Project funded by the European Commission under the 6th (EC) RTD Framework Programme (2002- 2006) within the framework of the specific research and technological development programme “Integrating and strengthening the European Research Area”

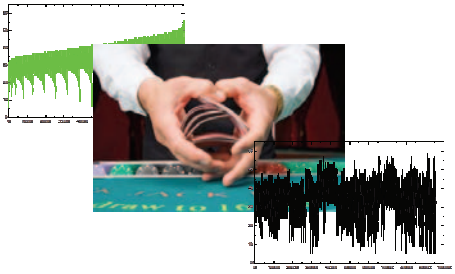
Project UpWind

Contract No.:
019945 (SES6)

“Integrated Wind Turbine Design”

Randomising New Wisper

- DEVELOPMENT AND IMPLEMENTATION OF A METHOD FOR RAINFLOW EQUIVALENT RANDOMISATION OF VARIABLE AMPLITUDE SEQUENCES -



AUTHORS:	Rogier Nijssen, T. Westphal, E. Stammes
AFFILIATION:	Knowledge Centre Wind turbine Materials and Constructions
ADDRESS:	Kluisgat 5, 1771 MV Wieringerwerf, the Netherlands
TEL.:	+31 (0) 227-504927/49
EMAIL:	r.p.l.nijssen@wmc.eu
FURTHER AUTHORS:	
REVIEWER:	
APPROVER:	

Document Information

DOCUMENT TYPE	report
DOCUMENT NAME:	WMC-2010-35
REVISION:	00
REV. DATE:	May 10, 2010
CLASSIFICATION:	R3: Restricted to WP members + PL
STATUS:	Draft

CONTENTS

LIST OF FIGURES

LIST OF TABLES

STATUS, CONFIDENTIALITY AND ACCESSIBILITY							
Status			Confidentiality			Accessibility	
S0	Approved/Released		**R0**	General public		Private web site	
S1	Reviewed		**R1**	Restricted to project members		Public web site	
S2	Pending for review		**R2**	Restricted to European. Commission		Paper copy	x
S3	Draft for commends	x	**R3**	Restricted to WP members + PL	x		

RECORD OF CHANGES			
Rev	**Date**	**Author**	**Description**
00	10.05.10	R.Nijssen	Initial version

1. BACKGROUND AND INTRODUCTION

Fatigue life description, modelling and prediction have been frequently supported by standardised load sequences that are not constant amplitude. The role of these variable amplitude (VA) load sequences is to provide a test standard to validate constant amplitude based fatigue models and experimental research. The background and necessity is described in e.g. [1]-[7].

Various standard variable amplitude test load sequences exist in fatigue research, of which WISPER and WISPERX are the most relevant examples in the wind turbine rotor blade industry and research. These particular load sequences are representative of the loads in a wind turbine blade, and were derived from strain measurements on blades in operation (blade root, flapwise direction). An important notion regarding the WISPER(X) sequences is, that the main intended use is not to represent a design load, but to compare materials in terms of VA fatigue response and/or to develop and validate fatigue models.

In the OPTIMAT blades project, a new WISPER-like sequence was developed [8] because it was the general opinion of the project partners, that the WISPER standard VA load sequences were no longer very representative of the load spectrum that a wind turbine blade experiences. This was because of the development in wind turbine size and control algorithms during the ~10 years after the definition of the original WISPER sequences. In other words, the cyclic content of the flapwise strain signal measured near the blade root was expected to have changed, due to changes in technology of various wind turbine components.

In addition, due to increasing turbine blade size, gravity loads had become more important and this influence on the cyclic load content in edgewise direction should be considered in a new standardised load sequence. However, in an early stage of the NEW WISPER development this was discarded because it was not clear how to implement this in a load signal for uni-axial testing.

Some doubt is justified as to whether a more 'modern' VA load sequence is more appropriate for the purposes mentioned above. Nevertheless, investigating the impact of technological development on blade loads is certainly worthwhile.

In [8], copying as much as possible the methodology used in the development of WISPER and WISPERX, a new load sequence was produced, and named 'NEW WISPER'. The most significant difference in the method, however, is that the last 'randomisation' step was not carried out. The two main reasons are, that the exact details of the randomisation were not entirely clear, and that at the time it was no longer feasible for project-organisational reasons.

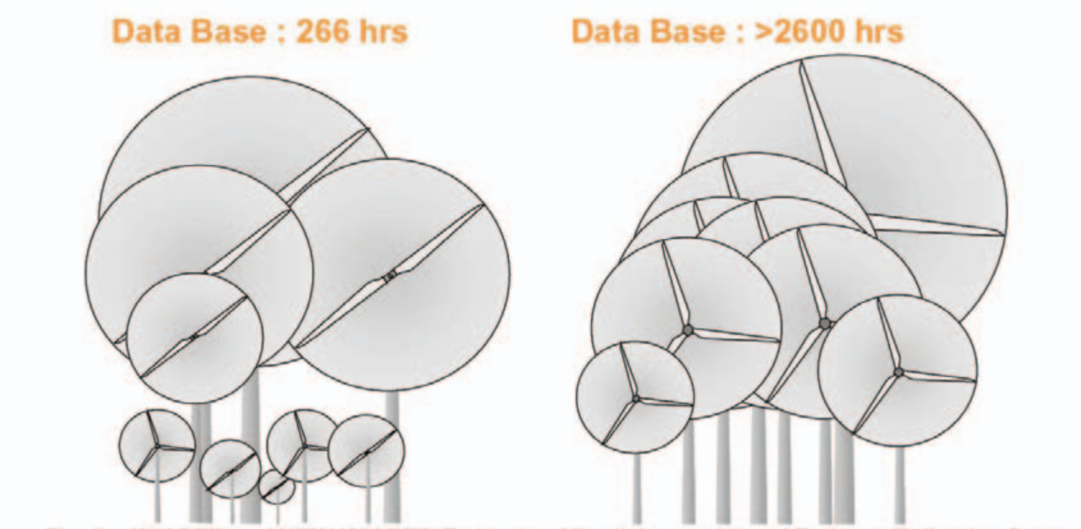


Fig. 2 : WISPER and NEW WISPER Turbines – Visual Comparison of Turbines Delivering Data

Figure 1: WISPER reference turbines, from [8]

The difference between the basis of NEW WISPER and its predecessors, and its development is described in Figure 1, and [8].

The sequences themselves are depicted in Figure 2 - Figure 4.

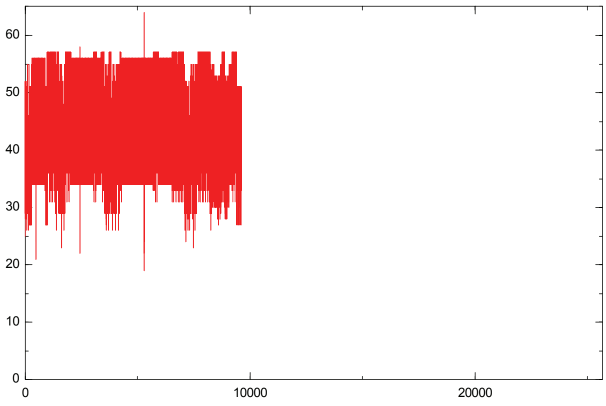


Figure 2: WISPERX load sequence

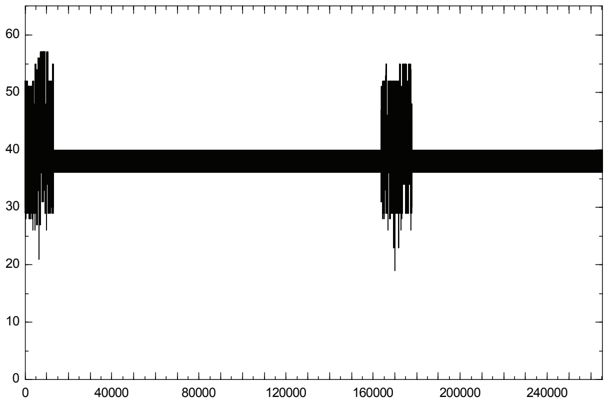


Figure 3: WISPER load sequence

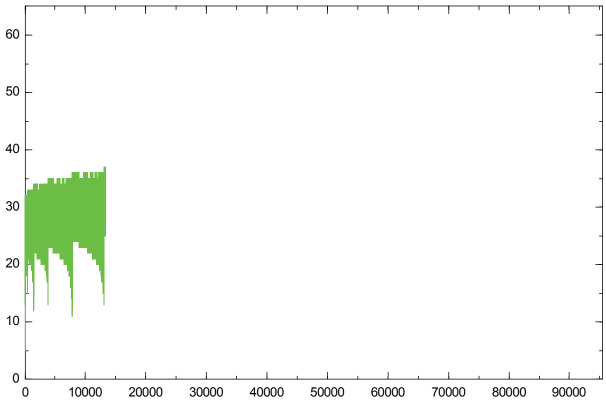


Figure 4: NEW WISPER load sequence

In various projects, fatigue testing has been carried out in the mean time, using all abovementioned load sequences including the NEW WISPER one. Based on these tests, some questions arose as to the influence of the NEW WISPER load sequence having a high degree of sequential ordering, essentially going from cycles with small ranges to large ranges and gradually increasing mean loads (where the WISPER and WISPERX load sequences are more random).

Furthermore the large transition in load which occurs between the end of the sequence and the start of a subsequent sequence might have important consequences for the damage, according to some theories, e.g. [9].

Within the UPWIND project, the randomisation of the NEW WISPER spectrum was therefore initiated.

The objective of this document is to describe the randomisation of the NEW WISPER load sequence.

Before the process of randomisation, extensive characterisation of the existing and initial load sequences was necessary, to ascertain that a resulting spectrum is 'more random', but all other characteristics are equivalent. In the following, special attention is paid to quantifying randomness of the load spectra.

2. CHARACTERISATION OF LOAD SEQUENCES

In this report, the load sequences under consideration are ‘dead’ sequences, meaning that they are inert sequences of numbers representing alternating peaks and valleys.

Various options exist for characterising a load sequence, some of which are:

- Number of peaks and valleys
- Values and locations within the sequence of the global extremes
- R-value information, where R = min valley/max peak
    - using the value of the global minimum valley and maximum peak
    - as an average of the R-values for each subsequent range
- Autocorrelation for a single lag parameter (for autocorrelation, see section 2.1)
- The length of the signal if it were laid out (sum of all ranges)
- Average range length
- Maximum and average ‘chunk’ length (subsequence of identical cycles)
- Results from a counting method
    - Rainflow counting
    - Level crossing
    - Range-mean

A range is defined here as the segment between peak and valley. The WISPER sequences are sequences of integers between 1 and 64, i.e. there are distinct levels. Level 25 represents zero load in WISPER(X), fo r NEW WISPER zero load is represented by level 22.

2.1 MEASURE OF RANDOMNESS: AUTOCORRELATION

Measuring randomness in a signal is typically done using the autocorrelation parameter. Autocorrelation for a signal (X_t) with known mean and variance is defined as [11] :

$$R(t,s) = \frac{E[(X_t - \mu)(X_s - \mu)]}{\sigma^2}$$

μ	=	Mean
σ^2	=	Variance
s, t	=	Lag parameter
E	=	Expected value operator

An autocorrelation value of 1 corresponds to a strongly correlated (not random) sequence and -1 indicates perfect negative correlation. A random sequence is characterised by an autocorrelation of 0.

In the case of sequences consisting of discrete peaks and valleys, and for a second-order stationary signal (where autocorrelation does not depend on location in the sequence), an estimate of the autocorrelation factor can be reduced to a single parameter function:

$$\hat{R}(k) = \frac{1}{(n-k)} \frac{\sum_{t=1}^{n-k} (X_t - \mu)(X_{t+k} - \mu)}{\sigma^2}$$

μ	=	Mean value of all points in sequence
σ^2	=	Variance of all points in sequence
k	=	Lag parameter

n = Number of points in sequence

This expression looks at each point and checks how it correlates with a point that is located *k* points further in the sequence, where *k* is the lag parameter.
The autocorrelation can be expressed in terms of different lag parameters *k*, which should always be specified. A value of *1* corresponds to autocorrelation of each subsequent point; a value of *2* means every 2nd subsequent point, etc.

Apart from that, the autocorrelation for a particular sequence can be calculated using all peaks and valleys, or on a reduced sequence with only peaks, or valleys, ranges or half cycle amplitudes, all giving different results.
Furthermore, perhaps contrary to intuition, autocorrelation for a sequence of only the peaks is not the same as the autocorrelation for a peak-and-valley-sequence with a lag parameter of 2, starting at a peak. The average and variance are different in these cases, giving different autocorrelation.

Note, that the plots in ANNEX B are for a selection of lag parameters. As a result, the autocorrelation factor for the complete sequence alternates between a negative and positive number for odd and even values of the lag parameter. This is, because for an odd value of the lag parameter, the correlation of a peak and a valley is negative.

Also note, that the autocorrelation is done for the sequence, without taking into account the zero mean stress level.
The only parameter in which the zero mean stress level is used, is the R-value.

It is clear from the characterisation, that indeed NEW WISPER is less random than its predecessors.

However, the degree of randomness depends on the parameter for which randomness is calculated. In all cases, the ranges give the lowest autocorrelation factor for small values of the lag parameter. Peak and valley autocorrelation are typically quite close to each other. Absolute values of randomness parameters vary strongly with lag parameter. Also, they vary considerably for different load spectra. When the lag parameter is expressed as a percentage of number of cycles, it seems that the autocorrelations converge to <0.2 for lags >10% of the number of records. So, the more cycles are skipped when looking at their relative correlation, the more random (autocorrelation→0) the sequence seems. This is explained as follows. The WISPER(X) sequences are made up of blocks of cycles which are random in length and in location. The longer the lag, the more cycles are picked from different loading blocks, the more random the sequence seems.

For the reshuffling of NEW WISPER, it seems appropriate to achieve a randomised spectrum which still consists of different constant amplitude blocks, instead of fully random cycles.

2.2 RAINFLOW COUNTING

Cyclic Rainflow counting is considered to be superior to non-cyclic counting and is used in this report. The method was taken from [12] and implemented in VBA code for Excel. The code is reproduced in ANNEX A.

2.3 DESCRIPTION OF THE CHARACTERISATION TOOL

Characterisation (and shuffling, see section 3) was implemented in an Excel workbook: “Rainflow counting and randomising load sequences.xls” [10]. A brief explanation is given in the first worksheet.

This workbook contains some macros, of which the VBA code is reproduced in ANNEX A and in ANNEX C.

ANNEX B gives some characteristics of the existing WISPER sequences, from this workbook.

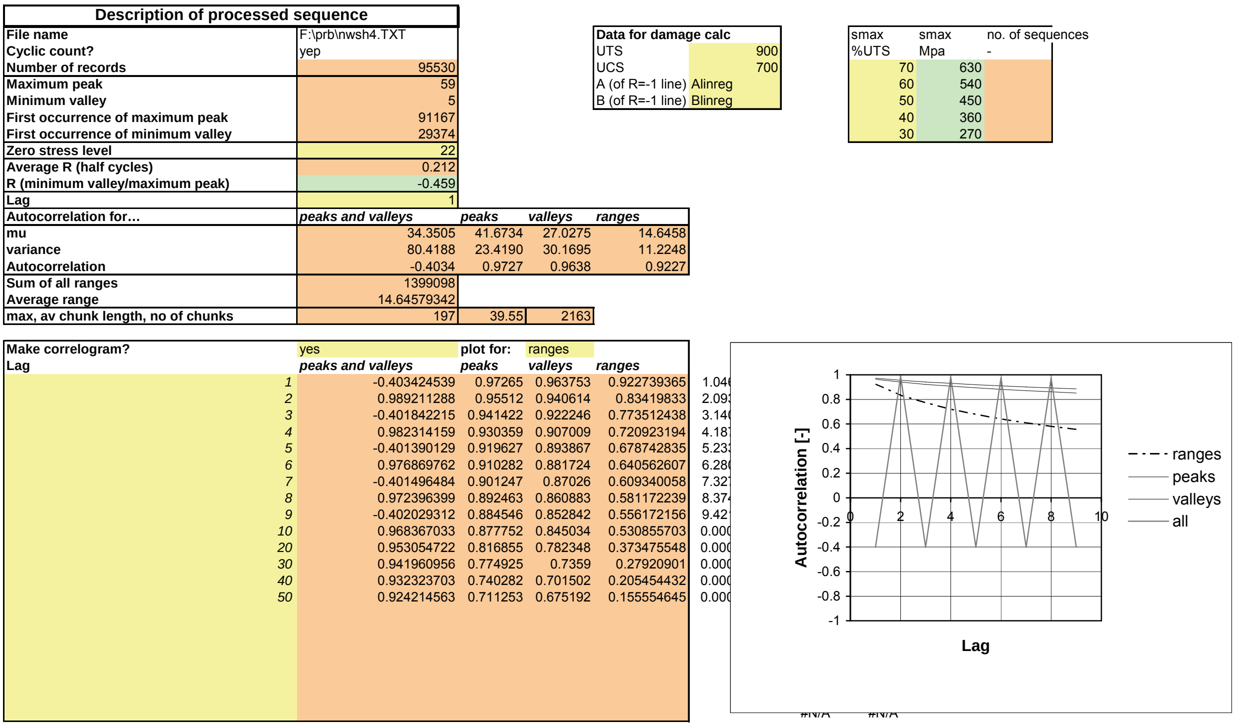
The sequence files are sequences of integers.

Furthermore, a table is given for autocorrelation parameters versus lag parameters and the resulting correlograms are shown.
Finally, an S-N curve with maximum load versus number of sequences is shown.

In the next worksheet ‘counting results’, see Figure 5, the results from a (cyclic) Rainflow count are displayed, including:

- Original sequence
- Rearranged sequence starting and ending with maximum load
- From-to matrix
- Min-Max-N matrix
- Min-Range-N matrix

(These matrices are generated by a macro and are overwritten each time it is run).



Description of processed sequence				
File name	F:\prb\nwsh4.TXT			
Cyclic count?	yep			
Number of records	95530			
Maximum peak	59			
Minimum valley	5			
First occurrence of maximum peak	91167			
First occurrence of minimum valley	29374			
Zero stress level	22			
Average R (half cycles)	0.212			
R (minimum valley/maximum peak)	-0.459			
Lag	1			
Autocorrelation for…	*peaks and valleys*	*peaks*	*valleys*	*ranges*
mu	34.3505	41.6734	27.0275	14.6458
variance	80.4188	23.4190	30.1695	11.2248
Autocorrelation	-0.4034	0.9727	0.9638	0.9227
Sum of all ranges	1399098			
Average range	14.64579342			
max, av chunk length, no of chunks	197	39.55	2163	

Data for damage calc	
UTS	900
UCS	700
A (of R=-1 line)	Alinreg
B (of R=-1 line)	Blinreg

smax	smax	no. of sequences
%UTS	Mpa	-
70	630	
60	540	
50	450	
40	360	
30	270	

Make correlogram?	yes	plot for:	ranges		
Lag	*peaks and valleys*	*peaks*	*valleys*	*ranges*	
1	-0.403424539	0.97265	0.963753	0.922739365	1.04
2	0.989211288	0.95512	0.940614	0.83419833	2.09
3	-0.401842215	0.941422	0.922246	0.773512438	3.14
4	0.982314159	0.930359	0.907009	0.720923194	4.18
5	-0.401390129	0.919627	0.893867	0.678742835	5.23
6	0.976869762	0.910282	0.881724	0.640562607	6.28
7	-0.401496484	0.901247	0.87026	0.609340058	7.32
8	0.972396399	0.892463	0.860883	0.581172239	8.37
9	-0.402029312	0.884546	0.852842	0.556172156	9.42
10	0.968367033	0.877752	0.845034	0.530855703	0.00
20	0.953054722	0.816855	0.782348	0.373475548	0.00
30	0.941960956	0.774925	0.7359	0.27920901	0.00
40	0.932323703	0.740282	0.701502	0.205454432	0.00
50	0.924214563	0.711253	0.675192	0.155554645	0.00

Figure 5: tables in worksheet ‘sequence description’, see also ANNEX B

3. SHUFFLING ALGORITHMS

Different algorithms can be conceived to randomise a load spectrum. When randomising a sequence, the objective is to change only the randomness, and not specifically any other properties, in other words, the new sequence should be 'equivalent'. For instance, the number of records, maximum and minimum, average R-value, Miner damage sum, etc, should be identical or at least very similar.

Formally, a clear distinction should be made between damage equivalence and cycle counting result equivalence. In a fatigue calculation employing Miner's sum, however, this distinction is irrelevant because order of the loads is assumed to be of no influence on the damage.

Furthermore, it is possible that there is a slight difference in some of the characteristics. For example, the cycle type and number of records may increase or decrease for a randomised spectrum because of transitional segments that can be introduced into a randomised sequence that are not present in the original sequence.

3.1 ALGORITHM 1 (SHUFFLING)

This algorithm is equivalent to taking a full deck of cards (peaks and valleys), and shuffling them.

- A set of constant amplitude cycles is selected from the sequence and extracted using Rainflow logic (the number of constant amplitude cycles is a random number between 1 and the maximum block length present in the sequence).
- All possible insertion points are located (insertion criteria meet Rainflow conditions).
- One of the insertion points is randomly chosen from the vector of potential insertion points.
- The constant amplitude cycles are inserted at the specified location.

3.2 ALGORITHM 2 (REBUILDING FROM RAINFLOW RESULTS)

This algorithm is equivalent to drawing random cards from an ordered set (e.g. all colours together or same numbers together). Cards are then inserted in a new stack based on Rainflow logic.

- The sequence is first Rainflow counted, the results are tabulated, e.g. in terms of minimum-range-N(umber of half cycles *minrangen*).
- A number of cycles *MaxNCand* is randomly selected (*MaxNCand* is between 1 and the maximum number of half cycles in the table *MaxN*).
- A selection is made of the rows of the table which have a number of half cycles which is equal to or larger than *MaxNCand*, this selection is stored in *candidaterows*.
- From this selection, a row is picked at random. Now we know what cycle type to insert (minimum *insertmin*, maximum *insertmax* and number of half cycles *insertN*).
- *InsertN* is limited by a value for 'chunk length'. This is to avoid self-ordering effects.
- A vector *inserthereorhere* is filled with record numbers which indicate possible insertion points. The last record of the new sequence is also included.
- A random point *inserthere* is chosen from this vector.
- If *inserthere* is equal to lastpoint, the set of cycles is inserted either within the sequence or at the end.
- The number of half cycles in the row associated with the inserted cycles is decreased by *insertN*.
- After completion of the final sequence, removal of 1 maximum peak or 1 minimum valley is required.

The advantage of algorithm 1 is, that no Rainflow counting is necessary on beforehand. On the other hand, every relocation of cycles involves a twofold shifting of sequence elements. This increases the administration effort and computation time.

Therefore, algorithm 2 was implemented in VBA for Excel.

3.3 SELF-ORDERING EFFECTS

Without taking into account all characteristics of the original sequence, both algorithms can have a tendency to self-ordering, resulting in a new sequence that is more ordered (higher autocorrelation) than the original sequence.
This is because blocks of cycles are drawn from a population which has a large number of cycles that are identical and a smaller number of cycles which are, typically, larger. When looking for insertion points for a particular batch, the probability is high that the insertion point is in a block of cycles identical to the cycles that are inserted.

There are several ways to avoid this. In this work, the number of cycles to be inserted (‘chunk length’) was limited. The advantage is, that by selecting an appropriate number of the maximum block length to be inserted, the new load sequence can be tailored to have similar properties in terms of average and maximum length of blocks with identical cycles.

3.4 EQUIVALENCE

The randomised spectrum should be equivalent to the original sequence in several ways. The sequence should ‘look’ the same, give similar counting results and similar damage.

3.4.1 Qualitative similarity

The procedure can be run a couple of times to obtain different results. Then the candidate giving the best overall score is selected. Some post-processing of the automatically generated result is required.

First of all, the sequence is inspected visually to check for obvious anomalies. Furthermore, the sequence should start and end with the zero stress level, i.e. level no. 22 (without violating Rainflow logic).

In addition, because the reshuffling is done from a Min-Range-N matrix from a cyclic Rainflow count, some additional maximum peaks and minimum valleys can occur (only one maximum peak and 1 minimum valley was present in the original sequence). Any excess peaks should be removed without violating Rainflow logic. In keeping with the WISPER(X) sequences, the peak(s) or valley(s) should be removed in such a manner that the peak and valley that remain are as far apart as possible.

These operations can be performed either manually or automatically.

3.4.2 Rainflow counting equivalence

Rainflow counting equivalence can be checked simply by comparing the Min-Range-N matrices of the original and new load sequence. There might be some differences caused by the ‘transitional’ cycles and by other modifications made during the post-processing stage.

A second check could be done in terms of damage equivalence, to increase confidence that the additional cycles do not introduce significant additional damage (or significantly less).

4. THE RANDOMISED NEW WISPER

The method described in this document was implemented and the result is shown in Figure 6.

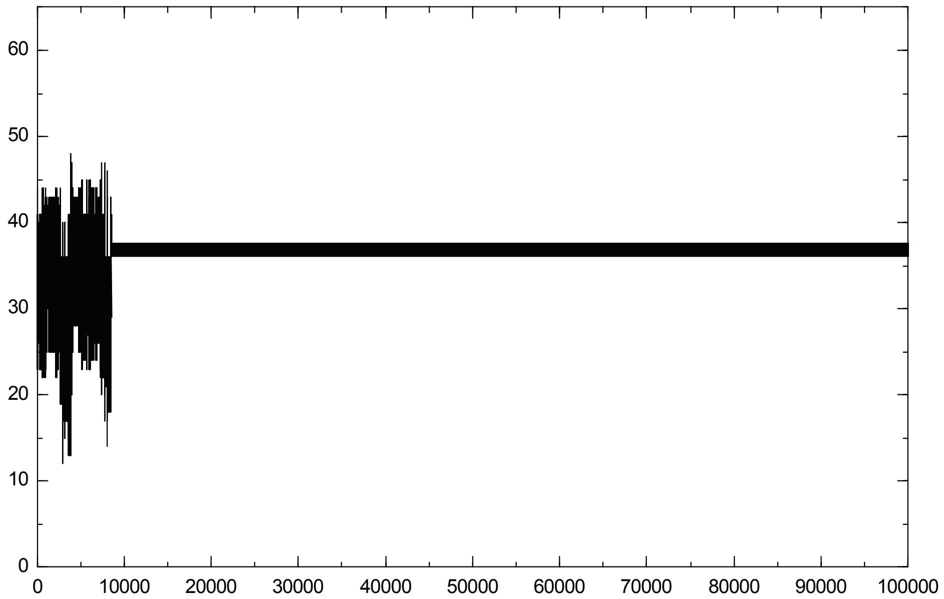


Figure 6: Randomised NEW WISPER

The properties of the new load sequence are shown in ANNEX D.

5. CONCLUDING REMARKS

Methods for shuffling of load sequences are described and a rebuilding algorithm was implemented. A randomised version of NEW WISPER was obtained with acceptable Rainflow count and damage equivalence.

This randomised version can be used instead of the original NEW WISPER load sequences in VA testing.

6. REFERENCES

[1] Germanischer Lloyd, ‘Requirements for the Determination of the Gradient of the S/N curve, Wac, Hamburg, 2001-01-08

[2] ISO, ‘Fibre-reinforced plastics – Determination of fatigue properties under cyclic loading conditions, ISO 13003, 2003-12-15

[3] ASTM, 'ASTM 3479/D3479M -96: Standard Test Method for Tension-Tension Fatigue of Polymer Matrix Composite Materials', 1996

[4] Sutherland, Herbert J., Mandell, John F., 'Optimized Goodman diagram for the analysis of fiberglass composites used in wind turbine blades', ASME/AIAA Wind Energy Symposium, paper AIAA-2005-0196, 2005

[5] Nijssen, R.P.L., Fatigue Life Prediction and Strength Degradation of Wind Turbine Rotor Blade Composites’, dissertation, Delft, 2006, ISBN 10:90-9021221-3

[6] ten Have, A.A., 'WISPER and WISPERX - Final definition of two standardised fatigue loading sequences for wind turbine blades', 1992

[7] ten Have, A.A., 'WISPER and WISPERX - A summary paper describing their background, derivation and statistics', 1992

[8] Bulder, B., Peeringa, J., et al., ‘NEW WISPER - Creating a new standard load sequence for modern wind turbine data’, OPTIMAT report, doc. no. 10278, April 2004

[9] Schaff, Jeffery R., Davidson, Barry D., 'Life Prediction Methodology for Composite Structures. Part II-Spectrum Fatigue', Journal of Composite Materials, Vol. 31, no. 2, 1997, pp. 158-181

[10] Rainflow counting and randomising load sequences, MS Excel file, on: UPWIND document site, 2010.

[11] www.wikipedia.org, accessed February 2008, search term ‘autocorrelation’.

[12] de Jonge, J.B., 'The analysis of load time histories by means of counting methods', Subchapter 3.4 of AGARDograph “Helicopter Fatigue Design Guide”, 1983

[13] Post, N., Reliability based design methodology incorporating residual strength prediction of structural fiber reinforced polymer composites under stochastic variable amplitude fatigue loading’, dissertation Virginia Polytechnic Institute and State University, March 18, 2008

ANNEX A Description of Rainflow counting code

To extract counting results from a series of peaks and valleys, code was programmed in VBA for Excel. Here, the code is given for reference.

The VBA code is reproduced here:

```
Sub RainflowCount(cyclic As Boolean, selectedrange As Range, selectedfile As String, resulttype As String,
usewhat As String)

'MAIN macro called by and using info acquired from userform usr_originalsequence

'performs a rainflow count on "originalsequence", which can be either a file or a worksheet range...
'...but has to be a column of longs!

'RN, February 2008

'initialize counts etc.

noofrecords = 0
usewhat2 = usewhat
cyclic2 = cyclic
resulttype2 = resulttype

Application.StatusBar = "Reading sequence"

If usewhat2 = "file" Then
   Call GetSequenceFromFile(selectedfile)
ElseIf usewhat2 = "range" Then
   Call GetSequenceFromRange(selectedrange)
End If

Application.StatusBar = "Analysing sequence"

Call DescribeSequence(originalsequence) 'calls countpeaksvalleys,
maxsandmins,make_rowncol_labels,autocorrelation,etc.

If cyclic2 Then
   Application.StatusBar = "Rearranging sequence"
   Call RearrangeSequence(originalsequence)
   Call CheckForNextNumberEqualToPrevious(rearrangedsequence)

   Application.StatusBar = "Copying sequence"
   Call Makecopyofsequence(rearrangedsequence)
Else
   Application.StatusBar = "Copying sequence"
   Call Makecopyofsequence(originalsequence)
End If

Application.StatusBar = "Range-pair count in progress"
Call RangePairCount(copyofsequence)

Application.StatusBar = "Range count in progress"
Call CheckForNextNumberEqualToPrevious(residualsequence)
Call RangeCount(residualsequence)

Application.StatusBar = "Converting to desired output type"
```

```
Call Convert_FromTo(fromtomatrix, resulttype)

Application.StatusBar = "Reporting"
Call DocumentResults(resultsmatrix, selectedfile, selectedrange)

Application.StatusBar = False

End Sub
Sub GetSequenceFromFile(whichfile As String)
'Fill an array "originalsequence" with a sequence of peaks and valleys
'RN, Feb 2008

a = 0

Open whichfile For Input As #1            'actually reads file
    Do While Not EOF(1)

        Line Input #1, pnt
        If IsNumeric(pnt) And Not IsEmpty(pnt) Then
            a = a + 1
            ReDim Preserve originalsequence(a)
            originalsequence(a) = CDbl(pnt)     'when reading a txt file, these strings need to be converted to
doubles (was longs until feb 29th, 2008 (RN))
        End If

    Loop

Close #1

End Sub
Sub GetSequenceFromRange(whichrange As Range)
'Fill an array "originalsequence" with a sequence of peaks and valleys
'RN, Feb 2008

For a = 1 To whichrange.Rows.Count
    ReDim Preserve originalsequence(a)
    originalsequence(a) = whichrange.Cells(a).Value
Next

End Sub
Sub ClearSequenceDescription()
'clears sheet "Sequence description" of previous results
'RN, March 2008

Sheets("Sequence description").Select
Range("B2:B8").Select
Selection.ClearContents
Range("B10:B11").Select
Selection.ClearContents
Range("B14:B18").Select
Selection.ClearContents
Range("C14:E16").Select
Selection.ClearContents
Range("B23").Select
Selection.ClearContents

lagtablerow = 1
```

```
Do
   For a = 1 To 4
      Range("lagtableheader").Offset(lagtablerow, a).Select
      Selection.ClearContents
   Next 'a
   lagtablerow = 1 + lagtablerow
Loop Until Range("lagtableheader").Offset(lagtablerow, 1) = ""

End Sub
Sub DescribeSequence(sequence As Variant)
'called by RainflowCount
'yields general information on the sequence; name, number of peaks, valleys, (location) of maxima/minima
'autocorrelation, etc.

'RN, March 2008

Call ClearSequenceDescription

Call CheckForNextNumberEqualToPrevious(sequence)
Call CountPeaksValleys(sequence)
Call MaxsAndMins(sequence)
Call MakeFromtoLabels(sequence)
Call SumOfAllRanges(sequence)
Call RValue(sequence)

k = Range("autocor_lag").Value

Call AutoCorrelation(sequence, k)
muall = mu
varianceall = variance
autocorall = autocor

Call PeaksValleysRanges(sequence)

   Call AutoCorrelation(sequenceofpeaks, k)   'do not calculate mu and variance in separate sub! autocorrelation should be calc'd using mu and variance of sequence at hand...
   mupeak = mu
   variancepeak = variance
   autocorpeak = autocor

   Call AutoCorrelation(sequenceofvalleys, k)
   muvalley = mu
   variancevalley = variance
   autocorvalley = autocor

   Call AutoCorrelation(sequenceofranges, k)
   murange = mu
   variancerange = variance
   autocorrange = autocor

make_correlogram = (Range("make_correlogram") = "yes")

If make_correlogram Then
   Call CreateCorrelogramTable(sequence, sequenceofpeaks, sequenceofvalleys, sequenceofranges)
End If

End Sub
```

```
Sub CreateCorrelogramTable(sequence, sequenceofpeaks, sequenceofvalleys, sequenceofranges)
'fills table with autocorrelations for user-defined lags
'RN, March 2008

lagtablerow = 1

Do

k = Range("lagtableheader").Offset(lagtablerow, 0)

Call AutoCorrelation(sequence, k)
Range("lagtableheader").Offset(lagtablerow, 1) = autocor

Call AutoCorrelation(sequenceofpeaks, k)
Range("lagtableheader").Offset(lagtablerow, 2) = autocor

Call AutoCorrelation(sequenceofvalleys, k)
Range("lagtableheader").Offset(lagtablerow, 3) = autocor

Call AutoCorrelation(sequenceofranges, k)
Range("lagtableheader").Offset(lagtablerow, 4) = autocor

lagtablerow = lagtablerow + 1
Loop Until Range("lagtableheader").Offset(lagtablerow, 0) = ""

End Sub
Sub PeaksValleysRanges(sequence As Variant)
'extracts peaks, valleys and ranges as separate vectors from a sequence
'RN, March 2008

b = 0: c = 0: d = 0  'indices of sequence of peaks, -valleys, and -ranges, respectively

For a = 1 To UBound(sequence) - 1
   If sequence(a) > sequence(a + 1) Then
      b = b + 1
      ReDim Preserve sequenceofpeaks(b)
      sequenceofpeaks(b) = sequence(a)
   ElseIf sequence(a) < sequence(a + 1) Then
      c = c + 1
      ReDim Preserve sequenceofvalleys(c)
      sequenceofvalleys(c) = sequence(a)
   End If
   d = d + 1
   ReDim Preserve sequenceofranges(d)
   sequenceofranges(d) = Abs(sequence(a + 1) - sequence(a))
Next  'a

End Sub
Sub CheckForNextNumberEqualToPrevious(sequence As Variant)
'if a sequence contains two identical subsequent numbers, this can confuse matters
'this sub generates a warning if this is the case
'RN, March 2008

For a = 1 To UBound(sequence) - 1
   If sequence(a) = sequence(a + 1) Then
      MsgBox ("double number, at element " & a)
   End If
Next
```

```
End Sub
Sub MakeFromtoLabels(sequence As Variant)
'Called by describeSequence
'Makes an unsorted vector containing all peak/valley values that exist in sequence...
'...and sorts it, resulting in fromtolabels

'RN February 2008

b = 0
ReDim unsortedfromtolabels(1)

For a = 1 To UBound(sequence)        'make labels vector
    If Not ElementExistsInVector(sequence(a), unsortedfromtolabels) Then
        b = b + 1
        ReDim Preserve unsortedfromtolabels(b)
        unsortedfromtolabels(b) = sequence(a)

    End If
Next 'a

'now we have a vector of unique but unsorted peak and valley values (unsortedfromtolabels)
'sort unsortedfromtolabels to create fromtolabels:...

For a = 1 To UBound(unsortedfromtolabels)
    For b = 1 To UBound(unsortedfromtolabels)
        If (FindRankOfElementInVector(unsortedfromtolabels(b), unsortedfromtolabels)) = a Then
            ReDim Preserve fromtolabels(a)
            fromtolabels(a) = unsortedfromtolabels(b)
        End If
     Next 'b
Next 'a


End Sub
Function FindRankOfElementInVector(element As Variant, vector As Variant)
'Called by MakeFromtoLabels
'gives rank of element in vector (duh)
'RN, March 2008

FindRankOfElementInVector = 0

For c = 1 To UBound(vector)
    If element >= vector(c) Then
        FindRankOfElementInVector = FindRankOfElementInVector + 1
    End If
Next 'c

End Function
Function ElementExistsInVector(element As Variant, vector As Variant)
'Called by MakeFromtoLabels and others
'ElementExistsInVector toggles to true if element is found in vector

'RN February 2008

ElementExistsInVector = False

For vectorelement = 1 To UBound(vector)
```

```
    If element = vector(vectorelement) Then
    ElementExistsInVector = True
    End If
Next 'c

End Function
Sub CountPeaksValleys(sequence As Variant)
'Called by describesequence
'Gives number of records
'Result stored in noofrecords

'RN February 2008

noofrecords = UBound(sequence)

End Sub
Sub MaxsAndMins(sequence As Variant)
'Called by describesequence
'Finds the first occurrence of the maximum and minimum and its recordnumber
'results are stored in:
'maxpeak (maximum of peaks); maxpeakrec (associated record)
'minvalley (minimum of valleys); minvalleyrec (associated record)

'RN February 2008

a = 0
maxpeak = -2147483647#
minvalley = 2147483647#

For a = 1 To UBound(sequence)
    If maxpeak < sequence(a) Then
        maxpeak = sequence(a)
        maxpeakrec = a
    End If

    If minvalley > sequence(a) Then
        minvalley = sequence(a)
        minvalleyrec = a
    End If
Next

End Sub
Sub AutoCorrelation(sequence As Variant, lag As Long)
'Called by describesequence

'Rdakje(k)=1/((n-k)sigma^2)*sumt=1 to n-k of:(Xt-mu)(Xt+k-mu)
'where mu=mean
'sigma=variance
'n=number of records
'k=lag
'Source: Wikipedia.org, 2006, http://en.wikipedia.org/wiki/Autocorrelation

'RN March 2008

mu = 0
variance = 0
autocor = 0
```

```
'calculate autocorrelation using each kth point in the sequence
autocor_lag = lag 'give lag here

For a = 1 To UBound(sequence)
    mu = mu + sequence(a)
Next 'a
mu = mu / UBound(sequence)

For a = 1 To UBound(sequence)
    variance = variance + (sequence(a) - mu) ^ 2
Next 'a
variance = variance / UBound(sequence)

For a = 1 To UBound(sequence) - autocor_lag
    autocor = autocor + (sequence(a) - mu) * (sequence(a + autocor_lag) - mu)
Next 'a

autocor = autocor / ((UBound(sequence) - autocor_lag) * variance)

End Sub
Sub RValue(sequence As Variant)
'Calculates R-value from sequence using average R-value of all ranges
'RN March 2008

averageRvalue = 0
zerostresslevel = Range("zerostresslevel").Value
remark2 = ""
b = 0

For a = 1 To UBound(sequence) - 1
    If Application.max(sequence(a) - zerostresslevel, sequence(a + 1) - zerostresslevel) = 0 Then 'exclude infinite R-values from average
    b = b + 1
    Else
        averageRvalue = Application.min(sequence(a) - zerostresslevel, sequence(a + 1) - zerostresslevel) _
        / Application.max(sequence(a) - zerostresslevel, sequence(a + 1) - zerostresslevel) + averageRvalue
    End If
Next 'a
averageRvalue = averageRvalue / (UBound(sequence))

If b > 0 Then
    remark2 = " (excluding " & b & " occurences of R=infinite from average R)"
End If

End Sub
Sub SumOfAllRanges(sequence As Variant)
'Sums ranges and calculates average range

'RN March 2008

sumofranges = 0

For a = 1 To UBound(sequence) - 1
    sumofranges = Abs(sequence(a + 1) - sequence(a)) + sumofranges
Next 'a
averagerange = sumofranges / (UBound(sequence) - 1)
```

```
End Sub
Sub RearrangeSequence(sequence As Variant)
'Rearranges sequence to start and end with absolute maximum

'RN Feb 2008

Application.StatusBar = "Rearranging sequence"

ReDim intermedsequence(noofrecords + 1)

a = 0

For b = maxpeakrec To noofrecords          'go from peak record to last record
   a = a + 1
   intermedsequence(a) = sequence(b)
Next

If intermedsequence(a) = sequence(1) Then    'if original sequence starts and ends with same number, skip
this number once
   startat = 2
Else
   startat = 1
End If

For b = startat To maxpeakrec             'go from first record to and including peak record
   a = a + 1
   intermedsequence(a) = sequence(b)
Next

a = 0
b = 0

Do                                  'avoid any subsequent intermediate points
   a = a + 1

   If a > UBound(intermedsequence) Then
      Exit Do
   End If

   b = b + 1
   ReDim Preserve rearrangedsequence(b)
   rearrangedsequence(b) = intermedsequence(a)

   If a < UBound(intermedsequence) - 2 Then
      If intermedsequence(a + 1) <= intermedsequence(a) And intermedsequence(a + 2) <=
intermedsequence(a + 1) Then
         a = a + 2
      End If
   End If

Loop

End Sub
Sub RangePairCount(sequence As Variant)

'Rainflow count using de Jonge's algorithm
'RN, March 2008
```

```
'---REFERENCE:
'   Annex 3.4 B from:
'   J.B. de Jonge, 'The analysis of load-time histories by means of counting methods',
'   NLR report MP 82039 U, National Aerospace Laboratory (NLR),
'   Amsterdam, the Netherlands,August 13th, 1982
'--------------

ReDim fromtomatrix(UBound(fromtolabels), UBound(fromtolabels)) As Variant     'clear from-to matrix for storing rainflow cycles
extractcycle = False

a = 0   'p=1
q = 0   'q=1
f = 0   'f=0

starthier:

a = a + 1
lastpercent = Format(q / UBound(sequence) * 100, "0")
q = q + 1

thispercent = Format(q / UBound(sequence) * 100, "0")

If lastpercent <> thispercent Then 'only update statusbar at whole percents
   Application.StatusBar = "Range-pair count in progress " & Format(thispercent, "0") & "%"
End If

ReDim Preserve readsequence(a)
readsequence(a) = sequence(q)

If q = UBound(sequence) Then
   f = 1
End If

checkp:
If a >= 4 Then
   If readsequence(a - 2) > readsequence(a - 3) And readsequence(a - 1) >= readsequence(a - 3) And readsequence(a) >= readsequence(a - 2) Then
      extractcycle = True
   ElseIf readsequence(a - 2) < readsequence(a - 3) And readsequence(a - 1) <= readsequence(a - 3) And readsequence(a) <= readsequence(a - 2) Then
      extractcycle = True
   Else
      If Not f = 0 Then
        GoTo makeresidual
      Else
        GoTo starthier
      End If
   End If
Else
   If Not f = 0 Then
      GoTo makeresidual
   Else
      GoTo starthier
   End If
End If
```

```
If extractcycle = True Then
    rowloc = Application.Match(readsequence(a - 2), fromtolabels)
    colloc = Application.Match(readsequence(a - 1), fromtolabels)

    fromtomatrix(rowloc, colloc) = fromtomatrix(rowloc, colloc) + 1 ' Like this, it only works for positive numbers in sequence.
    fromtomatrix(colloc, rowloc) = fromtomatrix(colloc, rowloc) + 1

    readsequence(a - 2) = readsequence(a)
    a = a - 2   'p=p-2
End If

extractcycle = False
GoTo checkp

'make residual sequence for range counting
makeresidual:
lasta = a
ReDim residualsequence(lasta)
For a = 1 To lasta
    residualsequence(a) = readsequence(a)
Next

End Sub
Sub RangeCount(sequence As Variant)
'Called by RainflowCount
'Can also be called by describesequence
'Creates from-to matrix with ranges...
'...or adds rangecount data to existing from-to-matrix

'RN, feb 2008

If UBound(sequence) - 1 > 1 Then            'to prevent that empty=subscript out of range
    For b = 1 To UBound(sequence) - 1

        If IsNumeric(sequence(b)) Then
            For c = 1 To UBound(fromtolabels)
                If sequence(b) = fromtolabels(c) Then
                    rowloc = c
                    Exit For
                End If
            Next 'c

            For c = 1 To UBound(fromtolabels)
                If sequence(b + 1) = fromtolabels(c) Then
                    colloc = c
                    Exit For
                End If
            Next 'c
        If rowloc <> colloc Then
            fromtomatrix(rowloc, colloc) = fromtomatrix(rowloc, colloc) + 1
        End If

        End If

    Next ' b

End If
End Sub
Sub Convert_FromTo(fromtomatrix As Variant, convertto As String)
'Called by RainflowCount
'Creates from from-to matrix either of the following, depending on the argument:
'From-to-matrix
'min-max-N
'min-range-N
'other variations can be implemented or done without VBA...

'RN, February 2008

Select Case convertto

    Case "fromto"           '--------------------------------------

        resultsmatrixrows = UBound(fromtomatrix, 1)
        resultsmatrixcols = UBound(fromtomatrix, 2)

        ReDim resultsmatrix(resultsmatrixrows, resultsmatrixcols)

        For a = 1 To resultsmatrixrows
            For b = 1 To resultsmatrixcols
                resultsmatrix(a, b) = fromtomatrix(a, b)
            Next
        Next

    Case "betweenminmaxN"   '--------------------------------

        Call cnvrt_betweenminmaxN

    Case "minrangeN"        '-------------------------------------

        Call cnvrt_betweenminmaxN

        ReDim minrangeNmatrix(resultsmatrixcols, resultsmatrixrows)
        For a = 1 To resultsmatrixrows
            minrangeNmatrix(1, a) = resultsmatrix(1, a)
            minrangeNmatrix(2, a) = resultsmatrix(2, a) - resultsmatrix(1, a)
            minrangeNmatrix(3, a) = resultsmatrix(3, a)
        Next 'a
        For a = 1 To resultsmatrixrows
            resultsmatrix(1, a) = minrangeNmatrix(1, a)
            resultsmatrix(2, a) = minrangeNmatrix(2, a)
            resultsmatrix(3, a) = minrangeNmatrix(3, a)
        Next 'a
End Select

End Sub
Sub cnvrt_betweenminmaxN()
'converts fromtomatrix into 3-column matrix, with minimum, maximum and N as columns

resultsmatrixrows = UBound(fromtomatrix, 1)    'rows are "from"
resultsmatrixcols = 3                          'columns are "to"

c = 0                                          'line in resultsmatrix

For b = 1 To resultsmatrixrows
```

```
    For a = 1 To UBound(fromtomatrix, 2)
        If fromtomatrix(a, b) > 0 Or fromtomatrix(b, a) > 0 Then
            If fromtolabels(a) <= fromtolabels(b) Then
                c = c + 1
                ReDim Preserve resultsmatrix(resultsmatrixcols, c)
                resultsmatrix(1, c) = fromtolabels(a)
                resultsmatrix(2, c) = fromtolabels(b)
                resultsmatrix(3, c) = fromtomatrix(a, b) + fromtomatrix(b, a)
            End If
        End If
    Next 'b

Next 'a

resultsmatrixrows = c

'copy to betweenminmaxN before further processing

For a = 1 To resultsmatrixrows
    ReDim Preserve betweenminmaxN(3, resultsmatrixrows)
    betweenminmaxN(1, a) = resultsmatrix(1, a)
    betweenminmaxN(2, a) = resultsmatrix(2, a)
    betweenminmaxN(3, a) = resultsmatrix(3, a)
Next 'a

End Sub
Sub DocumentResults(resultsmatrix, afile, arange)
'writes results to file and/or to range in worksheet
'RN, March 2008

Application.Calculation = xlCalculationManual
Sheets("Sequence description").Activate

If usewhat2 = "file" Then
    Range("sequencefile") = afile
ElseIf usewhat2 = "range" Then
    Range("sequencefile") = arange
End If

If cyclic2 Then
    Range("cycliccount") = "yep"
Else
    Range("cycliccount") = "nope"
End If

Range("noofrecords") = noofrecords
Range("averageRvalue") = averageRvalue & remark2

Range("mu") = muall
Range("variance") = varianceall
Range("autocor") = autocorall

Range("mupeak") = mupeak
Range("variancepeak") = variancepeak
Range("autocorpeak") = autocorpeak

Range("muvalley") = muvalley
Range("variancevalley") = variancevalley
Range("autocorvalley") = autocorvalley

Range("murange") = murange
Range("variancerange") = variancerange
Range("autocorrange") = autocorrange

Range("maxpeak") = maxpeak
Range("minvalley") = minvalley
Range("maxpeakrec") = maxpeakrec
Range("minvalleyrec") = minvalleyrec
Range("sumofranges") = sumofranges
Range("averagerange") = averagerange

'RN March 2008
Sheets("Counting results").Activate

With Sheets("Counting results")
    .Cells.Select
    Selection.ClearContents
    .Range("A1").Select
End With

remark1 = ""
noofrecordsplotted = noofrecords

If noofrecords > 65000 Then
    resultsmatrixrows = 65000
    noofrecordsplotted = 65000
    remark1 = " (only first 65000 lines plotted)"
End If

With Sheets("Counting results").Range("a1")
    'First print original sequence
    .Value = "Original sequence" & remark1
    ReDim Preserve originalsequence(noofrecordsplotted) 'otherwise you get a type mismatch error (exact reason unclear)
    .Offset(1, 0).Resize(noofrecordsplotted, 1) = Application.Transpose(originalsequence)

    'Then print rearranged sequence
    If cyclic2 Then
        .Offset(0, 1).Value = "Rearranged sequence" & remark1
        ReDim Preserve rearrangedsequence(noofrecordsplotted)
        .Offset(1, 1).Resize(noofrecordsplotted) = Application.Transpose(rearrangedsequence)
    End If

    '...then print residual sequence
    .Offset(0, 2).Value = "Residual sequence"
    .Offset(1, 2).Resize(lasta) = Application.Transpose(residualsequence)

    '...followed by from-to-matrix

    .Offset(0, 4) = "From-to-matrix"
    .Offset(1, 4) = "To"
    .Offset(0, 5) = "From"
    .Offset(1, 5).Resize(1, UBound(fromtolabels)) = fromtolabels
    .Offset(2, 4).Resize(UBound(fromtolabels)) = Application.Transpose(fromtolabels)
    .Offset(2, 5).Resize(UBound(fromtomatrix, 1), UBound(fromtomatrix, 2)) = Application.Transpose(fromtomatrix)
    'print between minmax (keep 1 column empty between fromto and minmax)
    .Offset(0, 6 + UBound(fromtolabels)) = "Min"
    .Offset(0, 7 + UBound(fromtolabels)) = "Max"
    .Offset(0, 8 + UBound(fromtolabels)) = "N"
    .Offset(1, 6 + UBound(fromtolabels)).Resize(UBound(betweenminmaxN, 2), 3) =
Application.Transpose(betweenminmaxN)

    'print minrange (keep 1 column empty between minrange and minmax)
    .Offset(0, 10 + UBound(fromtolabels)) = "Min"
    .Offset(0, 11 + UBound(fromtolabels)) = "Range"
    .Offset(0, 12 + UBound(fromtolabels)) = "N"
    .Offset(1, 10 + UBound(fromtolabels)).Resize(UBound(minrangeNmatrix, 2), 3) =
Application.Transpose(minrangeNmatrix)

    '...etc.

End With

Application.Calculation = xlCalculationAutomatic

End Sub
Sub Makecopyofsequence(sequence As Variant)
'called by RainflowCount
'makes a copy of the original sequence which can then be canibalised in the rainflow count
'results stored in copyoforoginalsequence

ReDim copyofsequence(UBound(sequence))

For a = 1 To UBound(sequence)
    copyofsequence(a) = sequence(a)
Next

End Sub
```

Intentionally left blank

ANNEX B CHARACTERISATION OF (new) wisper(x)

Table B - 1: Summary of WISPER characteristics

File name	wisper.dat			
Number of records	265423			
Maximum peak	64			
Minimum valley	1			
First occurrence of maximum peak	34482			
First occurrence of minimum valley	123303			
Zero stress level	25			
Average R (half cycles)	0.394			
R (minimum valley/maximum peak)	-0.615			
Lag	1			
Autocorrelation for…	***peaks and valleys***	***peaks***	***valleys***	***ranges***
mu	40.9132	47.7175	34.1090	13.6085
variance	56.3827	11.3774	8.7904	5.0943
Autocorrelation	-0.6875	0.8435	0.8973	0.7366
Sum of all ranges	3612006			
Average range	13.60854			

NB: characteristics expressed in levels

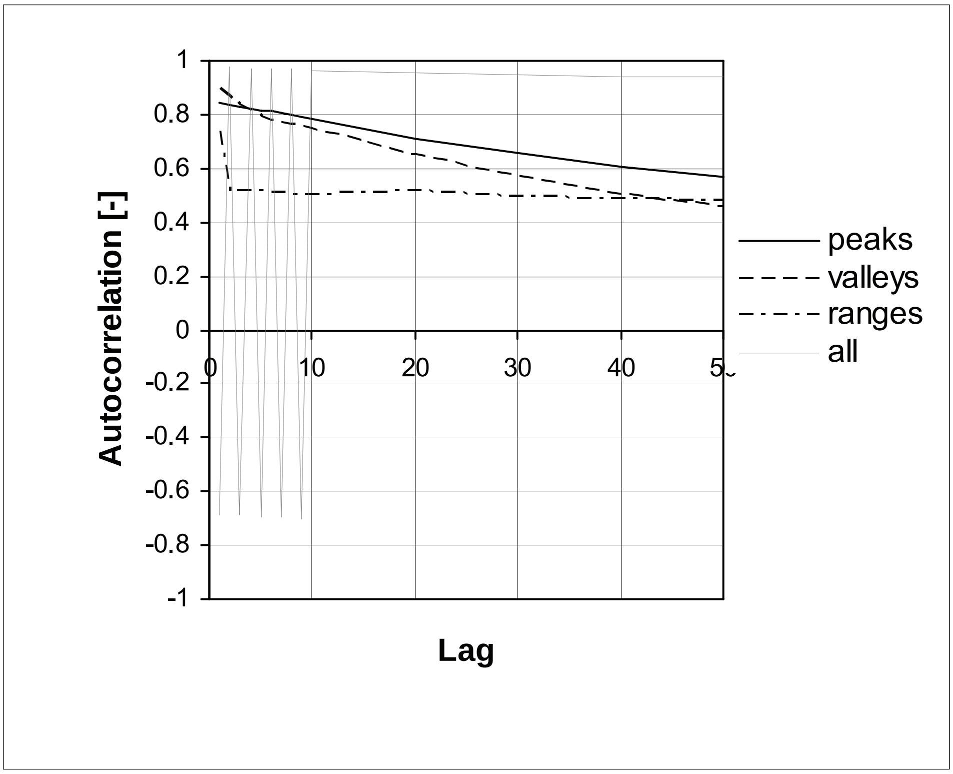


Figure B - 1: Autocorrelation plot for lag values 1-9, and 10, 20, 30, 40, and 50 (WISPER)

Table B - 2: Summary of WISPERX characteristics

File name	wisperx.dat			
Number of records	25663			
Maximum peak	64			
Minimum valley	1			
First occurrence of maximum peak	5298			
First occurrence of minimum valley	13481			
Zero stress level	25			
Average R (half cycles)	0.248			
R (minimum valley/maximum peak)	-0.615			
Lag	1			
Autocorrelation for…	***peaks and valleys***	***peaks***	***valleys***	***ranges***
mu	41.1810	50.6872	31.6762	19.0110
variance	102.7702	14.3925	10.4267	4.2200
Autocorrelation	-0.7790	0.8638	0.8543	0.5874
Sum of all ranges	487860			
Average range	19.01099			

NB: characteristics expressed in levels

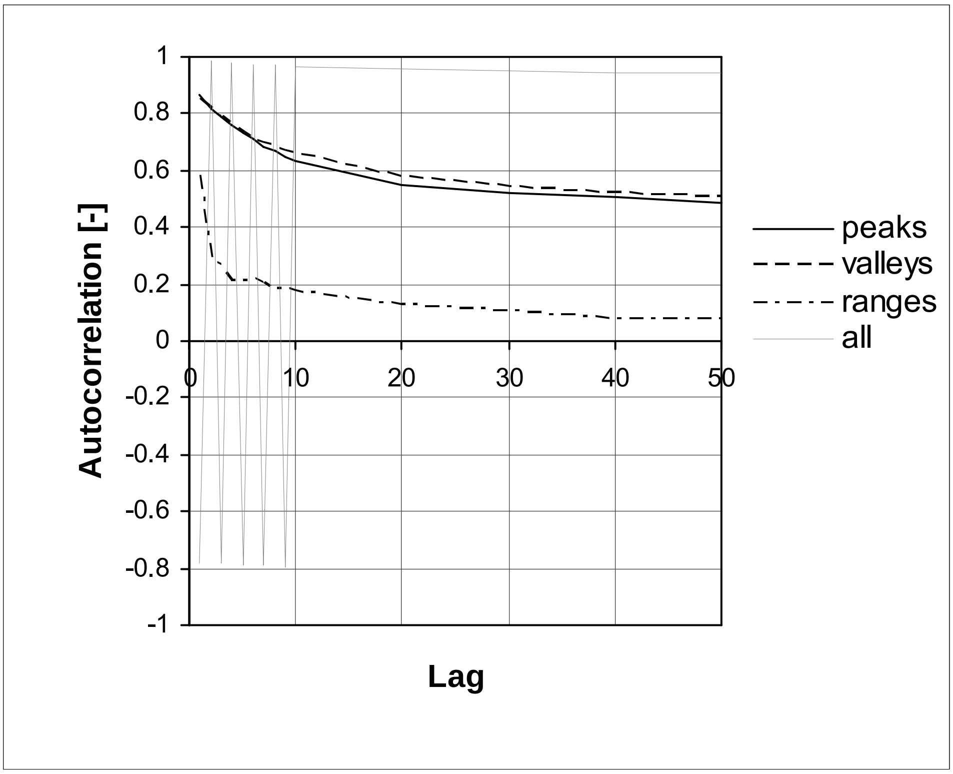


Figure B - 2: Autocorrelation plot for lag values 1-9, and 10, 20, 30, 40, and 50 (WISPERX)

Table B - 3: Summary of NEW WISPER characteristics

File name	NEW WISPER 64.TXT			
Number of records	95472			
Maximum peak	59			
Minimum valley	5			
First occurrence of maximum peak	95459			
First occurrence of minimum valley	2			
Zero stress level	22			
Average R (half cycles)	0.213			
R (minimum valley/maximum peak)	-0.459			
Lag	1			
Autocorrelation for…	***peaks and valleys***	***peaks***	***valleys***	***ranges***
mu	34.3515	41.6683	27.0341	14.6342
variance	80.2813	23.4211	30.0522	11.5710
Autocorrelation	-0.4060	0.9987	0.9954	0.9924
Sum of all ranges	1397142			
Average range	14.6342			

NB: characteristics expressed in levels

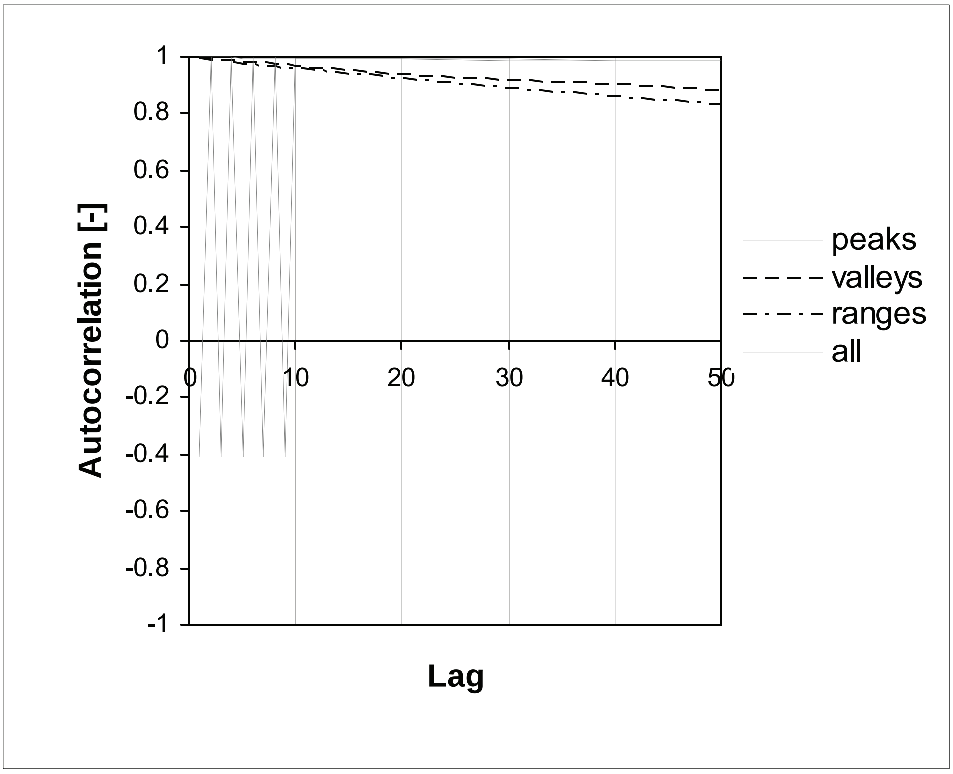


Figure B - 3: Autocorrelation plot for lag values 1-9, and 10, 20, 30, 40, and 50 (NEW WISPER)

Intentionally left blank

ANNEX C Reshuffling algorithm

```
Option Explicit
Option Base 1

'-----PARAMETER DECLARATION (ROUGHLY IN ORDER OF APPEARANCE)-------------------------------
'------------------------------------------------------------------------------------------

'*****debugging******************
Dim dummy As Integer, dummy2 As Integer             '(for debugging)
Dim IsNumber2InSequence As Boolean                  '(for debugging)
Dim IsNumber1InSequence  As Boolean                 '(for debugging)
Dim cyclesadded As Boolean                          '(for debugging)
'*******************************

Dim a As Long, b As Long                            'counters
Dim d As Long                                       'counters
Dim e As Long                                       'counters

Dim minrangen() As Variant                          '3-column matrix with columns minimum, range, and
number of half cycles
Public newsequence() As Variant                     'sequence reconstructed from minrangen (this
declaration allows for continuous monitoring during debugging)
Dim lastpoint As Long                               'last point (that is not empty) in newsequence
Dim lastpoint1 As Long                              'returned by findlastpoint (for debugging)
Dim totalhalfcycles As Long                         'number of half cycles in minrangen

Dim MaxN As Long                                    'maximum number of cycles in matrix minrangeN
Dim AverageN As Long                                'average number of cycles in matrix minrangeN
Dim MaxNCand As Long                                'random (even) number of halfcycles which will be
inserted/added in/to newsequence

Dim candidaterows() As Variant                      'selection of rows from minrangeN with at least
MaxNCand cycles
Dim candidaterow As Long                            'random row from candidaterows

Dim insertmin As Double, insertmax As Double        'minimum and maximum of halfcycle(s) to
insert in sequence (from candidaterow)
Dim insertN As Long                                 '(even) number of halfcycles to insert (from
candidaterow)
Dim insertfromrow As Long                           'row in minrangeN matrix from which cycles are taken
(from candidaterow)

Dim insertthereorhere() As Variant                  'vector of possible insertion points
Dim canbeinserted As Boolean                        'used as criterion for accepting possible insertion
points into insertthereorhere
Dim inserthere As Long                              'random selection of insertion point from
insertthereorhere

Dim insertpointismax As Boolean                     'TRUE if point after which cycle to be inserted is
maximum
Dim checkit As Boolean                              'checks file if this is TRUE

Sub rfeqshuffle()
```

```
usr_sequencetoshuffle.Show      'show userform which will initiate shuffling

'-----WRITE THE NEW SEQUENCE TO .txt-FILE------------------------------------------------------
'-----------------------------------------------------------------------------------------
Dim whichfile As String

whichfile = Application.GetSaveAsFilename

Open whichfile For Output As #1
   a = 1
   Do
      If IsNumeric(newsequence(a)) Then
         Write #1, newsequence(a)
         a = a + 1
      End If
   Loop Until IsEmpty(newsequence(a)) Or a = UBound(newsequence) + 1
Close #1
'-----------------------------------------------------------------------------------------

'-----WRITE THE NEW SEQUENCE AND THE MINRANGEN MATRIX TO SHEET
'-----------------------------------------------------------------------------------------
With Sheets("Counting results").Range("bl1")
   ReDim Preserve newsequence(UBound(newsequence))  'otherwise you get a type mismatch error (exact reason unclear)
   If UBound(newsequence, 1) < 60000 Then
      .Offset(1, 0).Resize(UBound(newsequence), 1) = Application.Transpose(newsequence)
   End If
   .Offset(1, 2).Resize(UBound(minrangen, 2), 3) = Application.Transpose(minrangen)
End With
'-----------------------------------------------------------------------------------------

checkit = True

If checkit Then
Call CheckThisSequence(newsequence)
End If


Application.StatusBar = False   'Statusbar reset to 'Ready'

End Sub
Sub CreateSequenceFromMinRangeN(selectedrange As Variant)

'Called by usr_sequencetoshuffle userform

'Creates a WISPER-type sequence from a 3 column min-range-N matrix
'(note, that it is assumed that the number of halfcycles is even)
'The matrix is input via selection of the min-range-N matrix from a worksheet using a userform
'The new sequence will consist of CA-blocks of varying lengths, and is constructed in the following manner:

'---   A matrix minrangeN is read
'---
'---
'---
'---
'---
```

```
'---
'---
'---

'RN March 2008

dummy2 = 0                              'for debugging purposes

'-----PUT SELECTED RANGE FROM SHEET INTO DYNAMIC ARRAY------------------------------------
'----------------------------------------------------------------------------------------
For a = 1 To selectedrange.Rows.Count
   ReDim Preserve minrangen(3, a)
   minrangen(1, a) = selectedrange.Cells(a, 1).Value
   minrangen(2, a) = selectedrange.Cells(a, 2).Value
   minrangen(3, a) = selectedrange.Cells(a, 3).Value
Next 'a
'----------------------------------------------------------------------------------------

'-----COUNT NUMBER OF CYCLES IN MINRANGEN------------------------------------------------
'----------------------------------------------------------------------------------------
Call CountCycles(minrangen)
'----------------------------------------------------------------------------------------

'-----MAKE SURE THERE IS ENOUGH ROOM IN THE NEWSEQUENCE-----------------------------------
'----------------------------------------------------------------------------------------
ReDim newsequence(2 * totalhalfcycles)
lastpoint = 1                           'Set lastpoint to 1
'----------------------------------------------------------------------------------------

'-----MAIN LOOP---------------------------------------------------------------------------
'----------------------------------------------------------------------------------------

Do

   Call FindMaxN(minrangen)
   Call FindAverageN(minrangen)

choosenumber:

   '--CHOOSE CYCLES TO INSERT------------------------------------------------------------------
   '----------------------------------------------------------------------------------------
      '-----Determine number of cycles to insert--------------------------------------------
      '----------------------------------------------------------------------------------------
   Randomize

   If AverageN >= 2 Then
      MaxNCand = Application.WorksheetFunction.Even(Int((AverageN - 1) * Rnd + 1))
   Else
      MaxNCand   =   Application.WorksheetFunction.Even(Int((MaxN   -   1)   *   Rnd   +   1))
'from F1, upperbound is MaxN, lowerbound=1
   End If

      '-----Create 3-column matrix CandidateRows, containing all rows in MinRangeNmatrix with at least
MaxNCand halfcycles
      '----------------------------------------------------------------------------------------
   b = 0
   ReDim candidaterows(4, 1)
   For a = 1 To UBound(minrangen, 2)
      If minrangen(3, a) >= MaxNCand Then
         b = b + 1                                                 'rows in candidaterows
         ReDim Preserve candidaterows(4, b)
         candidaterows(1, b) = minrangen(1, a): candidaterows(2, b) = minrangen(2, a)
         candidaterows(3, b) = minrangen(3, a)
         candidaterows(4, b) = a                                          '(we need to remember
where to subtract insertN from later on!)
      End If
   Next 'a

   If b = 0 Then GoTo choosenumber                                        'if CandidateRows is
empty, go back
      '------------------------------------------------------------------------------------

      '-----From the available cycle types in candidate rows, pick a random row for insertion-
      '-------------------------------------------------------------------------------------
   Call PickCyclesForInsertion(candidaterows)                             'choose a random row
from candidaterows
   '-----------------------------------------------------------------------------------------

   'now we have a min, a range, and a number of cycles ready for insertion in newsequence, all we have to do
is find a spot

   '-----CHOOSE LOCATION TO PUT CYCLES---------------------------------------------------------
   Call FindInsertionPoints(newsequence, insertmin, insertmax)

   Randomize
   inserthere = Int((UBound(inserthereorhere)) * Rnd + 1)     'choose another random number to find
row inserthere (random number is between 1 and ubound(inserthereorhere))
   inserthere = inserthereorhere(inserthere)
   '-----------------------------------------------------------------------------------------

   '******debugcode************************
   'If insertmin = 1 And insertmax = 64 Then
   'dummy2 = 1
   'End If
   '   Call FindLastPoint(newsequence)
   'If lastpoint1 <> lastpoint Then
   'dummy = 1
   'End If
   '****************************************

   '-----INSERT SELECTED CYCLES AT CHOSEN POINT IN NEWSEQUENCE--------------------------------
   '----------------------------------------------------------------------------------------
   If inserthere = lastpoint Then
      Call InsertCyclesAtEnd(newsequence, insertmin, insertmax, insertN, inserthere)
   Else
      Call InsertCyclesHere(newsequence, insertmin, insertmax, insertN, inserthere)
   End If
   '----------------------------------------------------------------------------------------

   '-----SUBTRACT INSERTED CYCLES FROM MinRangeNmatrix----------------------------------------
   '----------------------------------------------------------------------------------------
   minrangen(3, insertfromrow) = minrangen(3, insertfromrow) - insertN
   Call FindMaxN(minrangen)

   Application.StatusBar = "progress: " & Format(lastpoint / totalhalfcycles * 100, "0") & "%"

   '   '******debugcode*************************
   '   If MaxN < 10 Then
   '      dummy = 1
   '   End If
   '   '****************************************
   '   '******debugcode*************************
   '   If minrangen(3, insertfromrow) < 0 Then
   '      dummy = 1
   '   End If
   '   '****************************************

Loop Until MaxN <= 0
'-------------------------------------------------------------------------------------------

End Sub
Sub CountCycles(sequence As Variant)

'Called by CreateSequenceFromMinRangeN
'counts number of halfcycles by summing elements in third column of minrangen

totalhalfcycles = 0

For a = 1 To UBound(sequence, 2)
   totalhalfcycles = totalhalfcycles + sequence(3, a)
Next ' a

End Sub

Sub PickCyclesForInsertion(candidaterows As Variant)

'Called by CreateSequenceFromMinRangeN
'From the matrix candidaterows, which is a selection of rows from minrangeN with at least MaxNCand half
cycles...
'...a single row is randomly chosen
'this gives the minimum, maximum of the cycles for insertion
'the number of cycles in this row is used as an upper bound for the actual number of cycles to insert insertN
'insertN is then randomly chosen

Randomize

candidaterow = Int((UBound(candidaterows, 2)) * Rnd + 1) 'pick row insertthis from CandidateRows

insertmin = candidaterows(1, candidaterow): insertmax = candidaterows(1, candidaterow) +
candidaterows(2, candidaterow)
insertfromrow = candidaterows(4, candidaterow)

insertN = Application.WorksheetFunction.Even(Int(candidaterows(3, candidaterow) * Rnd + 1))

''-----attempt to prevent ordering effect by limiting seriously number of cycles inserted....(can be deleted)
Dim maxchunklength As Long
maxchunklength = 100
If insertN > maxchunklength Then
```

```
    insertN = Application.WorksheetFunction.Even(Int(maxchunklength * Rnd) + 1)
End If
'-----end of attempt


End Sub
Sub FindMaxN(sequence As Variant)

'Called by CreateSequenceFromMinRangeN
'Returns the maximum number of half cycles still present in minrangeN
'and stores it in parameter MaxN

MaxN = 0

For a = 1 To UBound(sequence, 2) '3 columns, lots of rows
    If MaxN < sequence(3, a) Then
        MaxN = sequence(3, a)
    End If
Next 'a

End Sub
Sub FindAverageN(sequence As Variant)

'Called by CreateSequenceFromMinRangeN
'Returns average number of half cycles from the 3rd column in the minrangeN-matrix'
'and stores it in AverageN

AverageN = 0

For a = 1 To UBound(sequence, 2)
    AverageN = AverageN + sequence(3, a)
Next 'a

AverageN = AverageN / (a - 1)

End Sub
Sub FindInsertionPoints(sequence, min, max)

'create a vector of possible insertion points insertthereorhere
'a possible insertion point is the first record in newsequence which satisfies rainflow logic...
        '...i.e. the record contains a point smaller than or equal to the min,...
        '...AND the next record contains a point larger than or equal to the max.

ReDim insertthereorhere(1)

b = 0
For a = 1 To lastpoint - 1

    '-----Determine if cycle can be inserted...--------------------------------------------
    '-------------------------------------------------------------------------------------
    canbeinserted = False

    If IsEmpty(sequence(a)) And IsEmpty(sequence(a + 1)) Then    'if both the current point and the next one are empty, this is the first run and entire sequence is empty
        Exit For
    End If
    If min <> Application.WorksheetFunction.min(sequence(a), sequence(a + 1)) Then
        If max <> Application.WorksheetFunction.max(sequence(a), sequence(a + 1)) Then

            If min >= sequence(a) And min < sequence(a + 1) Then
                If max > sequence(a) And max <= sequence(a + 1) Then
                    canbeinserted = True
                End If
            End If

            If min >= sequence(a + 1) And min < sequence(a) Then
                If max > sequence(a + 1) And max <= sequence(a) Then
                    canbeinserted = True
                End If
            End If

        End If
    End If
    '-------------------------------------------------------------------------------------

    '-----...If it can, add point in sequence to vector of possible insertion points-------
    '-------------------------------------------------------------------------------------
    If canbeinserted Then
        b = b + 1
        ReDim Preserve insherethereorhere(b)
        inserthereorhere(b) = a
        canbeinserted = False
    End If
    '-------------------------------------------------------------------------------------
Next 'a

'-----add the last point of the sequence as another possible insertion point---------------
b = b + 1
ReDim Preserve inserthereorhere(b)
inserthereorhere(b) = lastpoint
'-----------------------------------------------------------------------------------------

End Sub
Sub InsertCyclesAtEnd(sequence As Variant, insertmin As Double, insertmax As Double, insertcycles As Long, inserthere As Long)

'Called by main loop of CreateSequenceFromMinRangeN
'Inserts selected cycles at end of newsequence


'******debugcode************************
'If sequence(inserthere) = 64 Or sequence(inserthere) = 1 Then
'    dummy = 1
'End If
'****************************************

'******debugcode************************
'Call FindLastPoint(sequence)
'If lastpoint1 <> lastpoint Then
'dummy = 1
'End If
'*****************************************
```

```
cyclesadded = False

b = 0
If IsEmpty(sequence(inserthere)) Then     'for the first time (you only get into this macro if inserthere is lastpoint, and lastpoint is only empty if whole sequence is empty
    Do
        sequence(inserthere + b) = insertmax
        sequence(inserthere + b + 1) = insertmin
        b = b + 2
    Loop Until b + 1 >= insertcycles
    sequence(inserthere + b) = insertmax  'to get even number of cycles
    lastpoint = lastpoint + insertcycles

    '******debugcode************************
    '   Call FindLastPoint(sequence)
    '   If lastpoint1 <> lastpoint Then
    '   dummy = 1
    '   End If
    '*****************************************

    cyclesadded = True
    Exit Sub
End If

If sequence(inserthere) > sequence(inserthere - 1) Then
    insertpointismax = True
Else
    insertpointismax = False
End If

'either add the cycles to insert after or starting from the last point

If insertpointismax Then
    If insertmin >= sequence(inserthere) Then
        Do
            sequence(inserthere + b) = insertmax
            sequence(inserthere + b + 1) = insertmin
            b = b + 2
        Loop Until b + 1 >= insertcycles
        sequence(inserthere + b) = insertmax  'to get even number of cycles
        lastpoint = lastpoint + insertcycles

        '******debugcode************************
        'Call FindLastPoint(sequence)
        'If lastpoint1 <> lastpoint Then
        'dummy = 1
        'End If
        '*****************************************

        cyclesadded = True
        Exit Sub
    Else
        Do
            sequence(inserthere + b + 1) = insertmin
            sequence(inserthere + b + 2) = insertmax
            b = b + 2
        Loop Until b + 1 >= insertcycles
```

```
      sequence(inserthere + b + 1) = insertmin 'to get even number of cycles
      lastpoint = lastpoint + insertcycles + 1

      '******debugcode************************
      'Call FindLastPoint(sequence)
      'If lastpoint1 <> lastpoint Then
      'dummy = 1
      'End If

      '*****************************************
      cyclesadded = True
      Exit Sub
   End If
Else
   If insertmax <= sequence(inserthere) Then
      Do
         sequence(inserthere + b) = insertmin
         sequence(inserthere + b + 1) = insertmax
         b = b + 2
      Loop Until b + 1 >= insertcycles
      sequence(inserthere + b) = insertmin 'to get even number of cycles
      lastpoint = lastpoint + insertcycles

      '******debugcode************************
      'Call FindLastPoint(sequence)
      'If lastpoint1 <> lastpoint Then
      'dummy = 1
      'End If
      '*****************************************

      cyclesadded = True
      Exit Sub
   Else
      Do
         sequence(inserthere + b + 1) = insertmax
         sequence(inserthere + b + 2) = insertmin
         b = b + 2
      Loop Until b + 1 >= insertcycles
      sequence(inserthere + b + 1) = insertmax 'to get even number of cycles
      lastpoint = lastpoint + insertcycles + 1

      '******debugcode************************
      'Call FindLastPoint(sequence)
      'If lastpoint1 <> lastpoint Then
      'dummy = 1
      'End If
      '*****************************************

      cyclesadded = True
      Exit Sub
   End If
End If

''******debugcode************************
'If Not cyclesadded Then
'    dummy = 1
'End If
```

```
''******************************************

End Sub
Sub FindLastPoint(sequence As Variant)

'Called by debugcode
'Finds last point in sequence that is not empty, and stores element number in lastpoint1
'(this can be compared to lastpoint in debugcode)

lastpoint1 = 0
For d = 1 To UBound(sequence)
   If Not IsEmpty(sequence(d)) Then
      lastpoint1 = lastpoint1 + 1
   End If
Next 'a


End Sub
Sub InsertCyclesHere(sequence As Variant, insertmin As Double, insertmax As Double, insertcycles As Long, inserthere As Long)

'******debugcode************************
'Call FindLastPoint(sequence)
'If lastpoint1 <> lastpoint Then
'dummy = 1
'End If
'******************************************
'******debugcode************************
'If dummy2 = 1 Then
'   If Not AreNumbersInSequence(sequence, 1, 64) Then
'      dummy = 1
'   End If
'End If
'******************************************
'******debugcode************************
'If sequence(inserthere) = 64 Or sequence(inserthere) = 1 Then
'    dummy = 1
'End If
'******************************************
'shift cycles from inserthere+1 to lasta+1

 For a = 0 To (lastpoint - inserthere - 1) '
'******debugcode************************
'     If sequence(lastpoint + insertcycles + 1 - a) = 1 Or sequence(lastpoint + insertcycles + 1 - a) = 64 Then
'     dummy = 1
'     End If
'******************************************

    sequence(lastpoint + insertcycles - a) = sequence(lastpoint - a)

 Next 'a

'******debugcode************************
'If dummy2 = 1 Then
'   If Not AreNumbersInSequence(sequence, 1, 64) Then
'      dummy = 1
'   End If
'End If
'*****************************************

 'overwrite cycles that were already shifted with mins and maxes to insert
 If sequence(inserthere) > sequence(inserthere + 1) Then

    b = 0
    Do
       sequence(inserthere + b + 1) = insertmin
       sequence(inserthere + b + 2) = insertmax
       b = b + 2
    Loop Until b >= insertcycles

 ElseIf sequence(inserthere) < sequence(inserthere + 1) Then

    b = 0
    Do
       sequence(inserthere + b + 1) = insertmax
       sequence(inserthere + b + 2) = insertmin
       b = b + 2
    Loop Until b >= insertcycles
 End If
 lastpoint = lastpoint + insertcycles

'******debugcode************************
'If dummy2 = 1 Then
'   If Not AreNumbersInSequence(sequence, 1, 64) Then
'      dummy = 1
'   End If
'****************************************
'******debugcode************************
'Call FindLastPoint(sequence)
'If lastpoint1 <> lastpoint Then
'dummy = 1
'End If
'****************************************

End Sub
Function AreNumbersInSequence(sequence As Variant, number1 As Variant, number2 As Variant) As Boolean

'Called by debugcode
'Returns true if two numbers exist in the sequence

AreNumbersInSequence = False
IsNumber1InSequence = False
IsNumber2InSequence = False

For e = 1 To UBound(sequence)
   If sequence(e) = number1 Then
      IsNumber1InSequence = True
   End If
   If sequence(e) = number2 Then
      IsNumber2InSequence = True
   End If
    If IsNumber1InSequence And IsNumber2InSequence Then
```

```
        AreNumbersInSequence = True
        Exit Function
    End If
Next 'e
End Function

Sub CheckWhatSequence()


'-----READ A FILE INTO A VECTOR OF DOUBLES------------------------------------------------
'---------------------------------------------------------------------------------------
Dim whichfile As String
Dim asequence() As Variant
Dim a As Long
Dim pnt As Variant

whichfile = Application.GetOpenFilename

Open whichfile For Input As #1              'actually reads file
    a = 0
    Do While Not EOF(1)

        Line Input #1, pnt
        If IsNumeric(pnt) And Not IsEmpty(pnt) Then
            a = a + 1
            ReDim Preserve asequence(a)
            asequence(a) = CDbl(pnt)       'when reading a txt file, these strings need to be converted to doubles (was longs until feb 29th, 2008 (RN))
        End If

    Loop

Close #1

Call CheckThisSequence(asequence)


End Sub
Sub CheckThisSequence(sequence As Variant)

'called by...
'performs all kinds of checks on the sequence (add extra subroutines as required)
Dim noproblems As Boolean
Dim checkedfor As String
Dim dummy As Variant                    'for displaying msgbox including title
checkedfor = "Checked for: " & vbCr


noproblems = True

'-----SUBSEQUENT POINTS ARE NOT EQUAL----------------------------------------------------
'---------------------------------------------------------------------------------------
For a = 1 To UBound(sequence) - 1
    If Not IsEmpty(sequence(a)) Then
        If sequence(a) = sequence(a + 1) Then
            MsgBox ("equal elements detected at " & a & " and " & a + 1)
            noproblems = False
        End If
    End If
Next 'a
checkedfor = checkedfor & "-subsequent points are not equal" & vbCr

'-----------------------------------------------------------------------------------------

'-----SUBSEQUENT POINTS ARE REALLY ALTERNATING PEAKS AND VALLEYS---------------------------
'-----------------------------------------------------------------------------------------
For a = 1 To UBound(sequence) - 2
    If Not IsEmpty(sequence(a)) And Not IsEmpty(sequence(a + 1)) And Not IsEmpty(sequence(a
+ 2)) Then
        If sequence(a) < sequence(a + 1) And sequence(a + 2) > sequence(a + 1) Then
            MsgBox ("intermediate point detected at element" & a)
            noproblems = False
        End If
        If sequence(a) > sequence(a + 1) And sequence(a + 2) < sequence(a + 1) Then
            MsgBox ("intermediate point detected at element" & a)
            noproblems = False
        End If
    End If
Next 'a
checkedfor = checkedfor & "-subsequent alternating peaks and valleys" & vbCr

'-----------------------------------------------------------------------------------------

If noproblems Then
    dummy = MsgBox(checkedfor & vbCrLf & "No problems identified", 0, "Sequence Check")
End If


End Sub
```

Intentionally left blank

ANNEX D Characteristics of randomised NEW WISPER sequence

Table D - 1: Summary of randomised NEW WISPER characteristics

File name	newnewwisper.TXT			
Number of records	95530			
Maximum peak	59			
Minimum valley	5			
First occurrence of maximum peak	91167			
First occurrence of minimum valley	29374			
Zero stress level	22			
Average R (half cycles)	0.212			
R (minimum valley/maximum peak)	-0.459			
Lag	1			
Autocorrelation for…	***peaks and valleys***	***peaks***	***valleys***	***ranges***
mu	34.3505	41.6734	27.0275	14.6458
variance	80.4188	23.4190	30.1695	11.2248
Autocorrelation	-0.4034	0.9727	0.9638	0.9227
Sum of all ranges	1399098			
Average range	14.64579342			

NB: characteristics expressed in levels

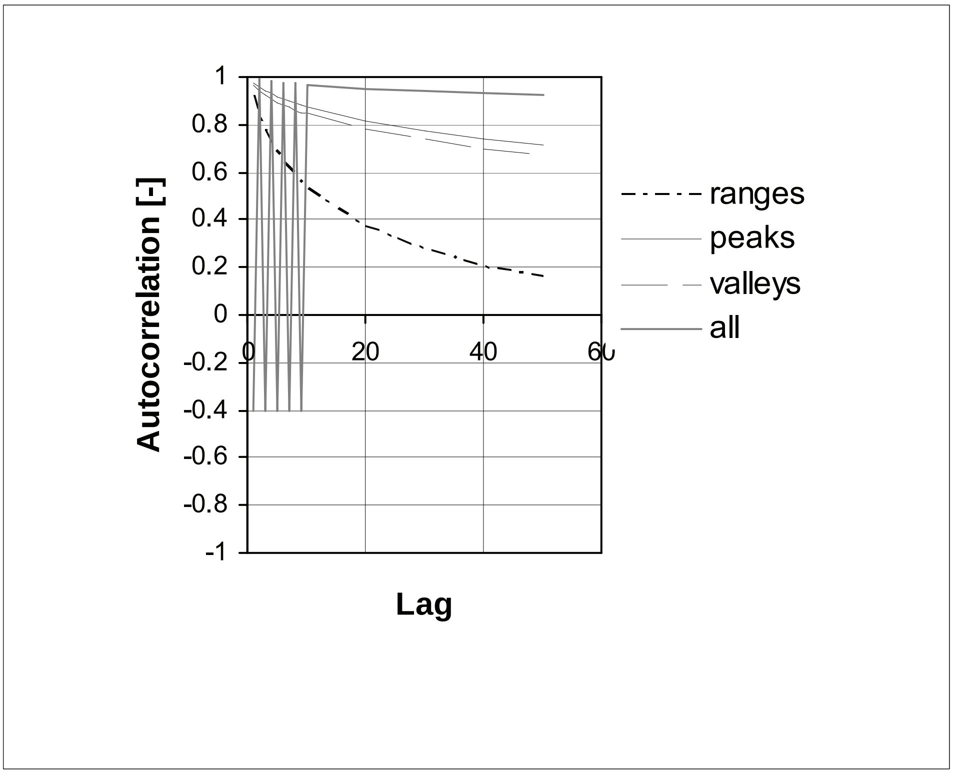


Figure D - 1: Autocorrelation plot for lag values 1-9, and 10, 20, 30, 40, and 50 (NEW NEW WISPER)