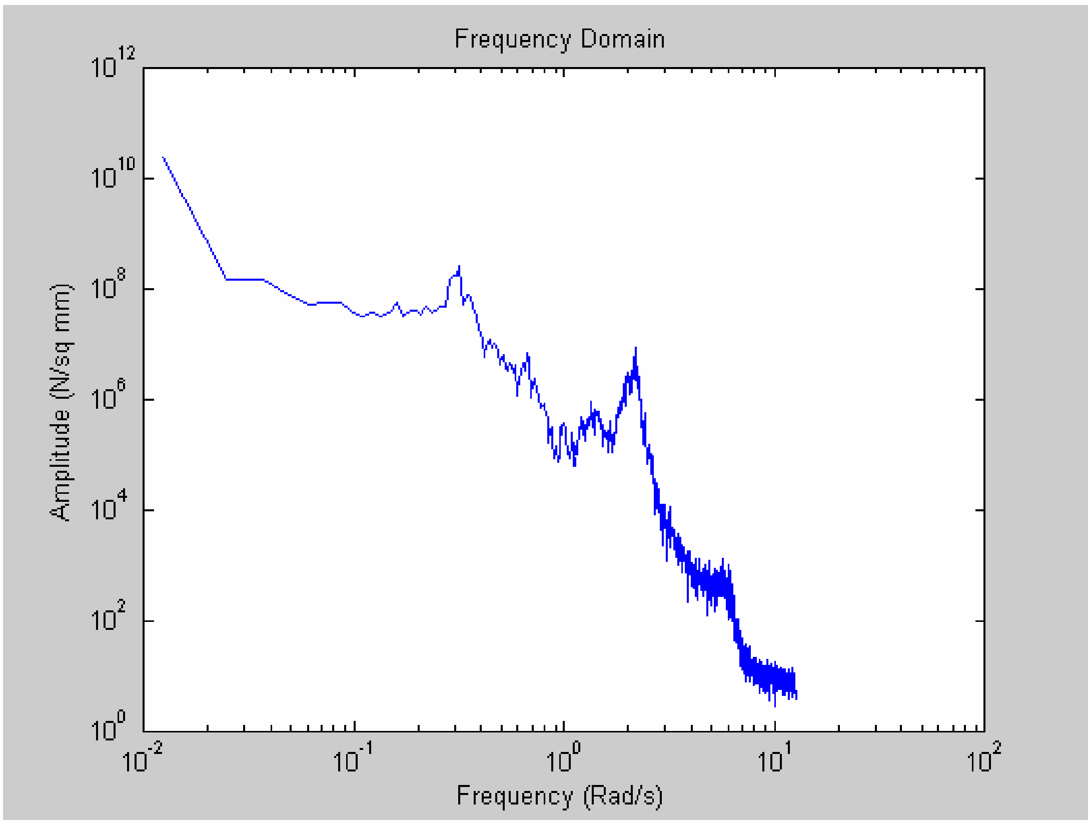


Figure 2 Example of Time Series Signal Converted to Frequency Domain

Although the concept of the frequency domain may be slightly harder to understand than the time domain, the mathematics involved the analysis are often easier.

Any signal can be represented in either time or frequency domain and a Fourier Transform is used to convert the signal from the time domain to the frequency domain. An inverse Fourier Transform is used to convert the signal from the frequency domain back to the time domain. Although the basic mathematical principal is the same, there are many different versions of the Fourier transform utilising variations on the mathematical formulae. The Fast Fourier Transform is most commonly used for these applications and is a version of the Discrete Fourier Transform which reduces the number of computations for a series with N points by a factor of $\log_2(N)$.

The Discrete Fourier Transform is briefly described as follows.

If the series

$$X_0, X_1, X_2, X_3 \ldots X_k \ldots X_{N-1}$$

is a complex series with N elements and that the series is out with this range N periodic so that $X_k = X_{k+N}$, then the Discrete Fourier Transform of this series is

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)e^{-jk2pn/N} \qquad \text{for } n = 0 \ldots N-1$$

The inverse function is as follows

$$X(n) = \sum_{k=0}^{N-1} X(k)e^{jk2pn/N} \qquad \text{for } n = 0 \ldots N-1$$

A non-complex series can be similarly represented by setting the complex component of each element to 0.

Weibull Distribution

Modern wind turbines are designed to operate throughout a range of wind speeds. The significant points within this range being cut-in speed, rated speed and cut-out speed. The cut-in speed is the minimum wind speed that the turbine requires to generate electricity. The rated speed is the wind speed at which the turbine is producing maximum power and the cut-out speed is the point at which the wind is has become too strong for safe operation of the turbine. Precise figures vary depending on the turbine used and the wind resource at the site in question. Wind turbines generally operate at their rated speed for around 25% of their lifetime and will produce electricity for 75 – 80% of the time[6]. Most wind turbines will produce 30-40% of their rated power over the course of their lifetime and this is mainly due to the fact that the winds that drive them tend to be fresh to moderate for most of the time. Turbine designers need to know the type of environment that their turbines will be subjected to and so use statistical tools to try and predict wind distribution speeds. The probability of wind speeds at a prospective wind farm site can be modelled using a probability density function (PDF). Wind speeds are known to follow a particular PDF called a Weibull distribution and an example of this can be seen in figure 3.

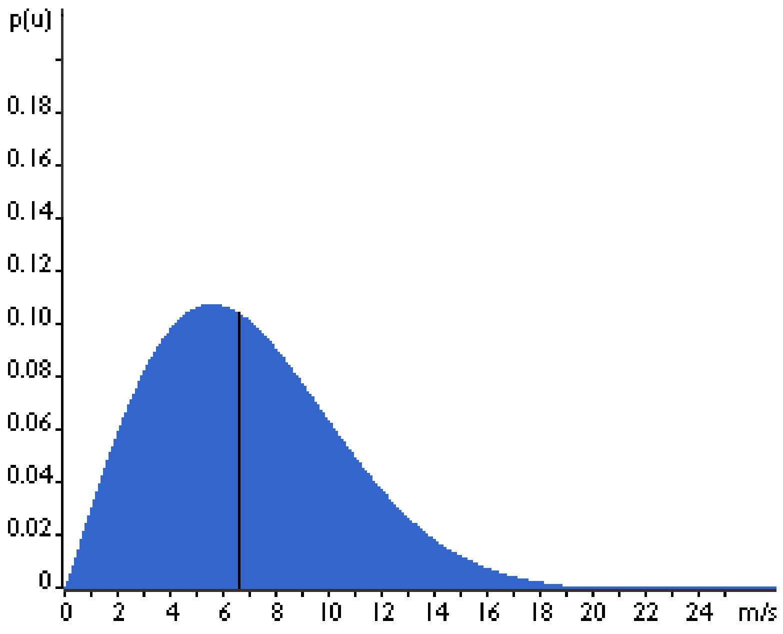


Figure 3 An Example of the Weibull Distribution

The Weibull distribution is a representation of wind patterns constructed by averaging wind speed over 5 or 10 minute intervals and is therefore suitable for long term wind estimation but would not take into account very short term effects like localised turbulence.

Power Spectral Density

Stress loading on a structure for the purposes of fatigue damage analysis is normally displayed in the time domain. Under most circumstances, this method produces satisfactory results however, if the loading is random in nature then extremely large time records are required to determine an accurate description of the loading. An alternative method is to represent the loading in the frequency domain. This is done by using a fast Fourier transform and the result is a 'Power Spectral Density (PSD)' plot. A Power Spectral Density plot is a normalised density plot. It displays the mean square amplitude of each sinusoidal wave with respect to its frequency. There are many variations in the calculation of a Power Spectral Density and in this case Welch's averaged, modified periodogram method was used, this method is conveniently pre-programmed as a function within Matlab. Welch's method divides a discrete-time signal vector into eight sections using a 50% overlap. A Hamming window is used on each section and eight modified periodograms are then computed and averaged. This method vastly reduces an effect called 'leakage' and leads to a more accurate result. If the signal is complex then Welch's method will produce a two-sided PSD, otherwise a one-sided PSD will be produced.

A typical PSD plot provided using Welch's method is shown in figure 4. For convenience, the PSD is shown as a log-log plot.

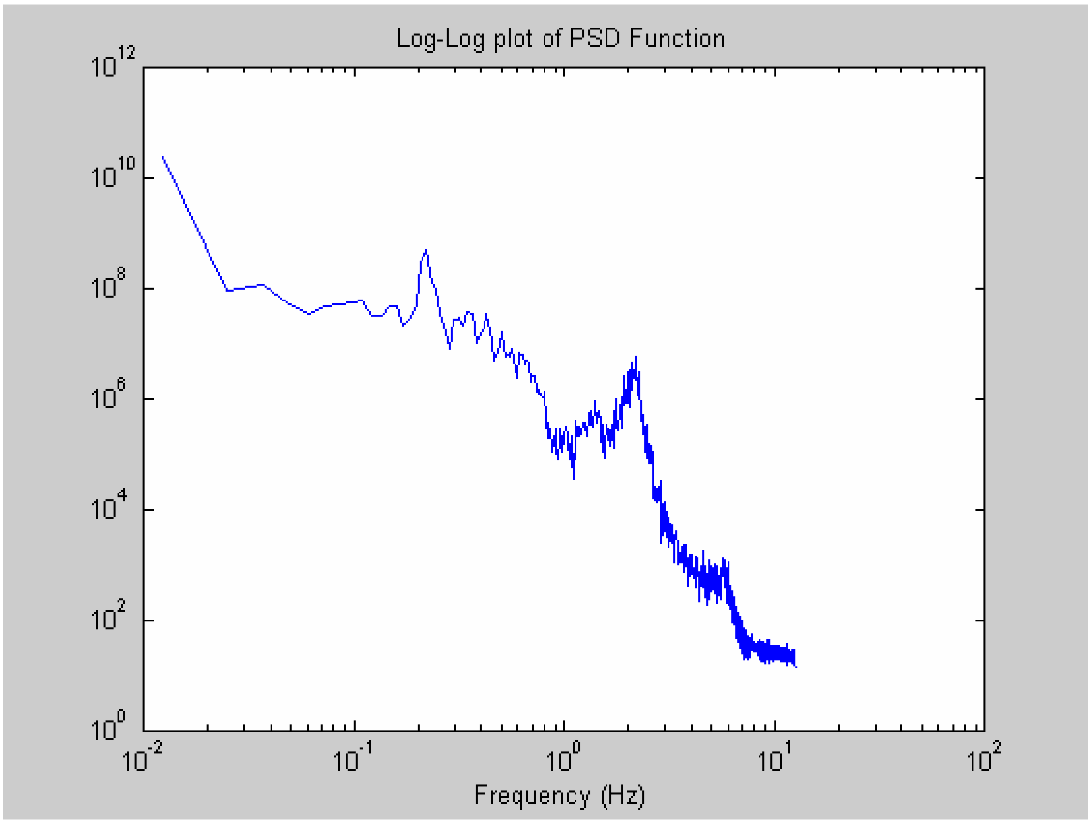


Figure 4. Log-log plot of typical PSD obtained using Welch's method

Rainflow Counting

Description

Rainflow analysis is a method of approximating stress cycles in a structure from time series data of loads on the structure. It was developed in 1968 by Tatsuo Endo and M. Matsuiski[9] and has been widely used since the late 1970's when advances in computer technology meant that this fairly simple algorithm could be easily programmed.

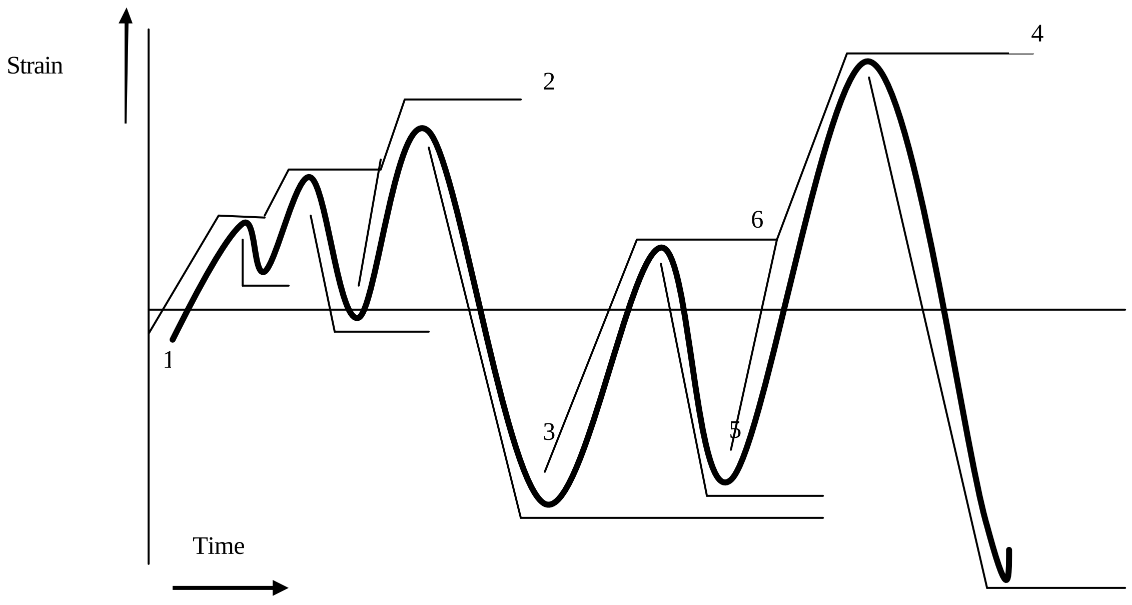


Figure 5 Rainflow method

Figure 5 shows a time series of random data. The additional lines indicate how water (or rain) would flow if the diagram was rotated clockwise through 90^0. The rainflow begins at each peak and ends when one of the following conditions is satisfied.

1. The end of the time series is reached.
2. It flows opposite a peak of greater magnitude to the one from which it started.
3. It is interrupted by a flow which started at an earlier peak.

In figure 5, the flow that begins at position 1 ends at position 2 because it satisfies the second condition. The flow that starts at position 3 ends at position 4 because it satisfies the first condition and the flow that starts at position 5 ends at position 6 because it satisfies the third condition.

Each flow is classed as a half cycle. If a half cycle can be matched to another half cycle of similar magnitude but opposite direction of flow then they are paired to make a full cycle. The final cycle count will be a combination of full and half cycles and these will be classed into bins of magnitude ranges.

Since a new cycle or half-cycle is started at every peak, the entire data range is counted.

Despite its apparent simplicity, the Rainflow method is widely regarded as being the most accurate method of stress cycle counting available. The reason it tends to give better results than other methods is that it successfully counts very small cycles which are often missed by other counting methods. A typical rainflow counting algorithm will also contain an adjustable filter which can eliminate cycles that are insignificantly small by setting a minimum significant height. It also accounts for the mean stress of each cycle which is the average of the positive and negative peaks of each cycle.

There are variations of the algorithm that forms the basis of the rainflow analysis but each variation ultimately performs the same task. A commonly used and relatively simple algorithm can be obtained from the American Society of Mechanical Engineers[2]. Rainflow counting consistently outperforms other methods of stress cycle analysis within the time series domain. Since its use is limited to time series data, rainflow counting is best applied to periodic or continuous signals. These limitations become more significant when dealing with random loading over a very long period of time. Wind turbines have an approximate life-span of 20 years and the damage due to fatigue in that period will be largely due to the wind, which is random and intermittent. It would be unfeasible to analyse a time series of data whose magnitude is counted in years and it is entirely conceivable that rainflow analysis of a data set lasting a few minutes would not give an accurate estimation when extrapolated to represent such a lengthy period of time. Random loading of lengthy time periods may therefore be best estimated by converting the data from the time series domain to the frequency domain for analysis. This can be done using a variation of the Fourier Transform.

The process of programming the Rainflow counting method was relatively straightforward. Figure 6 shows a typical data series of the type to be analysed. The data shows simulated stresses on a wind turbine tower over a 5 minute period. These stresses require to be analysed as described above.

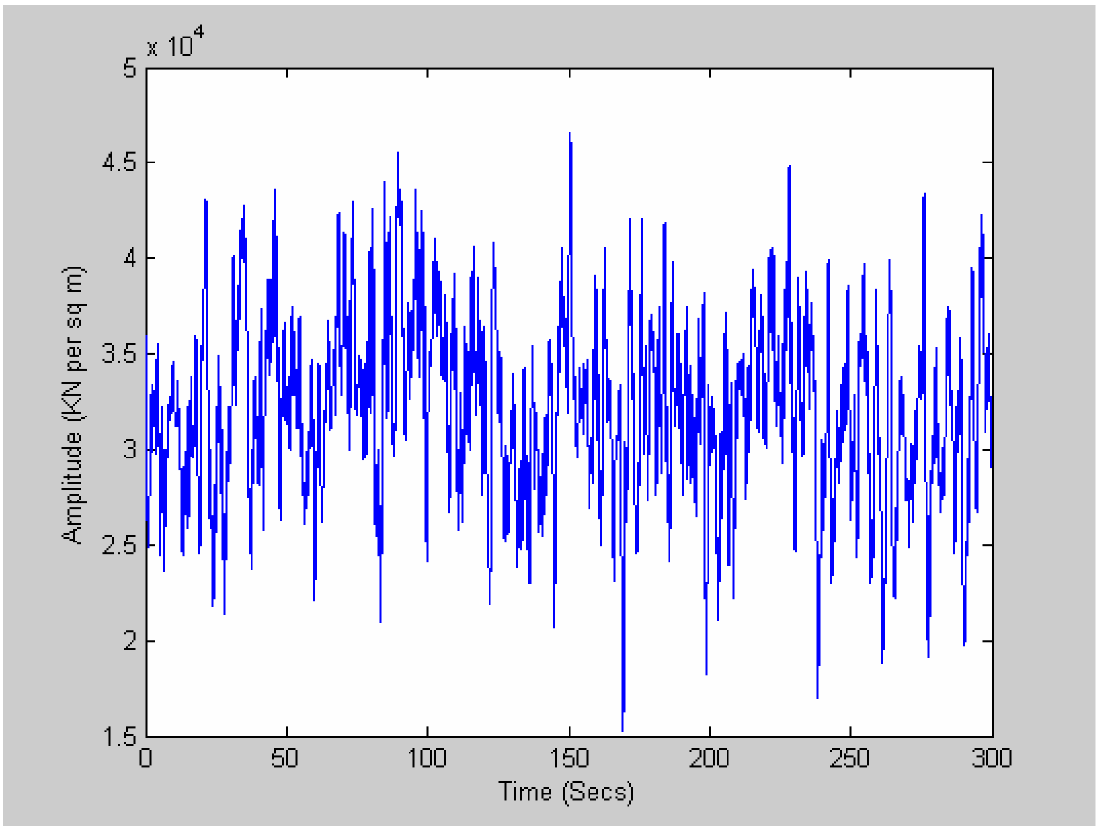


Figure 6 Example of Time Domain Data Series Showing 5 min of Stress Loading on a Wind Turbine

Rainflow Coding

There are two basic programs that were provided by Mathworks[4] and these were used as the basis of the Rainflow analysis. The first of these functions identifies the peaks in the data and the second performs the actual rainflow calculation. A series of Matlab scripts were then written which used these functions to analyse the available data. Many variations were considered and ultimately a very simple version was used. Since the Rainflow counting algorithm was not the process which was on test here, it was deemed unnecessary to perform it on the full range of data samples used for the Dirlik procedure. Therefore, the Rainflow counting program was simply tested on a few randomly selected data sets and the results compared with those of FLEX. Since the results were identical to those of FLEX it could be safely concluded that a fully operational Rainflow counting program had been created within Matlab, albeit with the assistance of Mathworks[4]. A simple version of the rainflow script can be found in appendices.

The data to be analysed contains information for wind speeds spanning the full operating range of the wind turbine (from 4 – 24m/s) in 2m/s steps. There are 2 sets of data for each wind speed and these are taken at a +/- 10^o offset from the direction of the wind. This can be seen in figure 7.

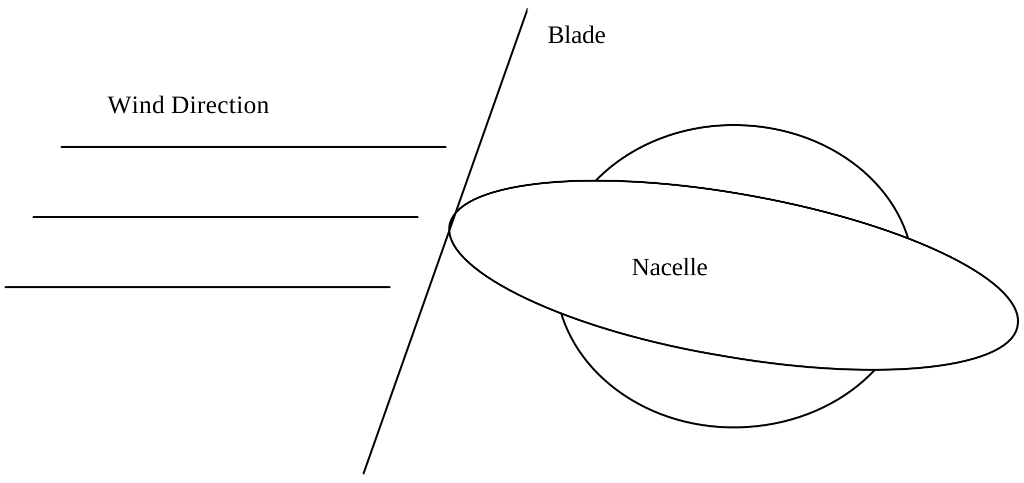


Figure 7. Plan View of Offset of Wind Turbine to Wind Direction

In addition to this, there are 2 versions of many of the data sets. The first version features undamped data and the second version features data with the natural vibration of the tower damped out. The difference in fatigue damage between these sets of data is given and it was originally intended to try and replicate these differences with the programs written here. It was later considered that while this would have been useful, it would also have been a somewhat restrictive practice. Since the calculated damage was provided with all the data sets, it was decided to run comparisons with as many different sets of data as possible and this would provide a much more robust test of both the Rainflow and Dirlik algorithms.

When the Rainflow cycles have been counted, there are further calculations to be performed in order to calculate the damage which is expected over the 20 year life span of the wind turbine. The damage calculation for a single wind speed is described in the following equation which was obtained from Thomsen[3].

$$\mathrm{R_{eq}} = \left(\frac{\sum n_i R_i^m}{n_{eq}} \right)^{1/m}$$

Where $\mathrm{R_{eq}}$ represents the damage equivalent, n_i is the number of cycles of magnitude R_i over i load range levels (usually 30–50), n_{eq} is a constant representing the corresponding number of load ranges and m is the Wöhler curve exponent. In the simple algorithm that was finally chosen, there was, in fact, no restriction on the number of ranges as each cycle was treated individually for increased accuracy. Therefore the only way that there would be more than one occurrence of a cycle is if there was more than one cycle of exactly the same size. The damage level for each wind speed also needs to be weighted according to the Weibull distribution to represent the statistical likelihood of that wind speed occurring. The data is then adjusted to correspond to a 20 year period and the damage for each wind speed is finally summated. The damage figures given by the final Rainflow program were identical to the figures supplied by FLEX. This was to be expected since FLEX also uses a Rainflow counting method to calculate damage and was simply a confirmation that a successful Rainflow counting program had been created specifically to determine the expected damage caused by random stress cycles to a wind turbine over a 20 year life span.

While graphical output of the Rainflow counting algorithm was neither required nor feasible with so many data sets, figures 8 to 10 show typical graphical output that is available when performing a Rainflow counting procedure using Matlab.

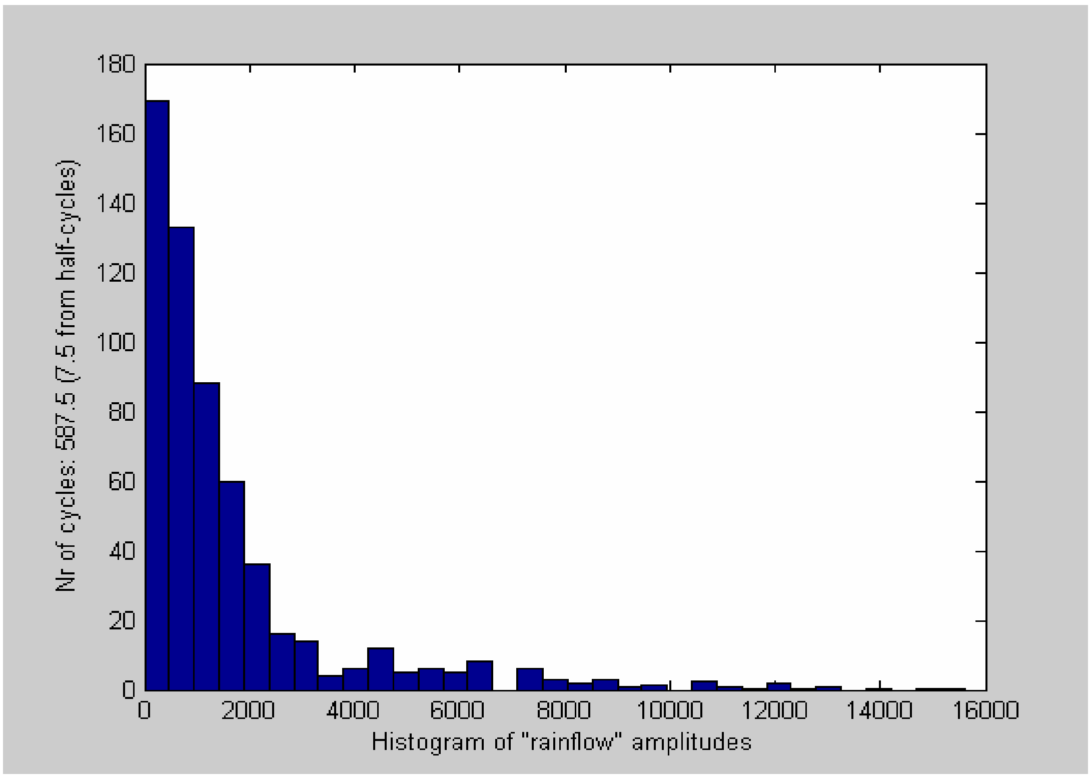


Figure 8 Histogram of Amplitudes

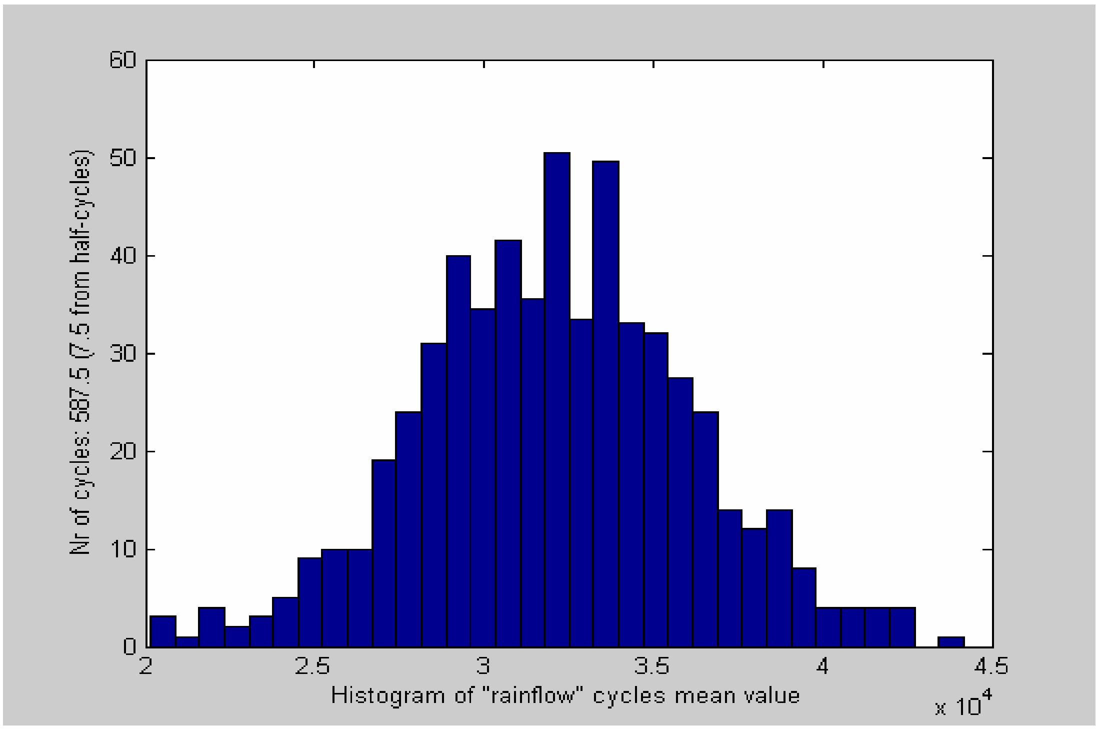


Figure 9. Histogram of Mean Values

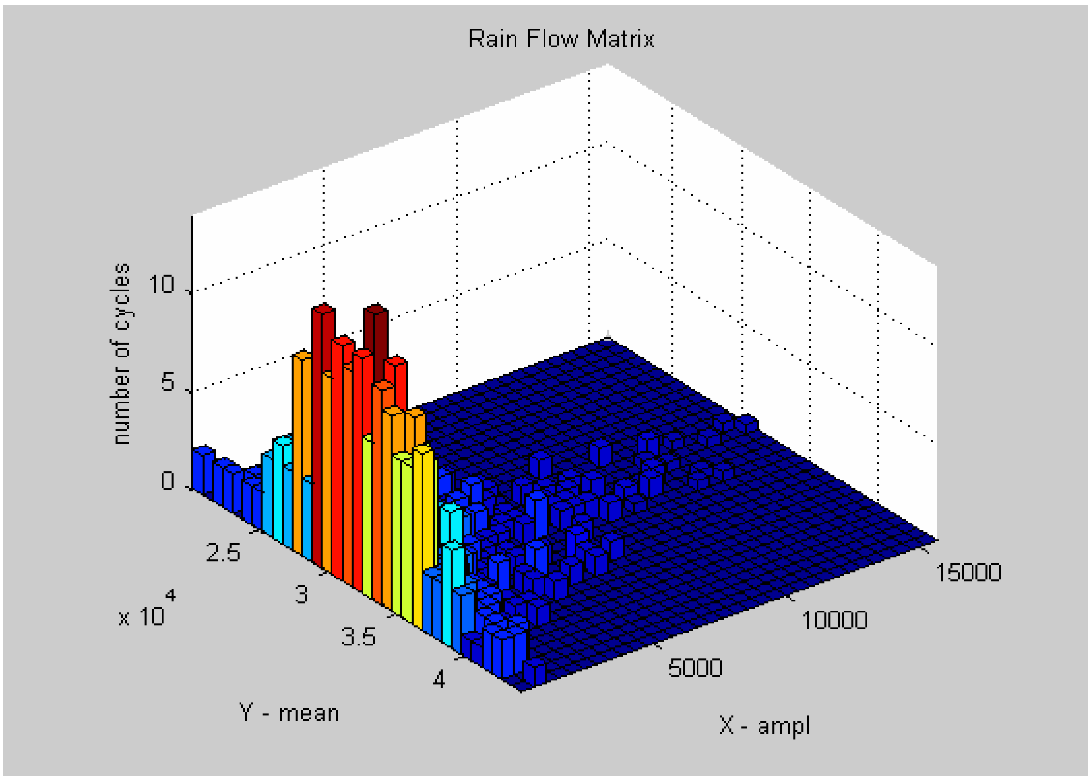


Figure 10. 3D Histogram of Amplitudes and Mean Values

Dirlik

Description

The Dirlik method is a mathematical tool which can be used when performing fatigue analysis. To use this method, signals to be analysed must first be transferred from the time domain to the frequency domain where they are presented in the form of a power spectral density (PSD) function. Stress analysis in the time domain is ideally suited to signals representing periodic or continuous stress loading but for random stress loading data, prohibitively large time records are often required for an accurate analysis. There are distinct advantages in performing stress analysis in the frequency domain rather than the time domain where a random stress loading signal forms the basis for analysis. The main advantage is much less time intensive calculations although this comes at a cost of slight loss in accuracy.

Any fatigue analysis process begins with the response of the structure or component. This is normally expressed as a time history of stress or strain. Stress cycles in the time history result in fatigue and the most important aspects of these cycles are the stress amplitude ranges and the mean stress values. These values are normally analysed using Rainflow Cycle Counting which is described in the previous chapter. The time history data can be converted to data in the frequency domain by using a variation of the Fourier Transform. In both formats, the y-axis displays the signal amplitude but in the frequency domain, the x-axis represents the signal frequency as opposed to time. The Fourier Transform effectively breaks the signal down into discrete sinusoidal waves. These waves vary in frequency, phase and amplitude and form the original time signal again when combined using an inverse Fourier Transform.

The Dirlik procedure was developed during the 1980's as a response to a need for effective fatigue analysis methods within the offshore oil industry. Oil platforms were becoming very large and were subject to random stress loadings at sea. The resultant Dirlik procedure was found to have much wider applications than previous methods and was also found to be very accurate when compared to other methods. The Dirlik method consists of a series of calculations which are based on four moments of area of the PSD function and was taken from Halfpenny[1]. These moments of area are m_0, m_1, m_2 and m_4. The nth moment of area is calculated as

$$m_n = \int f^n . G(f)\, df$$

where G(f) is the PSD function and f is the frequency in Hertz.

The expected number of peaks E[P] is calculated as

$$\mathrm{E[P]} = \sqrt{\frac{\mathrm{m}_4}{\mathrm{m}_2}}$$

The Dirlik method is calculated as

$$x_m = \frac{m_1}{m_0} . \sqrt{\frac{m_2}{m_4}}$$

$$g = \frac{m_2}{\sqrt{m_0 \cdot m_4}}$$

$$D_1 = \frac{2 \cdot (x_m - g^2)}{1 + g^2}$$

$$D_2 = \frac{1 - g - D_1 + D_1^2}{1 - R}$$

$$D_3 = 1 - D_1 - D_2$$

$$R = \frac{g - x_m - D_1^2}{1 - g - D_1 + D_1^2}$$

$$Q = \frac{1.25 \cdot (g - D_3 - D_2 \cdot R)}{D_1}$$

$$Z = \frac{S}{2 \cdot \sqrt{m_0}}$$

$$p(S) = \frac{\frac{D_1}{Q} \cdot e^{\frac{-Z}{Q}} + \frac{D_2 \cdot Z}{R^2} \cdot e^{\frac{-Z^2}{2 \cdot R^2}} + D_3 \cdot Z \cdot e^{\frac{-Z^2}{2}}}{2 \cdot \sqrt{m_0}}$$

$$N(S) = E[P] \cdot T \cdot p(S)$$

Where S is the stress range in N/mm^2, T is the time in seconds and $N(S)$ is the number of stress cycles calculated for stress range S.

Dirlik Coding

Although the Dirlik procedure may be complex in appearance, it is relatively simple to program in Matlab. A copy of the final Dirlik Matlab functions can be found in appendices. Once the Dirlik procedure had been programmed, it was decided to treat the information provided by it in more or less the same way as the information provided by the Rainflow counting algorithm since they were both cycle counting algorithms. So the stress amplitudes were divided into 50 range levels (this was the only real difference since the Rainflow algorithm had no restriction on the number of range levels) and the number of occurrences of each range were calculated. The number of range levels used appeared to affect the results more than was expected but it was decided to stick to 50 as this seemed to be the number of range levels used by FLEX in its rainflow counting

algorithm although FLEX seemed to combine 2 or mo re range levels into one for reasons unknown.
The information generated by the Dirlik method for each wind speed was subjected to the same formula as the Rainflow counting information, i.e.

$$R_{eq} = \left(\frac{\sum n_i R_i^m}{n_{eq}} \right)^{1/m}$$

Where R_{eq} represents the damage equivalent, n_i is the number of cycles of magnitude R_i over i load range levels (usually 30–50), n_{eq} is a constant representing the corresponding number of load ranges and m is the Wöhler curve exponent. The damage figures for each wind speed were then weighted according to the Weibull distribution and adjusted to represent a 20 year period.

After some manipulation of the supplied Dirlik algorithm to account for unit differences, it was found that the Dirlik algorithm was able to produce results with reasonable accuracy when compared with the Rainflow counting algorithm. These results compared favourably with previous studies by Halfpenny[1] which had declared successful results when the Dirlik method was, on average, within 4% of a time domain based stress cycle counting algorithm.

Results

The results of this study were generally quite good. Programming the Rainflow counting method was fairly straightforward and the results provided by that were identical to the results given by FLEX. This was to be expected since FLEX also uses a Rainflow counting algorithm in its fatigue analysis. Programming the Dirlik procedure was less straightforward as there were several unknowns involved. The actual equations based on the four moments of area of the power spectral density were, despite their complexity, simple enough to program and were rigorously checked for errors. The problems then arose when deciding what to do with the information provided by the Dirlik algorithm. It was decided to simply treat the Dirlik information in the same way as the information from the Rainflow counting algorithm. The stress levels were raised to the Wohler curve exponent (m), multiplied by the number of cycles at that level and the relevant Weibull distribution coefficient. These figures were divided by the Neq figure, raised to the power of the reciprocal of the Wohler exponent and summated. In theory this process should have provided similar results to the rainflow process since one cycle counting algorithm was being compared with another in exactly the same way but the results were not similar. It was not immediately apparent why this was the case. There was a likelihood of different units used between the power spectral density calculation in Matlab and the units provided by the Dirlik method which measured stress in N/mm^2. Since very little was known about the Dirlik procedure other than the supplied equations, it was decided to add a series of factors to the Dirlik calculation in order to replicate the Rainflow results. It was not expected to be able to match the Rainflow results exactly but if an acceptable level of accuracy could be sustained for different values of m then it could be reasonably assumed that there was a conflict concerning the units and that applying a factor to resolve this difference was, in fact, a legitimate method of correction. Should a reasonable level of accuracy not be achieved then there were several possible reasons for this including incompatible data sets selected for analysis and the possibility of more appropriate weightings being placed in different parts of the equations.

30 data sets were chosen at random to be analysed. This was reduced to 18 after several were removed because it was thought that they were unsuitable and would give pointless results when compared. One was removed specifically because it appeared to give rogue

results. Although this did not guarantee absolute suitability of the remaining data sets, it was still desirable to keep the selection as random as possible and not ‘cherry pick’ data sets for analysis although it was later discovered that many of the data sets chosen were not the most accurate available. For each of the data sets, the FLEX results were obtained and the Dirlik analysis performed. The Rainflow result given by FLEX for each data set was plotted against the Dirlik estimation and these plots can be seen in appendix 3. The results were then compared. Each FLEX result was compared with every other FLEX result and the same process was performed with the corresponding Dirlik results. The comparisons were performed by dividing each result by each of the other results. They were then plotted against each other in a scatter graph. If the Dirlik ratios were consistently similar to the FLEX ratios then it could be reasonably concluded that the Dirlik procedure provided a very good estimation of fatigue damage. When the results were plotted it appeared that they gave very poor results for very low m figures. Figure 10 shows the Dirlik comparisons against the FLEX comparisons for m = 3.

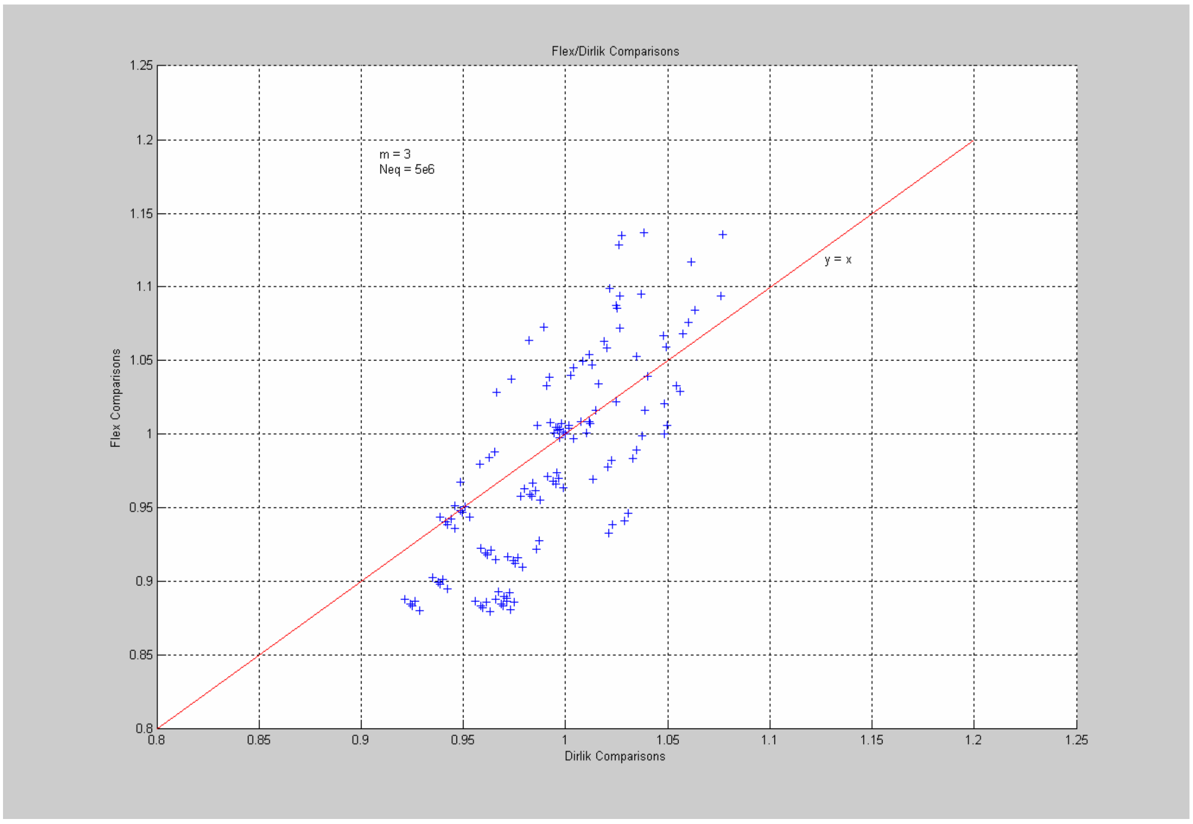


Figure 11. FLEX/Dirlik comparison (m=3)

It was hoped that the points would roughly follow a line equivalent to y=x (shown in red) and while this basic pattern was observed, it was thought that the points were too varied to declare a positive result.

The results for m = 4, however were much better. These can be seen in figure 11 and the results for m = 6, 8 10, 12 can be seen in figures 12 to 15.

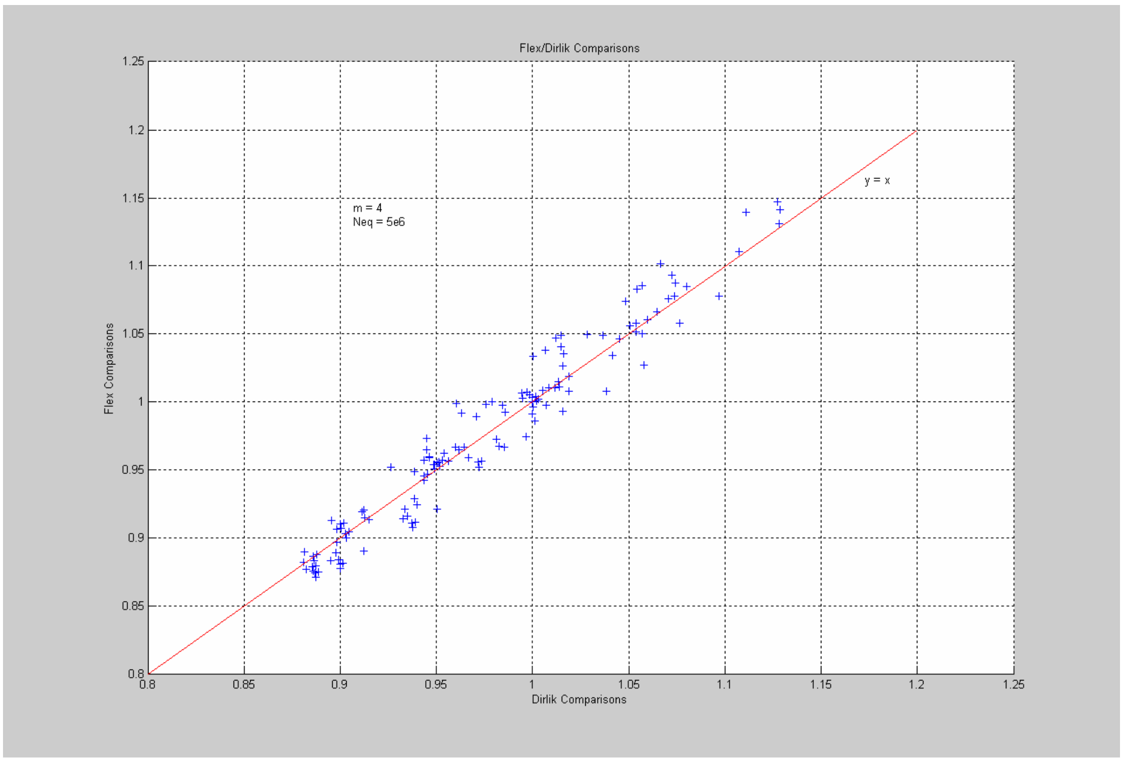


Figure 12. FLEX/Dirlik comparison (m=4)

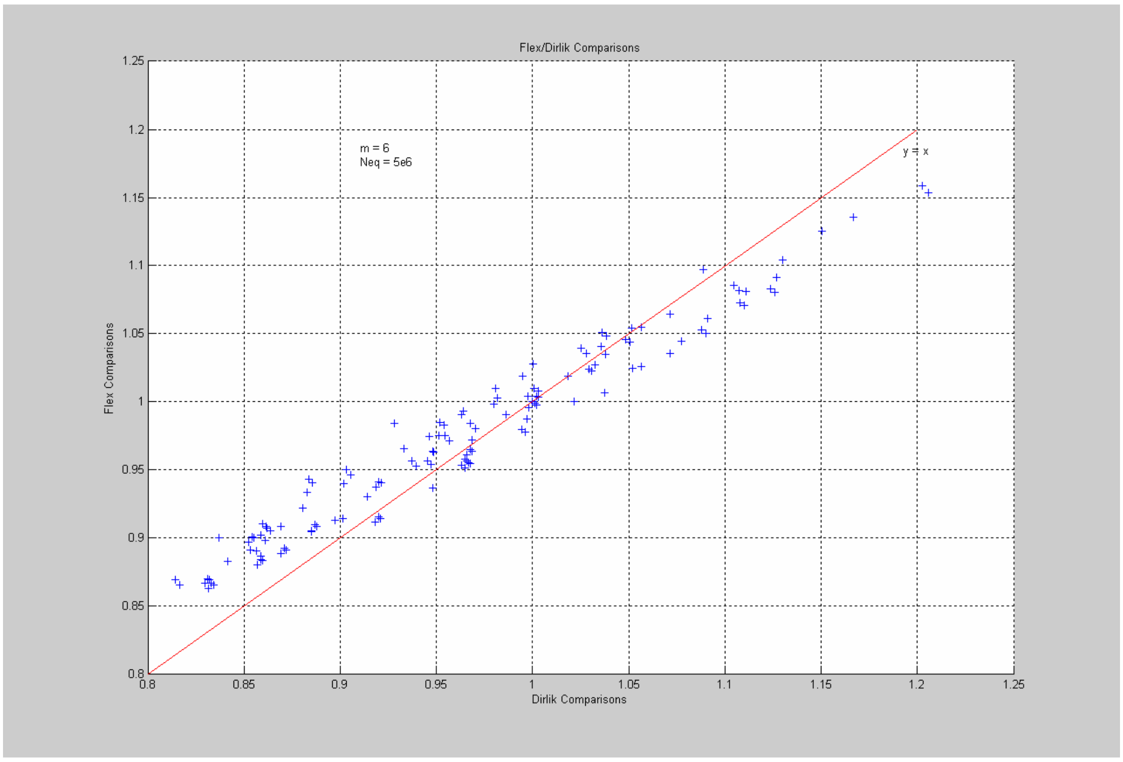


Figure 13. FLEX/Dirlik comparison (m=6)

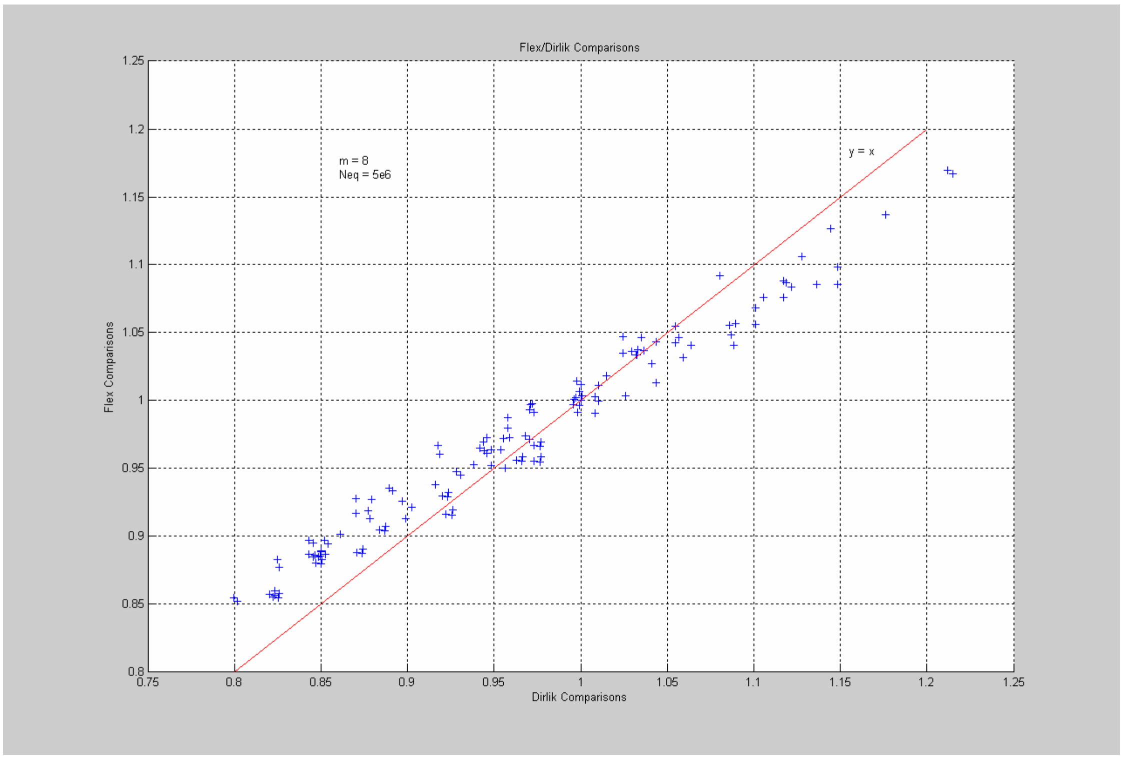


Figure 14. FLEX/Dirlik comparison (m=8)

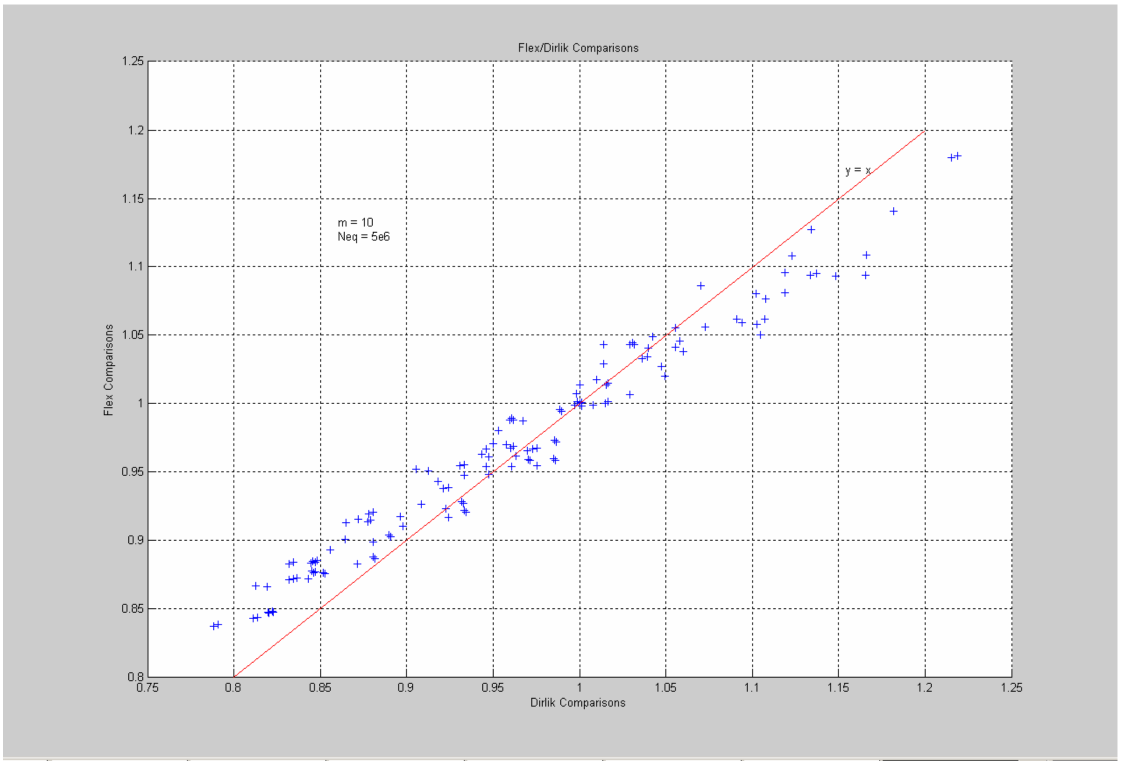


Figure 15. FLEX/Dirlik comparison (m=10)

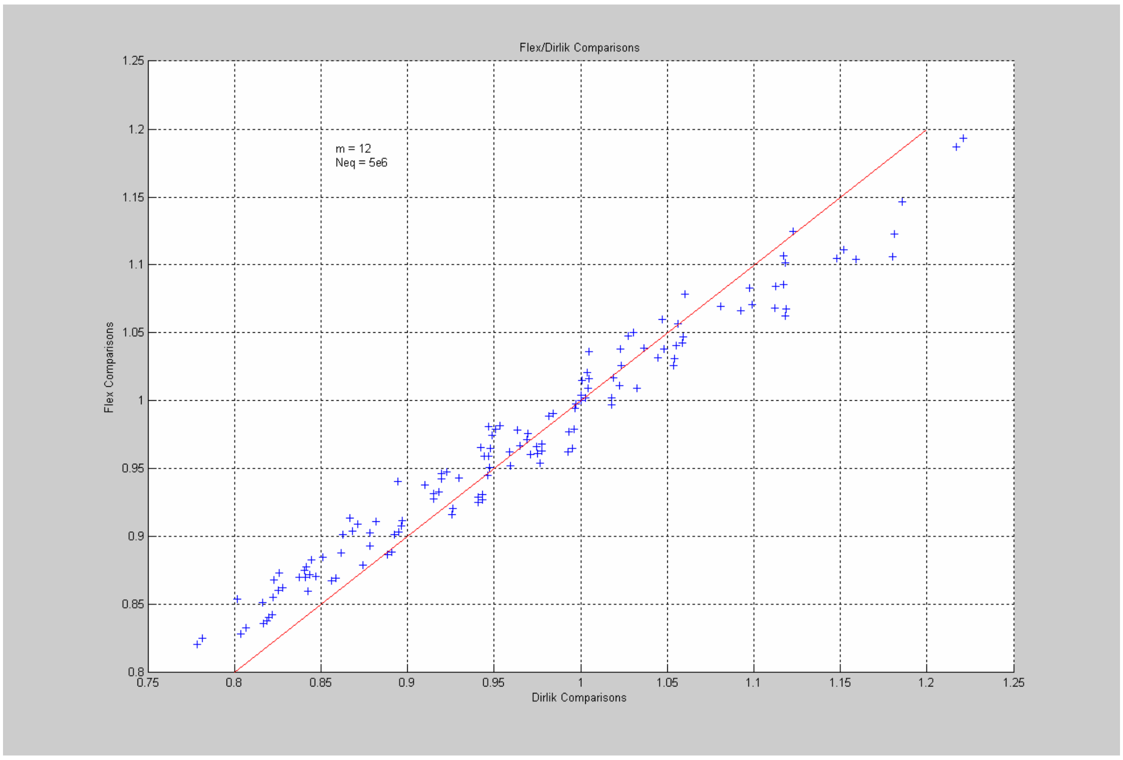


Figure 16 FLEX/Dirlik comparison (m=12)

These results were much more encouraging and show that the Dirlik procedure does in fact provide a very good approximation for fatigue damage for higher values of m. What was less clear was why such relatively poor results were obtained for very low values of m. Since the results for higher m values are consistently good, it can be reasonably assumed that there are no undiscovered errors within the Dirlik program itself and that the process of placing factors within the program to weight the results was correct. The most likely reasons for this are, therefore, unsuitable data sets or poorly placed weightings. Closer examination of the points furthest from the line y = x revealed that almost all of the data sets were commonly involved but to differing degrees. No pattern could be established between these data sets however and it was thought that enough sets of data had been removed already and to remove more would reduce the random nature of their selection and subsequently the validity of the tests. The investigation therefore proved largely inconclusive however it was noted that there were accuracy issues with some data sets. The possibility that the weighting factors were wrongly placed within the program remained and this was examined by placing different factors at different points in the calculation. The number of different permutations possible here was very large and so some basic rules had to be applied in order to reach a quick and reasonable conclusion. Factors were subsequently moved between key points in the equations and the effects on the positioning of the scatter graph points rather than the magnitudes of the results were observed. It was decided that the current placing of the factors produced the best possible results.

Further numerical investigation of the results indicated that the observed results for low values of m were perhaps not as bad as they seemed. Previous studies by Halfpenny[1] indicated that the Dirlik approach showed an average discrepancy of only 4% from fatigue life calculated in the time domain. This discrepancy arises from a simple summation of comparisons and performing the same summation here gives an average discrepancy of 5.8% and a best of 0.4% at m = 12. These results can be seen in column 1 of the table in figure 16. This method of simply taking the average of the comparisons can be misleading though as positive and negative differences can cancel each out. A more indicative method is to average the absolute differences between the results and this can be seen in column 2 of the table. Here, the average is only slightly higher at 6.4% with a best of 1.5% at m = 12. A better method again of testing the variance of the results is to calculate the standard deviation. Columns 3 and 4 of the table show 2 standard deviation measurements. The first, headed Std Dev 1, is the standard deviation of the direct ratios between the FLEX results and the Dirlik results. These seem to be rather good and are consistently around 0.027 for values of m = 6 and above with an average of 0.022. The second column of standard deviations is based on the process of comparing each FLEX result with every other FLEX result and doing the same with the respective Dirlik results. The table shows the standard deviation of the ratio between the respective FLEX and Dirlik comparisons under the heading Std Dev 2. This shows a best standard deviation of 0.025 and an average of 0.029. Graphical representation of these results can be seen in appendix 4.

m	Simple Average	Absolute Average	Std Dev 1	Std Dev2
3	0.006	0.0292	0.0352	0.0479
4	0.1597	0.1597	0.0145	0.017
6	0.1074	0.1063	0.0235	0.027
8	0.0503	0.0503	0.0223	0.0279
10	0.02	0.0247	0.0204	0.02688
12	0.0039	0.0154	0.0186	0.0254
Ave	0.057883333	0.064266667	0.022417	0.02868

Figure 16. Table of Results

Considering the apparent poor quality of the random data sets which were used, these results must be considered as rather good and demonstrate that with some manipulation, the Dirlik procedure can be confidently used as a method of predicting fatigue damage in wind turbines.

Discussion

The rainflow counting method of fatigue analysis is widely used and its accuracy is generally accepted to be very high. As such it was used as the control model here and the Dirlik method was judged against it. The rainflow counting method does, however have limitations and the most obvious of these is that it is used on data based in the time series domain. Its accuracy, while remarkably high, can only be guaranteed with any real certainty on data which is periodic or has a short time span. Its suitability for analysis of stress loading which is random and has a very long time span (typically several years) could feasibly be questioned by those seeking a high level of speed and accuracy in fatigue prediction. Extrapolation of random stress loading of data that lasts a few minutes to give fatigue estimations for a period of 20 years will inevitably lead to inaccuracies within the results due to the relatively unpredictable nature of the loading, in this case wind. These inaccuracies may be offset somewhat by the relative accuracy of the rainflow counting algorithm when compared with other fatigue estimation algorithms. It is also known that wind speed levels in the longer term follow a Weibull distribution and if average wind speed for the site is known then a suitable Weibull distribution can be applied for each of the wind speeds within the operational range of the wind turbine in question. Use of these methods is standard within the wind industry and the predictions given by them are regarded to be of acceptable accuracy. The dangers, however, lie in the fact that regardless of the statistical techniques being applied, random loading data is being extrapolated from a few minutes to several years and that the statistical weightings applied rely on the average wind speed remaining relatively unchanged for the 20 year period. These factors may or may not reduce the ultimate accuracy of the fatigue predictions but they certainly affect the level of certainty that can be applied to them and it is a bit impractical to wait 20 years to test their accuracy.

Another drawback of the Rainflow counting method is the fact that the calculations are computationally time intensive and this is a very important issue at the design stage. A fatigue analyst who is testing the effects of many small parameter alterations is likely to be prepared to sacrifice a small amount of accuracy in what are, after all estimations, in order to gain a huge reduction in the time taken to carry out the procedures. If the fatigue analysis process can also be incorporated into the control design stage then this provides a further bonus in terms of time and efficiency of the design process.

One method of resolving these issues is to convert the stress loading data from the time series domain to the frequency domain for analysis. Here it is possible to categorise the random stress loadings with power spectral density functions. The purpose of this project was to test the validity of the Dirlik method which predicts fatigue using data presented in the frequency domain. The Dirlik method was specifically developed in the 1980's for the offshore oil industry. The fact that it analysed data in the frequency domain instead of the time domain meant that, theoretically, fatigue caused by random loadings could be more quickly predicted over very long periods of time albeit with a small reduction in accuracy. This would make the Dirlik method preferable to time series based methods such as the Rainflow method as long as the relative accuracy of the method itself could be established.

Some difficulties were encountered during the programming and subsequent testing of the Dirlik method and not all of these were resolved with absolute certainty. The main problems centred around the suitability of the random sets of data that were chosen for analysis and the weightings that had to be applied at various points of the Dirlik calculations. It was considered desirable to select data sets at random and simply compare results between each. If data sets had been "cherry picked" then the validity of the results could be undermined. If the ratios between the FLEX results for different data sets could be reproduced using the Dirlik method with sufficient consistency then that would be a positive result even if the actual figures were different. It was later discovered that many of the data sets chosen were 'short' data sets and the accuracy of the FLEX figures were not particularly great. It is highly likely that this caused the relatively poor accuracy for data that was tested at a value of m = 3 and the subsequent improvement in results as m increased. The validity of attaching weighting factors to various stages in the Dirlik calculation was upheld. The whole point of the project was to emulate the fatigue estimations given by the Rainflow counting algorithm using a stress cycle counting algorithm which has its basis in the frequency domain and this has been achieved with reasonable accuracy. The results show that the Dirlik method produced, on average, fatigue estimations that were within 6.4% of the estimations provided by FLEX. This must be considered a very positive result considering the reduction in calculation time over the Rainflow method. These results can be easily seen in appendices 3 and 4. To create these charts, each FLEX estimation was compared with every other FLEX estimation and the same process was applied to the Dirlik estimation. The respective comparisons were then plotted against each other in the form of scatter charts shown in appendix 3 where perfect matches would be expected to follow the line y=x. The Dirlik comparisons were then divided by their respective FLEX comparisons and these results are shown in the graphs in appendix 4 where perfect matches would be expected to follow the line y=1. The reason for this form of comparison, as opposed to the table in the previous chapter which merely shows a comparison between each FLEX and Dirlik result, is that consistency here would indicate a much more robust result. The scatter charts in appendix 3 appear to show very good comparisons in all cases except m=3, however this view is somewhat deceptive due to the scale of the graph. A clearer picture can be seen in appendix 4 which essentially depicts the same results in a slightly different manner. Here it can be clearly seen that for m=3, the Dirlik comparisons are within 10% of the FLEX comparisons. The best results are for m=4 where the Dirlik comparisons are all within 4% of the FLEX comparisons while the other values of m are all reasonably consistent around 6-7%. These results are, in fact, comparable with the direct comparisons tabled in the previous chapter.

While these results are very encouraging some further comment must be made about the data sets. As previously stated, the data sets were chosen at random and contained a high proportion of short data sets with slightly questionable accuracy. This was not known at the time of selection. On initial analysis of results, some data sets were found to be common to extreme rogue results and these were legitimately removed, however it was desirable to minimise the practise of removing ‘bad’ data sets and so only a very few were eliminated in order to keep the data selection as random as possible. Any continuation of this work may consider careful selection of data sets rather than random selection. The reason being that if the accuracy of the data sets can be assured then this factor can be eliminated as a cause for any inaccuracies in the result, particularly for m=3. This was not done here as random selection of data was seen to be fundamental to the validity of the results. Further investigation could also concentrate on the application of weighting factors to the Dirlik calculations. It can be seen that the graphs in appendix 4 have slight periodic characteristics and these could possibly be eliminated with further investigation into the weightings.

Conclusions

1. The Dirlik procedure is much less computationally intensive than the Rainflow counting procedure therefore should produce faster results for fatigue analysts.
2. The Dirlik procedure should lend itself to incorporation into control analysis since it analyses data in the frequency domain.
3. The Dirlik procedure can produce fatigue estimations which are, on average, within 6-7% of the Rainflow counting algorithm.

References

1. *A frequency domain approach for fatigue life estimation from Finite Element Analysis*
Andrew Halfpenny
nCode International Ltd, Sheffield UK

2. ASME 1985, *Standard Practices for Cycle Counting in Fatigue Analysis*, ASTM E1049-85(1997),
American Society of Mechanical Engineers, USA.

3. *The Statistical Variation of Wind Turbine Fatigue Loads*
Kenneth Thomsen
Riso National Laboratory, Roskilde, Denmark (1998)

4. *Review of Control Algorithms for Off-Shore Wind Turbines*
Prof. W.E.Leithead, Sergio Dominguez
University of Strathclyde (2003)

5. *On the Fatigue Analysis of Wind Turbines*
Herbert J. Sutherland
Sandia National Laboratories, Albuquerque, New Mexico (1999)

6. *http://www.bwea.com/ref/faq.html*

7. *http://www.bwea.com/marine/index.html*

8. *http://www.history.rochester.edu/class/solar/solar.htm*

9. *http://en.wikipedia.org/wiki/Rainflow-counting_algorithm*

10. *http://www.c-power.be/applet_mernu_en/index01_en.htm?windturbines /index_wit.htm~right_frame*

11. *http://www.windpower.org/en/tour/econ/offshore.htm*

Bibliography

Fatigue of Metallic Materials
M. Klesnil, P. Lukas
Elsevier

Analysis of the FLEX Model
Prof. W.E.Leithead, Sergio Dominguez
University of Strathclyde (2003)

Analysis of the Fatigue Loading of an Offshore Wind Turbine Using Time and Frequency Domain Methods
Silke Schwartz, Kimon Argyriadis
Germanischer Lloyd Windenergie Gmbh, Johannisbollwerk 6-8, 20459 Hamburg

http://www.absoluteastronomy.com/encyclopedia/F/Fa/Fatigue_(material).htm

http://www.answers.com/topic/fatigue-material

http://www.maths.lth.se/matstat/staff/pj/Research/BELU/Description/

http://ocw.mit.edu/NR/rdonlyres/Materials-Science-and-Engineering/3-11Mechanics-of-MaterialsFall1999/93EAB4DF-4E69-47EE-8FED-7015DDEC2D99/0/fatigue.pdf

Appendices

1a. Rainflow Script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%% Script to perform rainflow
%%%% analysis on FLEX data
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

l=dir('*.int');        % Access .INT files
load Ti;               % load Weibull distribution coefficients
Neq=5e6;
N=3;

for i=1:length(l),              % Call rainflow function
  x=readint(l(i).name)           % to analyse data
  d(i)=raindef(x,N,Ti(i));
end

(sum(d)/Neq)^(1/N)      % Sum results
```

1b. Rainflow Function

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%%%%  Function to perform
%%%%  rainflow analysis
%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function D=raindef(x,M,time)

  Rinx = sig2ext(x.Myt0);       %% Find extremes of data
  Rin = rainflow(Rinx);     %% Perform rainflow analysis on data
  Ramp = Rin(1,:)*2;         %% Isolate amplitudes and
  Rno = Rin(3,:);           %% number of occurrences
  T=max(x.time);

  D=sum((Ramp.^M).*Rno)*time*3600/T;  %% Apply Weibull coefficient and sum
```

2a. Dirlik Function 1

```
function f=loadir(m)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                                          %%%%
%%%%  Script to process Dirlik evaluation    %%%%
%%%%  of Flex results of fatigue damage      %%%%
%%%%  to wind turbines                        %%%%
%%%%                                          %%%%
%%%%  Created by Gordon Day on 18/5/05   %%%%
%%%%                                          %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Neq = 5e6;
%Neq = 1e7;                          %% Select value for Neq
%Neq = 5.8e8;
disp(['Neq = ',num2str(Neq)]);
%m = input(['Please select a value for m (3, 4, 6, 8, 10 or 12)    ']);

if m == 3
   k = 1;
elseif m == 4
   k = 2;
elseif m == 6                %% Ensure correct calculation
   k = 3;                    %% for chosen value of m
elseif m == 8
   k = 4;
elseif m == 10
   k = 5;
elseif m == 12
   k = 6;
else
   error('You must select m from 3, 4, 6, 8, 10, or 12');
end

if Neq == 5e6
   p = 1;                  %% Ensure correct calculation
elseif Neq == 1e7       %% for chosen value of Neq
   p = 2;
elseif Neq == 5.8e8
   p = 3;
end
```

```
cd ('D:\Gordons project\DirlikG');
l = dir;
l.name;
for j = 3:20
   cd(l(j).name);
   cd int;                              %% Loop to access
   q = dir;                             %% and read
   q.name                               %% relevant .int
   DD(j-2) = loadall3(m, Neq);          %% files from
   cd .. ;                              %% data sets and
   cd('rain');                          %% send to loadall function
   FlxDam = ReadFRO('91_Myt0.rfo');
   DamFig(j-2) = FlxDam(p,k);
   cd ('D:\Gordons project\DirlikG');
end

DamFig
DD
for i = 1:length(DD)
   avrge(i) = DD(i)/DamFig(i);
   bestav(i) = 1+abs(1-(DD(i))/(DamFig(i)));
end
Avrge = mean(avrge)                  %% Calculate and
Bestav = mean(bestav)                %%
figure;                              %% graph
scatter(DD,DamFig);                  %%
title('Flex/Dirlik Comparisons');    %%
xlabel('Dirlik ');                   %% results
ylabel('Flex ');                     %%
Results(1,:) = DamFig;               %% using a
Results(2,:) = DD;                   %%
R = Results';                        %% variety of
Res = sortrows(R);                   %%
Results = Res';                      %% methods
Damfig = Results(1,:);
DDD = Results(2,:);
figure;
plot(Damfig,DDD);
for i = 1:17
   RD(i) = DamFig(i+1)/DamFig(1);
   DirD = DD(i+1)/DD(1);
end
Rat = DamFig./DD
StanDev1 = std(Rat)
DD2 = DD;
```

```
DF2 = DamFig;

count = 0;
figure;
for i = 1:17                          %% The following nested
   for j = (i+1):18                   %% loops were created
      DD3 = DD2(j)/DD2(i);            %% To identify
      DF3 = DF2(j)/DF2(i);            %% rogue data sets

      if DD3 > 1
         if DF3 < 1
            Diff = abs(DD3-DF3)
            if Diff >= 0.05
               disp(['DD3 = ',num2str(DD3)]);
               disp(['DF3 = ',num2str(DF3)]);
               disp(['i = ',num2str(i)])
               disp(['j = ',num2str(j)]);
               disp(['Difference = ',num2str(Diff)])
               disp(' ');
               disp(' ');
            end
         end
      elseif DD3 < 1
         if DF3 > 1
            Diff = abs(DD3-DF3)
            if Diff >= 0.05
               disp(['DD3 = ',num2str(DD3)]);
               disp(['DF3 = ',num2str(DF3)]);
               disp(['i = ',num2str(i)])
               disp(['j = ',num2str(j)]);
               disp(['Difference = ',num2str(Diff)])
               disp(' ');
               disp(' ');
            end
         end
      end

      scatter(DD3,DF3,'+');
      hold on;
      count = count + 1;
   end
end

x = 0.8:0.1:1.2;

for i = 1:length(x)
```

```
    y(i) = x(i);
end
plot(x,y,'r');

count
grid;
title('Flex/Dirlik Comparisons');
xlabel('Dirlik Comparisons');
ylabel('Flex Comparisons');

count = 0;                              %% Calculate and plot
figure;                                 %% Comparison figures
for i = 1:17
    for j = (i+1):18
        count = count + 1;
        DD4 = DD2(j)/DD2(i);
        DF4 = DF2(j)/DF2(i);
        DRat(count) = DD4/DF4;
        %plot(count,DRat,'+');
        hold on;
    end
end
plot(DRat);
Deviate = std(DRat)

x=0:1:(count-1);
for i = 1:count
    y(i) = x(i)*0+1;
end
%plot(x,y,'r');

ylim([0.5,1.5]);
title('Flex/Dirlik Comparisons');
xlabel('Index');
ylabel('Flex/Dirlik');
```

2b. Dirlik Function 2

```
function Dd = loadall3(M, Neq)

%%%%%%%%%%%%%%%%%%%%%%%%
%%%%  Function to process data     %%%%
%%%%  from Loadir script and        %%%%
%%%%  send to Dirlikdef function    %%%%
%%%%%%%%%%%%%%%%%%%%%%%%

A=dir('*.int');                    %% Access directory of .int files
load Ti;                           %% Load Weibull distribution figures

A.name
for i=1:length(A),                 %% Loop to send
    x=readint(A(i).name);          %% all .int files
    d(i)=Dirlikdef(x,M,Ti(i));     %% to Dirlikdef function
end
Dd=(sum(d)/Neq)^(1/M);             %% Final damage calculation
disp('loadall finished')
```

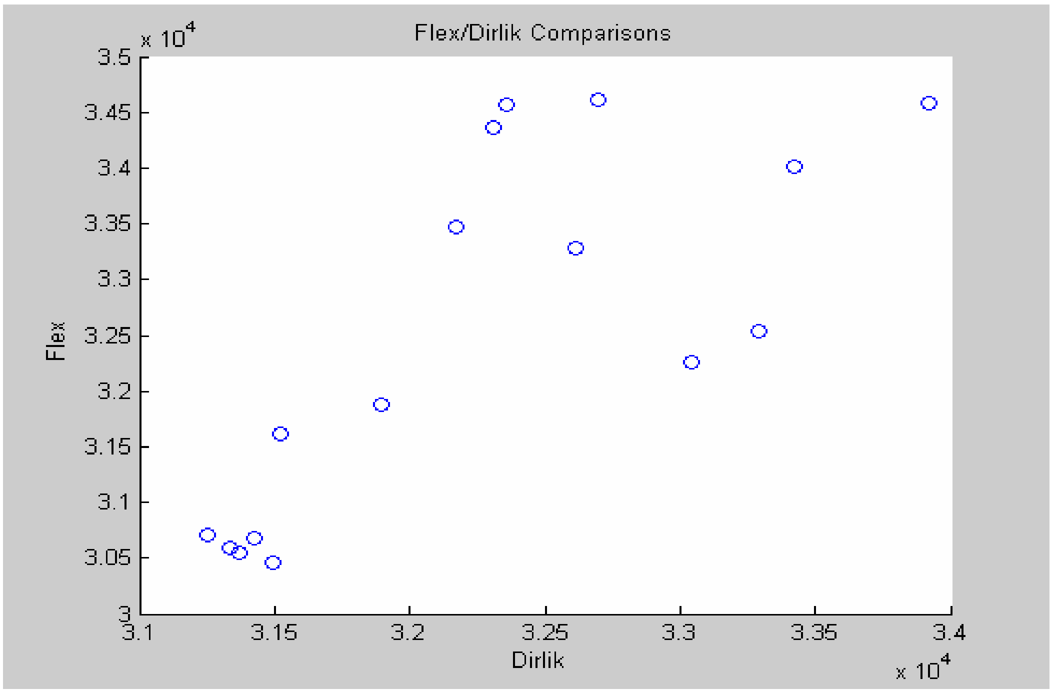
2c. Dirlik Function 3

```
function D = Dirlikdef(x,m,Weibcoef)

%%%%%%%%%%%%%%%%%%%%%%%%
%%%%  Function to perform Dirlik    %%%%
%%%%  analysis on data sent from     %%%%
%%%%  loadall3 function              %%%%
%%%%%%%%%%%%%%%%%%%%%%%%

f = 1/(x.time(2)-x.time(1));
[xs,fxs]=pwelch(x.Myt0,2^11,2^10,2^11,f);               %% Generate PSD using Welch's
method
Mzero = simp(fxs,xs);                                %%
Mone = simp(fxs,fxs.*xs);                            %% Calculate 0th - 4th...
Mtwo = simp(fxs,(fxs.^2).*xs);                       %% moments of area
Mfour = simp(fxs,(fxs.^4).*xs);                      %%

Ep = sqrt(Mfour/Mtwo);                               %%
time = max(x.time);                                  %%
stressmax = (max(x.Myt0)-min(x.Myt0));               %% Set maximum value in stress range
slightly above value of highest stress peak
lambda = Mtwo/(sqrt(Mzero*Mfour));                   %%
Xm = (Mone/Mzero)*(sqrt(Mtwo/Mfour));                %% Generation of...
```

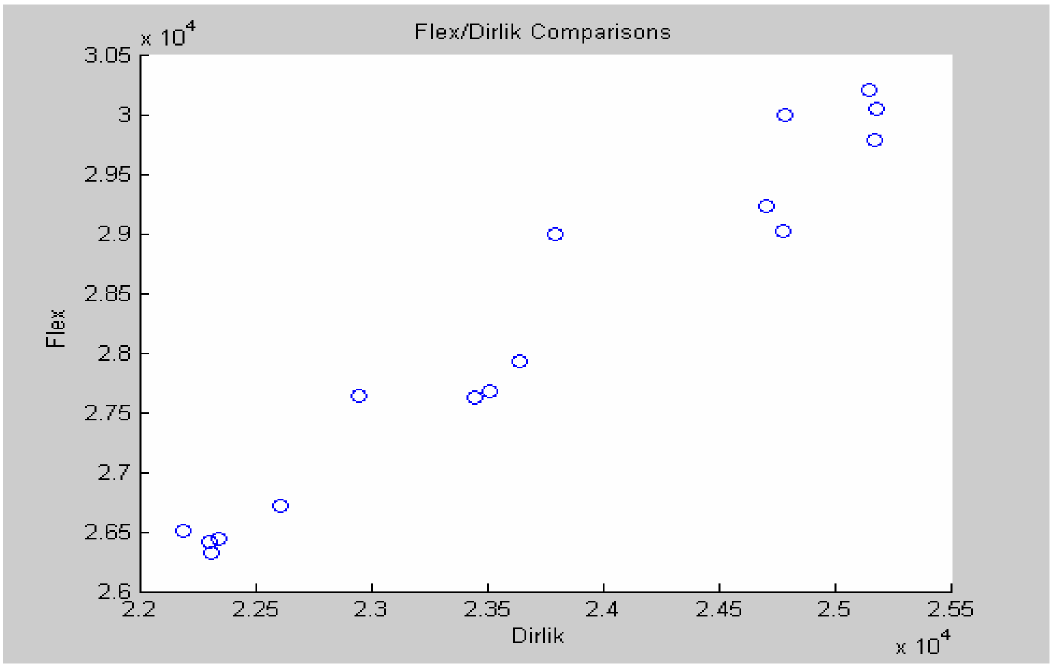
```
Done = 2*(Xm-lambda^2)/(1+lambda^2);                %% calculation constants
R = (lambda-Xm-Done^2)/(1-lambda-Done+Done^2);      %%
Dtwo = (1-lambda-Done+Done^2)/(1-R);                %%
Dthree = 1-Done-Dtwo;                               %%
Q = 1.25*(lambda-Dthree-Dtwo*R)/Done;
stresstep = stressmax/50;
stresslev = zeros(1,50);
for b = 1:50;
   stresslev(b) = b*stresstep;                      %% Loop to calculate ranges of
   Z(b) = 0.5*stresslev(b)/(sqrt(Mzero));           %% stress and expected occurrence
   Ps(b)          =          ((Done/Q)*exp((-Z(b))/Q)+(Dtwo*(Z(b))/(R^2))*exp((-
((Z(b))^2))/(2*R^2))+Dthree*Z(b)*exp(-((Z(b))^2)/2))/(2*sqrt(Mzero));
   Ns(b) = Ep*631152000*time*Ps(b)*(2*pi)/1000;
   %Ns(b) = Ep*time*Ps(b);
end
stresstotD = 0;
D = sum(stresslev.^m.*Ns)*Weibcoef*3600/time/1000;


%D=sum((Ramp.^m).*Rno)*time*3600/T;
```

3. Scatter Plots of FLEX Vs Dirlik

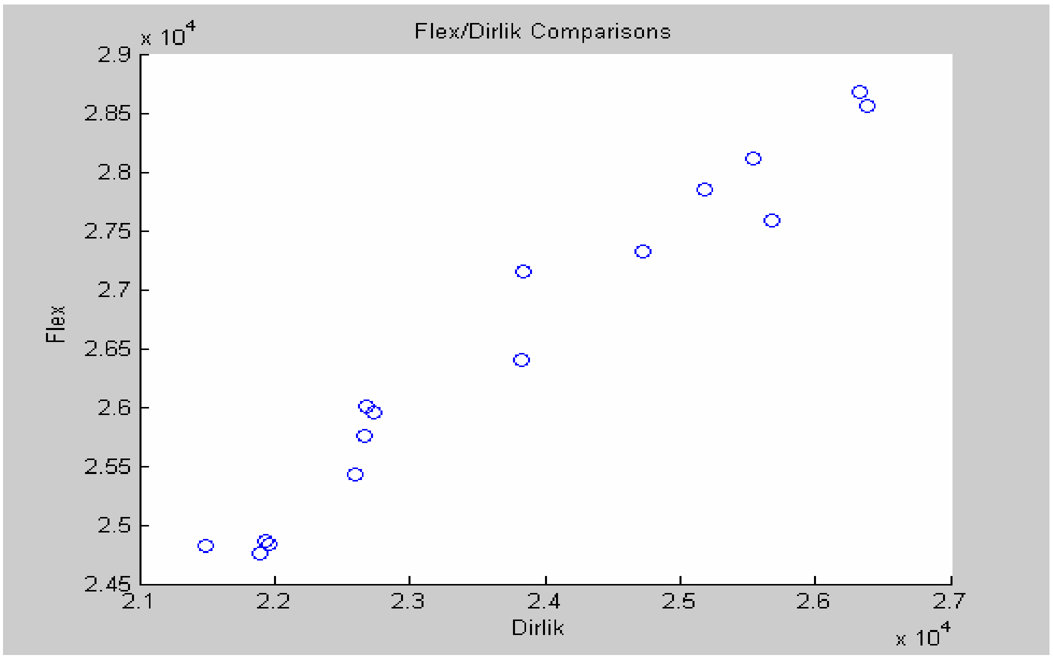
These plots are designed to be an indicator of accuracy of the Dirlik procedure when compared to the Rainflow counting results supplied by FLEX. Each FLEX estimation was compared with every other FLEX estimation by division and the same procedure was undertaken with the Dirlik estimations. The respective FLEX and Dirlik comparisons were then plotted against each other in the form of scatter graphs. Perfect results would be expected to follow the line y=x. This form of comparison is regarded to be a more robust indicator of accuracy than simply comparing the FLEX estimation of a data set with the respective Dirlik estimation.



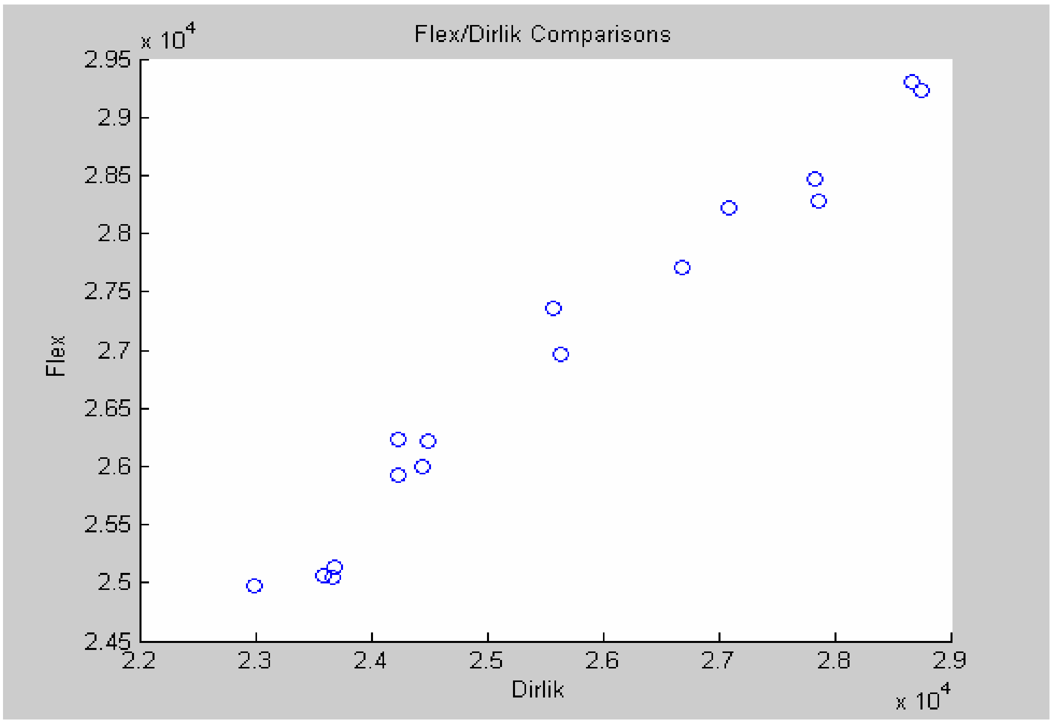
m = 3


Flex/Dirlik Comparisons
x 10^4
3.05
3
2.95
2.9
2.85
2.8
2.75
2.7
2.65
2.6
Flex
2.2
2.25
2.3
2.35
2.4
2.45
2.5
2.55
Dirlik
x 10^4

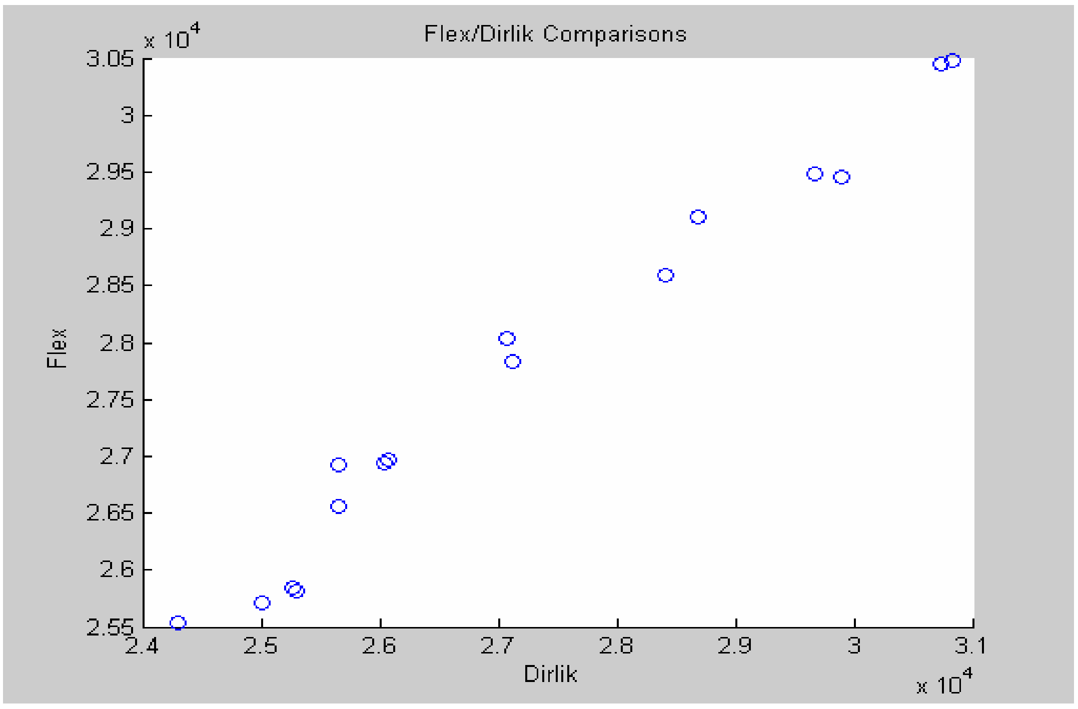
m = 4


Flex/Dirlik Comparisons
x 10^4
2.9
2.85
2.8
2.75
2.7
2.65
2.6
2.55
2.5
2.45
Flex
2.1
2.2
2.3
2.4
2.5
2.6
2.7
Dirlik
x 10^4

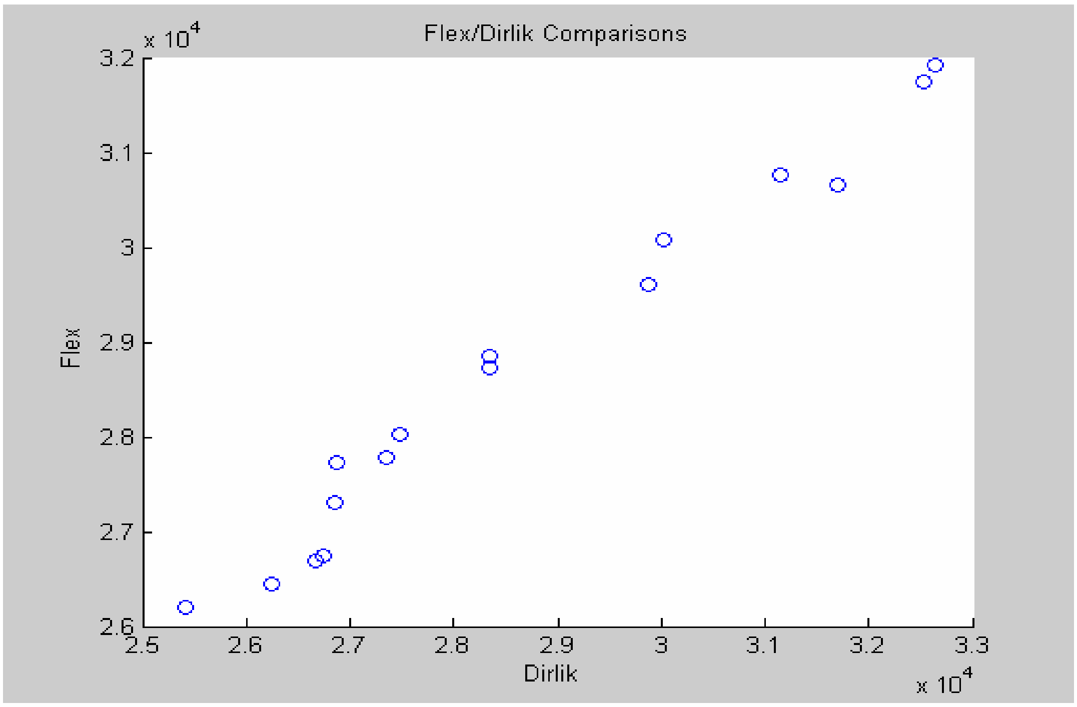
m = 6


Flex/Dirlik Comparisons
x 10^4
2.95
2.9
2.85
2.8
2.75
2.7
2.65
2.6
2.55
2.5
2.45
Flex
2.2
2.3
2.4
2.5
2.6
2.7
2.8
2.9
Dirlik
x 10^4

m = 8


Flex/Dirlik Comparisons
x 10^4
3.05
3
2.95
2.9
2.85
2.8
2.75
2.7
2.65
2.6
2.55
Flex
2.4
2.5
2.6
2.7
2.8
2.9
3
3.1
Dirlik
x 10^4

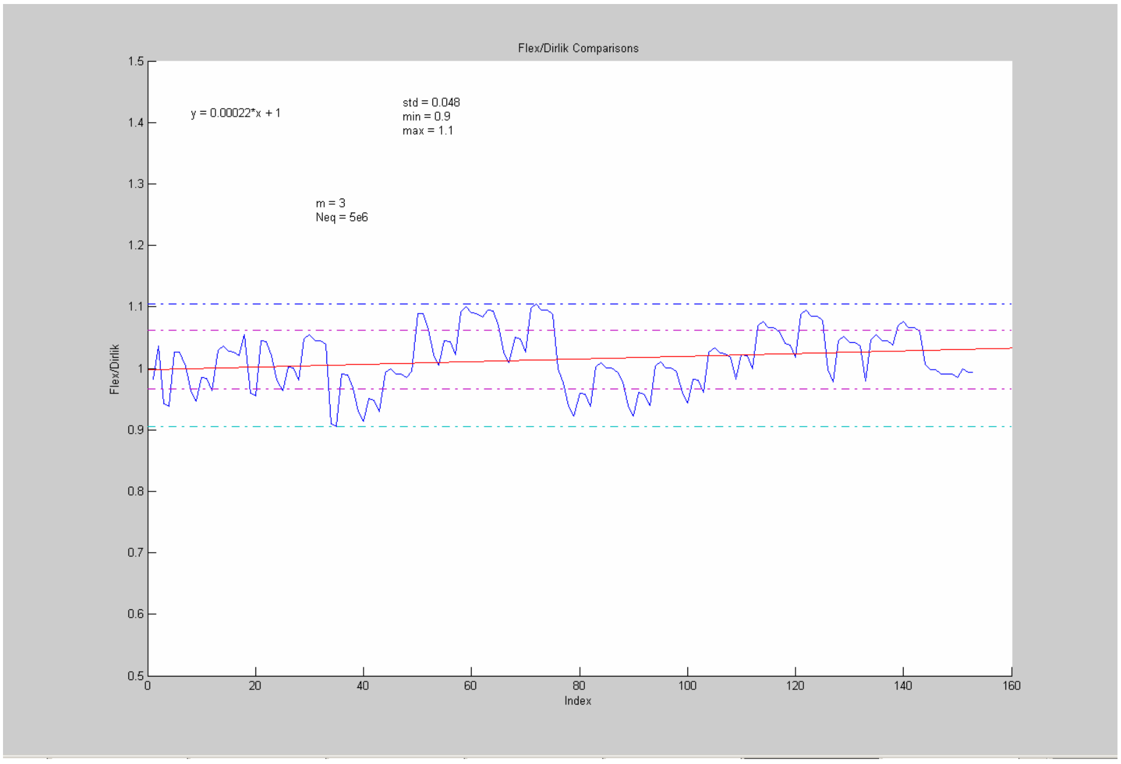
m = 10


Flex/Dirlik Comparisons
x 10^4
3.2
3.1
3
2.9
2.8
2.7
2.6
Flex
2.5
2.6
2.7
2.8
2.9
3
3.1
3.2
3.3
Dirlik
x 10^4

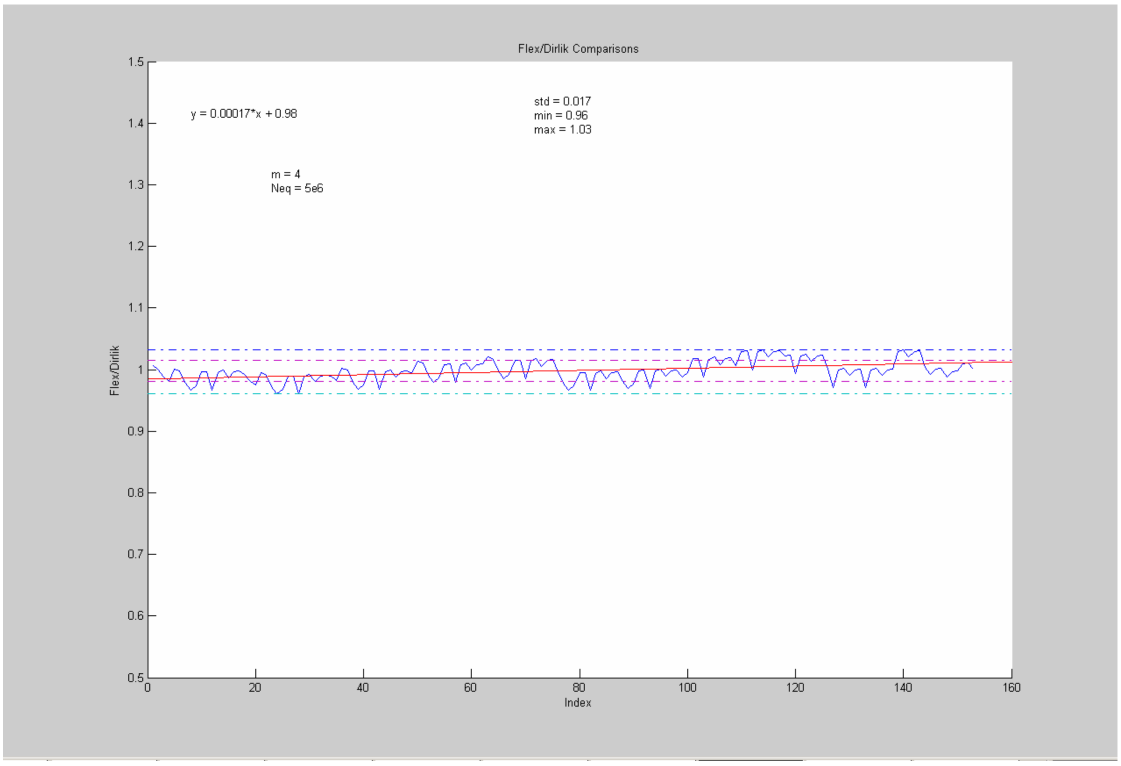
<u>m = 12</u>

4. Graphical Results of FLEX Comparisons Vs Dirlik Comparisons

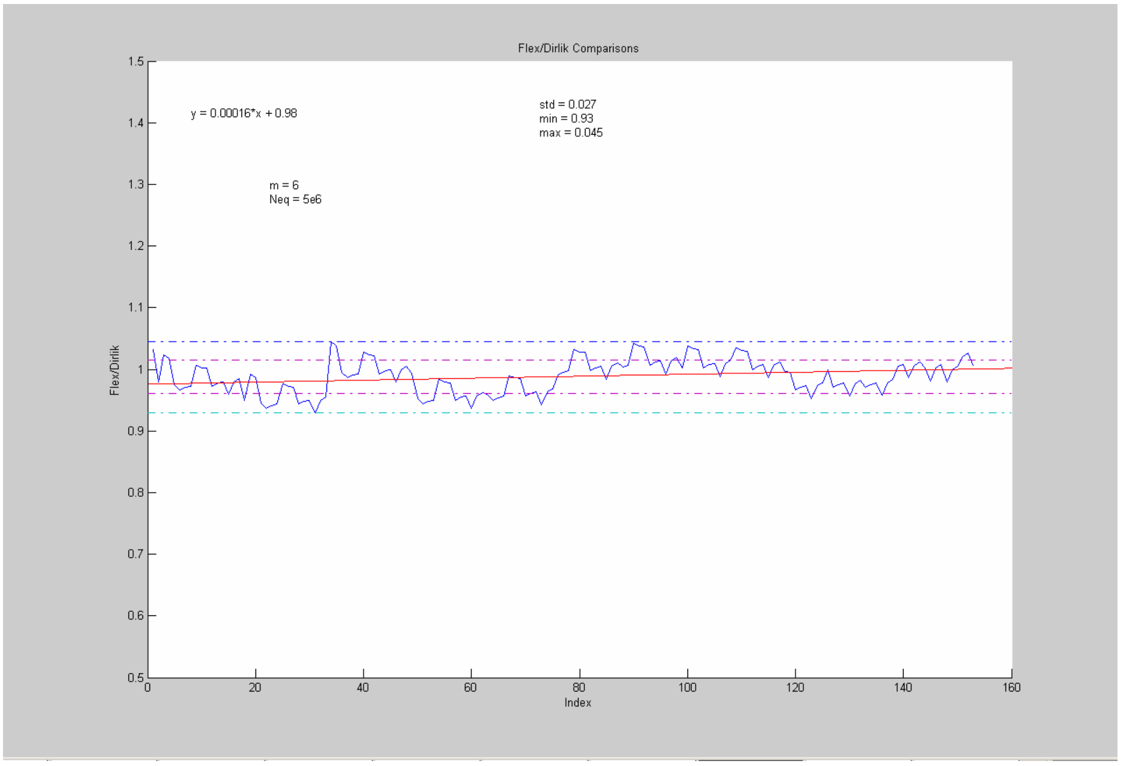
The graphs in this appendix depict the same information as the graphs in appendix 3 but in a slightly different way. The Dirlik comparisons and the FLEX comparisons shown in appendix 3 have been compared by division and plotted. Perfect comparisons would be expected to follow the line y=1.



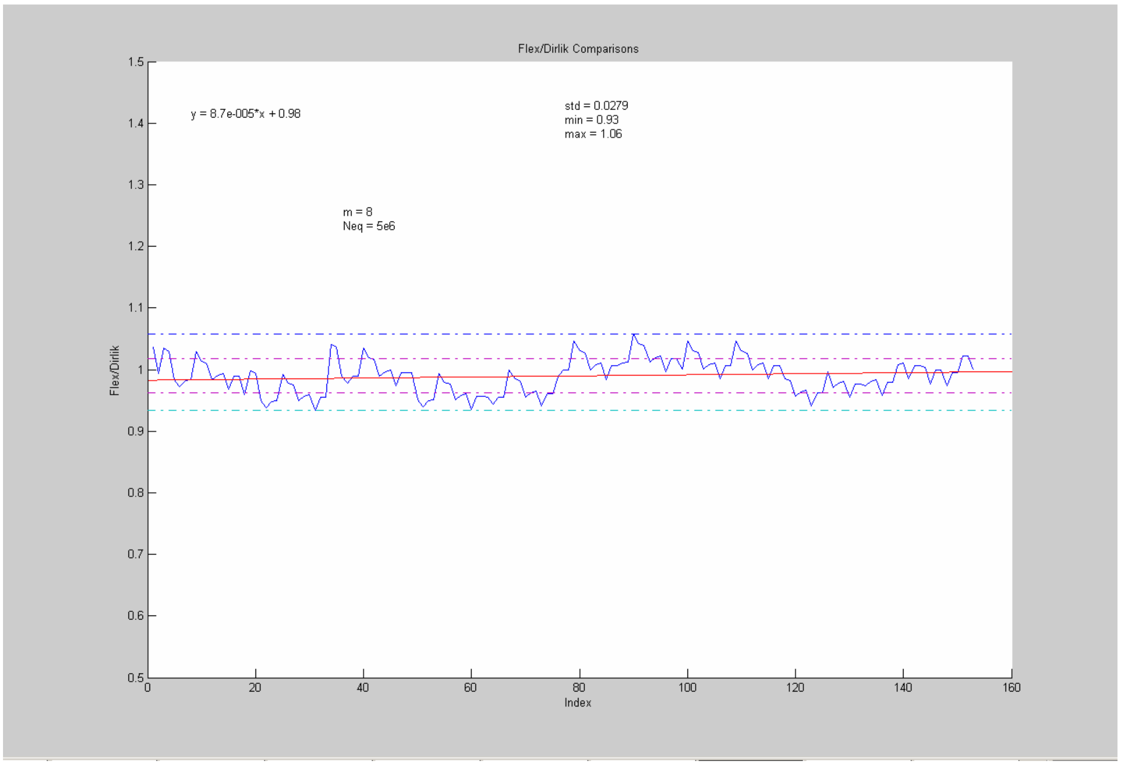
m = 3


Flex/Dirlik Comparisons
y = 0.00017*x + 0.98
std = 0.017
min = 0.96
max = 1.03
m = 4
Neq = 5e6
Flex/Dirlik
Index
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
120
140
160

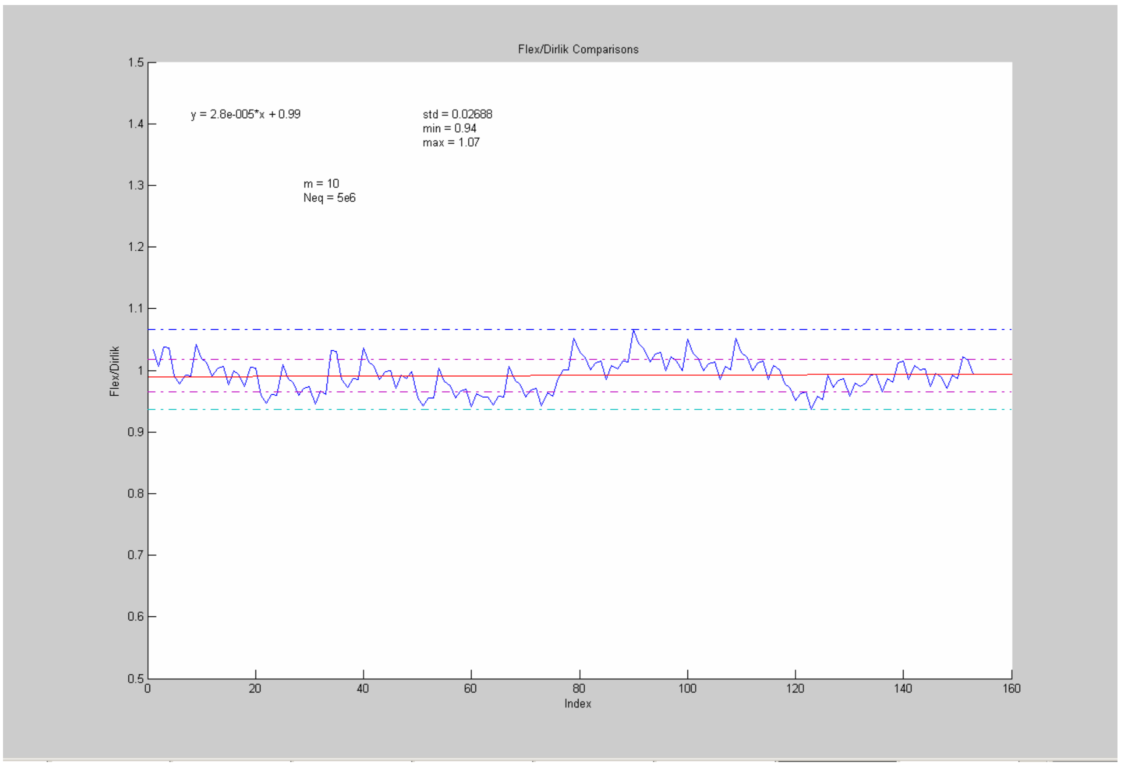
m = 4


Flex/Dirlik Comparisons
y = 0.00016*x + 0.98
std = 0.027
min = 0.93
max = 0.045
m = 6
Neq = 5e6
Flex/Dirlik
Index
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
120
140
160

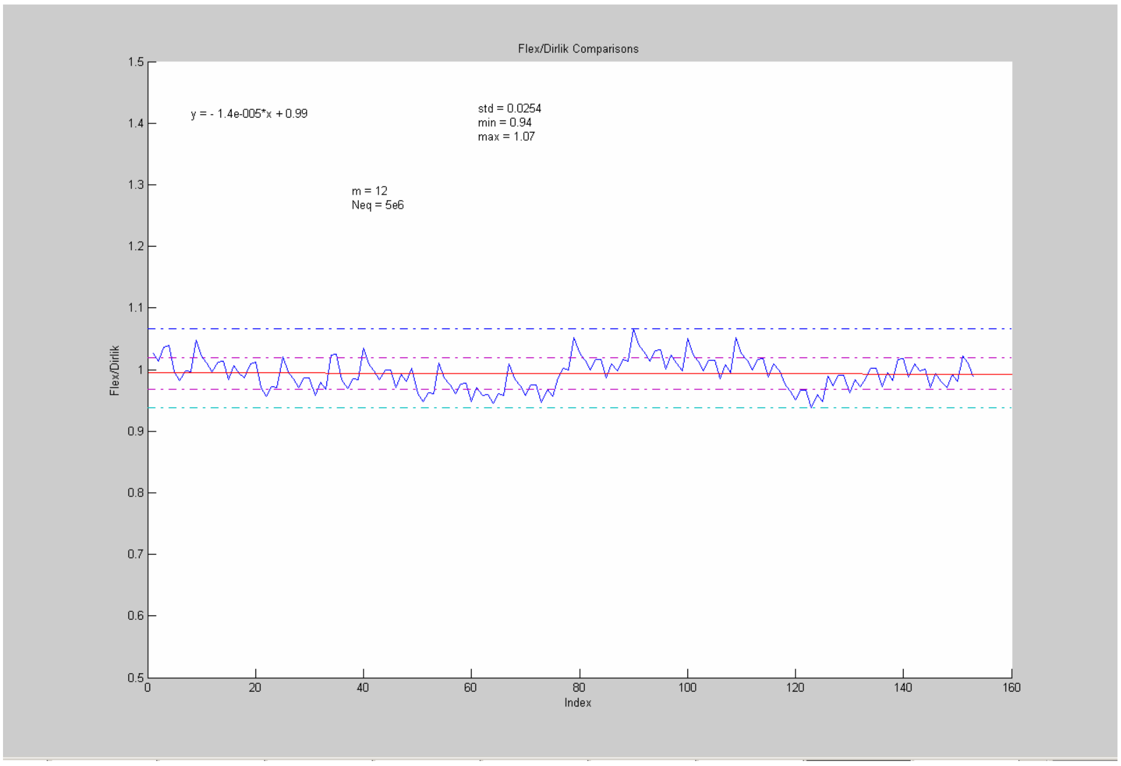
m = 6


Flex/Dirlik Comparisons
y = 8.7e-005*x + 0.98
std = 0.0279
min = 0.93
max = 1.06
m = 8
Neq = 5e6
Flex/Dirlik
Index

m = 8


Flex/Dirlik Comparisons
y = 2.8e-005*x + 0.99
std = 0.02688
min = 0.94
max = 1.07
m = 10
Neq = 5e6
Flex/Dirlik
Index

m = 10


Flex/Dirlik Comparisons
y = - 1.4e-005*x + 0.99
std = 0.0254
min = 0.94
max = 1.07
m = 12
Neq = 5e6
Flex/Dirlik
Index
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
120
140
160

m = 12