




Technical Report
NREL/TP-XXXXX
October 2012

MLife User’s Guide for Version 1.00

G. J. Hayman and M. Buhl, Jr.

Revised October 19, 2012
for MLife v1.00.00

NREL is a national laboratory of the U.S. Department of Energy, Office of Energy Efficiency & Renewable Energy, operated by the Alliance for Sustainable Energy, LLC.
Contract No. DE-AC36-08GO28308

Available electronically at http://www.osti.gov/bridge
Available for a processing fee to U.S. Department of Energy
and its contractors, in paper, from:
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
phone: 865.576.8401
fax: 865.576.5728
email: mailto:reports@adonis.osti.gov

Available for sale to the public, in paper, from:
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
phone: 800.553.6847
fax: 703.605.6900
email: orders@ntis.fedworld.gov
online ordering: http://www.ntis.gov/help/ordermethods.aspx

Cover Photos: (left to right) PIX 16416, PIX 17423, PIX 16560, PIX 17613, PIX 17436, PIX 17721

Printed on paper containing at least 50% wastepaper, including 10% post consumer waste.

Background

When striving for type certification of a wind turbine, engineers usually must perform a loads analysis for the certification agencies. This usually involves running hundreds or thousands of simulations modeling different conditions. Engineers usually have to generate extreme-event tables and fatigue-life predictions from the simulation output.

Engineers at the National Wind Technology Center (NWTC) at the National Renewable Energy Laboratory (NREL) had created a program called MCrunch that included these features and many more, but MCrunch must store the entire dataset concurrently in memory. For such a large number of simulations, the memory requirements would exceed the amount available in an engineer's typical computer. Staff at the NWTC recently addressed this problem by creating MExtremes for the extreme-event tables and MLife for fatigue-life predictions. These specialized tools process files sequentially, so only one data file is stored in memory at a time.

Introduction

MLife is a MatLab-based tool created to post-process results from wind turbine tests, and aero-elastic, dynamic simulations. MLife computes statistical information and fatigue estimates for one or more data files. The program reads a text-based settings file in conjunction with one or more time-series data files. Alternatively, the program can read parameter variables which were created using MatLab, outside of MLife.

The program generates results in the form of MatLab variables, text output files, and/or Excel formatted files. This allows you to make other calculations or present the data in ways MLife cannot.

The statistical calculations include minimum value, maximum value, mean, standard deviation, skewness, kurtosis, and maximum range.

The short-term fatigue calculations are based on each input file or an aggregate of all the input files and include:

- short-term damage-equivalent loads (DELs)
- damage rates

The lifetime fatigue calculations are based on the entire set of input files, a design lifetime period, an availability factor, and a wind speed distribution. These include:

- lifetime DEL
- lifetime damage
- time until failure

NREL distributes the source scripts, sample files, and a compiled version of MLife. If you do not own a MatLab license, you will have to use the compiled version. This requires you to download the free MatLab libraries (MCRInstaller.exe) from *http://wind.nrel.gov/designcodes/postprocessors/MLife/*. If you own MatLab and use the source scripts, you will need the Statistics Toolbox.

Retrieving Files from the Archive

You can download the MLife archive from our web server at *http://wind.nrel.gov/designcodes/postprocessors/*. The file has a name similar to *MLife_v1.00.00.exe*, but may have a different version number. Create an MLife folder somewhere on your file system and put this file there. When you double click on it from Windows Explorer, it will create some files and folders. To use the scripts, you need to add MLife's *Source*, *Source\datatablepackage*, and *Source\Rainflow* folders to the MatLab search path. If you do not have Visual Studio 2010 SP1 run-time libraries installed on your computer then you will also need to install *Redist\vcredist_x86.exe* (win32) or *Redist\vcredist_x64.exe* (win64).

Distributed Files

The files included in the MLife archive are:

AlphaChangeLog.txt	The list of changes to MLife.
Documentation.pdf*	The user's guide and theory manual in PDF format.
CertTest\Data.out*	Time-series data used for the certification tests.
CertTest.m*	Scripts for the certification tests.
CertTest.mlif*	The MLife input files used by the certification script.
*CertTest\NREL_Results**	The results from the developer's certification test.
Compiled\MLife_win32.exe	The compiled, 32-bit Windows, MLife executable for those who do not own MatLab.
Compiled\MLife_win64.exe	The compiled, 64-bit Windows, MLife executable for those who do not own MatLab.
Source.m*	MLife source files.
Source\datatablepackage.m*	DataTable object source files.
Source\Rainflow.**	Rainflow counting MEX-files and source.
Redist\vcredist_x86.exe	Visual Studio 32-bit run-time libraries
Redist\vcredist_x64.exe	Visual Studio 64-bit run-time libraries

DataTable Object Source Code

The DataTable object used in MLife was written by Paul Mattern. The code's license may be found in Appendix E. The code was obtained from the MatLab Central file exchange website:

http://www.mathworks.com/matlabcentral/fileexchange/24766

Rainflow Counting Source Code

The *rainflow* function used in MLife is based on code written by Adam Nieslony and is compiled as both a 32-bit and 64-bit MEX-files. The 64-bit MEX-files were generated using Microsoft Visual C++ 2010 and require that Microsoft Visual Studio 2010 run-time libraries be available on the computer they are run on. The code's license may be found in Appendix F. The code was obtained from the MatLab Central file exchange website:

http://www.mathworks.com/matlabcentral/fileexchange/3026-rainflow-counting-algorithm

The downloaded source file rainflow.c has been modified to include these changes:

- abs() has been replaced everywhere with fabs().
- The function now applies the Goodman correction to the damage cycle load ranges using a user-supplied fixed-mean load, or a fixed-mean load of zero.
- The user can supply the value of a partial damage cycle: *uc_mult*.

Certification Script

Before using MLife, you should run the certification script. It is a MatLab script called *mlife_certification_tests.m*; make sure that the MLife *Source, Source\Rainflow, and Souce\datatablepackage* folders are in the MatLab search path and the *CertTest* folder is the current directory, and then enter *mlife_certification_tests* at the MatLab command prompt.

MLife will run multiple times. The test procedure will generate outputs into the *CertTest\Results* folder and compare the results to those stored in the *CertTest\NREL_Results* folder. The procedure will write the differences between the output results to files called *Results\TestXX_differences.txt*, where *XX* is the test index number. Scan through the files; the only differences should be the date and time stamps in the headers. If you run MLife with a version other than the one used in development, there may be slight differences in the last digits of some of the numbers.

Running MLife

Prior to running MLife, make sure that the MLife *Source, Source\Rainflow, and Souce\datatablepackage* folders are in the MatLab search path. There are five ways to call the MLife program. The first three make use of an MLife settings file and one or more data files. The fourth relies on input parameters instead of a settings file, but still reads data from files. The final approach reads everything necessary to run MLife via input parameters and does not access files. Appendices B and C define the input and output data structures in detail.

Using an MLife Settings File

```
[Fatigue, Statistics]= mlife( settingsFile, dataDirectory, outputDirectory );
```

If the parameter *settingsFile* consists of a simple filename, the file must be located in the current MatLab working directory.

```
settingsFile = 'mysettingsfile.mlif'
```

Optionally, the *settingsFile* parameter may include a fully described path to the settings file.

```
settingsFile = 'C:\fatigue\mysettingsfile.mlif'
```

The parameter *dataDirectory* specifies the location of data files. However, if the file names specified in the settings file include a fully described path to the file, then this parameter is ignored.

```
dataDirectory = 'C:\fatigue\data\'
```

The parameter *outputDirectory* determines where output files are written.

```
outputDirectory = 'C:\fatigue\outputs\';
```

```
[Fatigue, Statistics] = mlife( settingsFile, dataDirectory );
```

In this case, the *outputDirectory* will be the same as the location of *settingsFile*.

```
[Fatigue, Statistics] = mlife( settingsFile );
```

In this case, the *dataDirectory* and *outputDirectory* will be the same as the location of *settingsFile*.

Using Function Parameters and Data Files

```
[Fatigue, Statistics] = mlife([], dataDirectory, outputDirectory,
                        StatisticsOptions, FileInfo, FatigueOptions);
```

In this case, a settings file is not read, but data files are used. The *StatisticsOptions*, *FileInfo*, and *FatigueOptions* structures must be populated by the caller in lieu of using a settings file. For example, the location and number of input files is defined by the *FileInfo* structure (see Appendix B). The first parameter is ignored, but the *dataDirectory* determines the full path location of input files and *outputDirectory* is used to set the location of any output files.

Using Function Parameters Only

```
[Fatigue, Statistics] = mlife( [], [], outputDirectory,
StatisticsOptions, FileInfo, FatigueOptions, inputData);
```

In this case, all necessary data is obtained via the input parameters and no files are read. The first two parameters are ignored, but the *outputDirectory* is used to determine the location of any output files.

Running MLife without MatLab

If you do not own a license to MatLab, you can run the compiled version of MLife, but it will work only with Microsoft Windows. We have done only the minimum required to make it work, but initial tests have shown that it is inferior to running within MatLab. For example, you will see none of the informational messages that would normally go to the MatLab *Command Window*. The program *appears* to do nothing at first and the command prompt returns immediately after you start it, which can give you the impression that it has completed. Eventually, output files will show up in the current folder.

To use the compiled version of MLife, you first have to install the MatLab Component Runtime (MCR). You need do this only once per version of MatLab. It must be compatible with the MatLab version used to compile MLife. You can find a compatible copy of the MCR Installer on the MLife Web page (see the Introduction). The file is very large, so be prepared to take some time downloading it. Once you have downloaded it, execute the file (*MCRInstaller_v*.*_win64.exe, or MCRInstaller_v*.*_win32.exe, where v*.* is the version number of the MCR*) to install it, following the prompts.

Next, you have to modify the Windows search path to include the MCR *win32* folder (for example, *C:\Program Files\MATLAB\MATLAB Component Runtime\v77\bin\win32*). To do this in Windows XP, right click on *My Computer* and choose *Properties*. In the *System Properties* window, click on the *Environment Variables* button on the *Advanced* tab. In the *System Variables* list, select *Path* and click on *Edit*. Hit the *Home* key on your keyboard and enter the appropriate path followed by a semicolon. Click on *OK* several times to save your changes.

Using MCR requires that you copy *MLife_win32.exe* (or *MLife_win64.exe*)to any folder that contains MLife input files you want to process. To process an MLife input file, open a *Command Prompt* window (*Start*, *Programs*, *Accessories*, *Command Prompt*) and change the current directory to one containing your input file(s). In the window, enter *MLife_win32 <input file>* (or *MLife_win64.exe)*. For example, in MLife's *CertTest* folder, enter *MLife_win32 test_01.mlif*.

Types of Output

For most analyses, you have the option of writing results to plain text files, Excel workbooks, or a combination of these. In addition if you are running MLife from within MatLab, you can return results in the structure variables outlined in Appendix C.

Plain text output files generated for individual files use the original data file's root name and append the *txt* extension. Output files generated for aggregate or lifetime analyses use a user-specified root name and also use the *txt* extension.

For output to Excel workbooks, MLife does not create a separate output file for each input file plus one for the aggregate. Instead, it creates a single workbook. These files utilize the user-specified root name and the *xlsx* extension.

Appendix D details the naming convention for all output files.

File Headers

All output files have similar headers. The headers contain the following information:

- Program name, version, compile date, and run date.
- Number of records used in the analysis and, for aggregates, the number of files comprising the aggregate.

The Settings File

Settings File Format

Use one of the sample *.mlif* files found in the *CertTest* folder as a template. Except for a few block tables and the list of data files at the end, you should not add or remove any lines from the sample settings file. For data files, list the file names one per line and enclose them in double quotes. A line beginning with *==EOF==* must come after the last data file. MLife ignores anything after the *==EOF==* line.

As long as the parameter(s) being read is (are) the first "word(s)" on the line and it is (they are) separated from the comment portion of the line with white space, MLife will not be affected. The amount of spacing is not important—use whatever looks good to you. You must separate any lists of numbers with white space—not commas. Block tables have their own special formats.

Title Line

The second line of the file is for a job title. The line can be of any length and contain anything you might find useful. Although MLife does not currently use this line for anything, it may someday include it in output files.

Job Options

There are currently five inputs in the Job Options section of the settings file. The first, *EchoInp*, allows you to request MLife to echo your settings file input parameters to another file so you can debug your settings file. The echo file has the same root name as your settings file and *echo* for an extension.

One common complaint about our old Crunch postprocessor is that if you add new channels to your data files, you have to change the channel numbers used in the settings file. We find that quite tedious and prone to error when we have more than 100 channels in our data files. We decided to allow users to specify channels by their names instead of numbers. If you want to take advantage of this feature, set *StrNames* to true and use the channel names surrounded by $ anywhere channel numbers would normally go. Here's an example:

$RootFxc1$ $RootFyc1$ $RootFzc1$ SFChans

If you enable *StrNames*, you don't have to use them exclusively; numbers will still work as MLife simply replaces all the *$ChanName$* strings found in the settings file with the channel numbers before processing the file.

Use the *RealFmt* parameter to specify the formatting of fixed-point numbers in the output. The standard MatLab *sprintf* syntax applies. You must enclose this string in double quotes.

RootName holds the root name of output files that contain analysis results based on using all input files. You must enclose this string in double quotes.

Input-Data Layout

MLife can automatically determine the number of columns and rows in your data files, or you can specify these things explicitly. If you want this, set the number of input channels to zero and don't give it a list of channels.

If you want MLife to parse channel names and units from the input file, specify the lines that contain such information. If you specify a zero for either, MLife assumes there is no such line. If you tell MLife which line contains names, it will use that line to determine the number of channels. It assumes that channel titles are a contiguous group of letters and/or symbols that do not contain any whitespace. For example, if the channel-names line is as follows:

Chan1 "Chan2"'Chan_3, Chan-4 " Chan 5",

MLife would find six channels whose titles would be <*Chan1*>, <"*Chan2*"'*Chan_3*,>, <*Chan-4*>, <">, <*Chan*>, and <*5"*> (without the angle brackets). If your input file has a title line similar to this:

“Chan 1” “Chan 2” “Chan 3”

and it has three columns of data, you cannot use the auto-detection feature for columns because MLife will think it has six channels (three named *"Chan*). If something such as this is the case, you should specify the channel information in the parameter input file.

Currently, you must either specify channel names and units within the settings file, or tell MLife to read them from the data file(s).

If you choose auto-detection of channels and give a zero for the line containing the units, MLife will not include any unit strings in the output. If that is the case, you must not specify the units for calculated channels.

If you want to specify channel layout, set *NumChans* to a number greater than zero. After the comment line describing the format for the channel table, enter *NumChans* lines with the channel names in double quotes, the units in doubles quotes, a fixed-point number for the scale, another for the offset, and another value for the partial safety factor (PSF) type. For the *PSF_Type* parameter valid options are 0 = do not apply a PSF, 1, 2, 3, or 4. The actual PSF for a specific *PSF_Type* is found in the Input Files section and is different for each design load case (DLC).

MLife cannot currently reorder or use a subset of channels, but it can rename channels from the original data file with this feature. It will apply scales and offsets as it reads the data. If you let MLife auto-detect channels, it cannot apply scales and offsets.

Unlike Crunch, MLife does not require that all files have the same number of lines. Like Crunch, it does require that the channel layout be the same for all files; that is, all data files must have the same number of channels and the same number of header lines.

MLife can also create new channels using typical MatLab expressions with references to other channels. (See the Calculated-Channels section.)

Calculated Channels

You can create new channels of data through the calculated-channels feature, which allows you to specify a single MatLab expression for each new channel. MLife numbers calculated channels in the order created; the number of the first one is one more than the number of input channels.

NumCChan specifies the number of new channels being created. If a calculated channel will need a seed to initialize a pseudo-random sequence, use *Seed* to store that value. Even if you don't use the random-number generator, you must include *seed* in the input file. A comment line describing the format for the lines describing the calculated channels is next. After that, enter one line for each calculated channel. These lines contain three fields; each should be enclosed in double quotes and separated by white space. The first field is for the channel name. The second is for the channel units (omit this if you are not using units). The last field is the MatLab-style expression. The time-series data is stored in a MatLab array named *timeSeriesData* and has dimensions of (*nLines*, *nChannels*), where *nLines* are the number of lines in the time-series and *nChannels* is the number of channels found in the time-series.

Figure 2 is an example of the calculated channels section of the settings file when string names are used (*StrNames=true*):

```
-----  Calculated Channels  ---------------------------------------
2            NumCChan The number calculated channels to generate.
1234567890   Seed     The integer seed for the random number generator
Col_Title    Units    Equation  Put each field in quotes.  Titles and…
"RootFMxy1" "(kN)" "sqrt( timeSeriesData(:,$RootFxc1$).^2 +
      timeSeriesData (:,$RootFyc1$).^2 )"
"RootMMxy1" "(kN·m)" "sqrt(timeSeriesData (:,$RootMxc1$).^2 +
      timeSeriesData (:,$RootMyc1$).^2 )"
```

Figure 1. Example input for calculated channels.

Load Roses

MLife can generate load roses. You can have multiple roses in a single MLife job. The user enters one line of input for each rose. The lines contain a quoted root name, a 0º load column, a 90º load column, and the number of sectors. One new column will be added to the time-series arrays for each sector. The names of the new columns will be the root name with a sequential sector number appended. The units for the two load columns must be the same. The two load columns can be any of the input channels or calculated channels.

The calculation of the new channels uses the following equations:

```
Angle = (iSector - 0.5)*180/NumSectors
Load(iSector) = Load0*COS(Angle) + Load90*SIN(Angle)
```

If the number of sectors is three, the three new columns will be for angles 30, 90, and 150º. You will never get 0 or 180º points. There is no need to go beyond 180º, because those loads will be the negatives of the load on the opposite side (the sector that is 180º less). MLife generates loads only for angles between 0 and 180º.

It may be advisable to use an even number of sectors so that you do not waste space by reproducing the 90º-load channel. I believe the IEC standard calls for 15º resolution for load roses. That requires 12 sectors at the following angles: 7.5, 22.5, 37.5, …, 172.5º.

Here is an example of the load rose section of the settings file:

```
-----  Load Roses  -----------------------------------------------------------
1             nLoadRoses        The number of load roses to generate.
Rose Name     Units    Channel1       Channel2     nSectors
"RootFxyc1"   "kN"     $RootFxc1$     $RootFyc1$   12
```

Figure 2. Example input for load roses.

Statistics

MLife can generate the following statistics:

- Minimum
- Mean
- Maximum
- Standard Deviation
- Skewness
- Maximum Range (Maximum-Minimum)

If you set *WrStatsTxt* to *true*, MLife will write a table of statistics for each input time-series using the naming convention *data_file_name*_Statistics.txt. When there is more than one time-series, MLife will use the aggregate of all the data to generate a statistics table with the naming convention *root_name*_Statistics.txt.

If you set *WrStatsXLS* to *true*, MLife will create an Excel workbook with one tab for each time-series and one for the aggregate if you processed more than one time-series. The tabs will have names using the root names of the time-series (or FileXX where XX is the index in the time-series list) and *Aggregate* for the aggregate statistics. MLife will use the root name of the settings file followed by *_Statistic.xls* to name the workbook.

MLife can also create summary files of the statistics for selected channels. Set *NumSFChans* to the number of channels you want processed this way and list the channels on the *SFChans* line that follows. MLife generates files of statistics for each requested channel. The summary output files will have the naming convention *channel_name*_Statistics.txt. Each summary file contains one line for each data file with the statistics following on the line. If *WrStatsXLS* is true the summary statistics will be written to a single workbook file called *root_name*_Summary_Statistics.xlsx.

Fatigue Analysis

MLife generates rainflow cycles of the variable-amplitude load ranges found in the time-series data. Cycle counting uses the one-pass method of Downing and Socie. You can tell MLife to discard unclosed cycles by setting *UCMult* to zero. Setting it to unity will cause MLife to count unclosed cycles as full cycles. However, we strongly encourage you to set *UCMult* equal to 0.5 unless you have a specific reason for choosing a different value.

MLife allows you to use a racetrack filter to eliminate small cycles (Veers et al. 1989) before you generate the rainflow cycles. The variable *FiltRatio* specifies what fraction of the maximum range of each channel will be the cutoff.

You tell MLife how many (*nFatigueChannels*) channels you want to process. If you are binning the load range cycles set *BinCycles* to true.

To calculate short-term DELs and damage rates, set *DoShortTerm* to true. The short-term DELs and damage rates are calculated for each individual time-series. Setting *DoAggregate* to true generates aggregate DELs and damage rates using all time-series. You must enter the equivalent frequency of the DEL (*EquivalentFrequency*). If you wish, DELs can be outputted as peak-to-peak ranges (*DEL_AsRange = true*) or as one-sided amplitudes (*DEL_AsRange = false*). To apply a Goodman correction to the load range cycles, set *GoodmanFlag* to 1. Setting *GoodmanFlag* to 0 tells MLife to use the load range cycles without applying a correction, and to use a zero fixed-mean load. A *GoodmanFlag* value of 2 tells MLife to compute results with and without the Goodman correction. If the Goodman correction is being used, you can specify what type of correction is used via the *DEL_Type* parameter. Setting *DEL_Type* to 1 corrects the load ranges to a specified fixed-mean load. Setting *DEL_Type* to 2 corrects the load ranges to a zero fixed-mean load. A value of 3 tells MLife to compute both types of corrections. For computations without the Goodman correction, a zero mean load is used.

To produce lifetime-related calculations, set *DoLife* to true. The fatigue cycles corresponding to an input time-series are scaled out across the design lifetime based on their design load case (DLC) grouping. There are three DLC groups in MLife. The first is a normal operation DLC. The second is an idling or parked DLC. The final DLC is a catch-all for discrete events which occur over the design lifetime. The normal operation and parked DLC group cycles are weighted using a Weibull distribution of wind speeds. The normal operation DLC is further weighted by the turbine *Availability* parameter and the parked DLC is weighted using (1-*Availability*).

WeibullShape specifies the shape factor of the Weibull distribution. *WeibullScale* is the scale factor of the distribution, if *WeibullShape* equals 2, then *WeibullScale* is interpreted as the mean wind speed of the distribution. Calculations are performed using wind speed bins, these bins are defined across three consecutive wind speed ranges, 0 to *WSin*, *WSin* to *WSout*, and *WSout* to *WSmax*, where *WSin* is the cut-in wind speed of the turbine, *WSout* is the cut-out wind speed, and *WSmax* is the maximum wind speed bin to include in the analysis. *WSMaxBinSize* establishes the maximum bin size for these wind speed ranges. The design lifetime is set using *DesignLife*. For example, to use a design lifetime of 20 years, set *DesignLife* to 630720000 (seconds).

Each fatigue channel requires a number of input parameters: associated Wohler exponent(s), load range binning values, the fixed-mean load, and the ultimate load. An example of this input table (green highlight) is shown in Figure 3.

MLife allows for fatigue calculations to be performed against multiple Wohler exponents. The second column of the channel table specifies how my exponents you want to use and the third column contains the list of exponent values. These values must be separated by whitespace, not commas.

You can specify to use either the number of load range bins (*BN*) or the width of each load range bin (*BW*) in column four. The fifth column must contain either the number of bins or the bin width depending on your choice of *BN* or *BW*, respectively. The first bin always starts at zero, and the last bin will contain the maximum time-series load value for that channel. If you are not binning the load ranges, placeholder values must still be present in columns four and five.

The fixed mean load for each channel is specified in the sixth column. There are three ways to specify the fixed mean load. 1) You may explicitly enter the value of the fixed mean. 2) You can enter a value of *AM* which tells MLife to compute the fixed mean using the channel's aggregate mean across all input files. 3) You can enter a value of *WM* which tells MLife to compute the fixed mean by weighting the channel's means on a per file basis using the specified Weibull distribution.

When the results are presented in table form, you may choose to group the fatigue channels. This allows you to create tables of results for related input channels. The blue highlighted section of Figure 3 shows how to specify your DEL groups. Specify the number of groups with *nDELGroups*. If *nDELGroups* = 0, all channels will be reported in the same table. You must then enter one line for each group. The first column specifies the name of the group and must be enclosed in double quotes. The second column specifies how many channels are included in the group. The third column contains a whitespace-separated list of channels indices. These indices correspond to the lines in the channel properties table entered directly above (green hightlight). In the example, the first group consists of the *$YawBrMxp$* and *$YawBrMyp$* channels.

If you set *WrShortTermTxt* to *true*, MLife will write tables of short-term DELs for each group specified in the settings file and use a file name *RootName* followed by _Short-term_DELs.txt. MLife will also generate a text file containing the short-term damage-rates in a file named *RootName* followed by _Short-term_Damage_Rate.txt

If *WrShortTermXLS* is set to true, MLife will generate the same short-term tables in an excel worksheet and use the value of *RootName* followed by _Short-term.xlsx.

If you set *WrLifeTxt* to *true*, MLife will write tables of lifetime damage and time until failure for each channel represented in the DEL groups table and use the value of *RootName* followed by _Lifetime_Damage.txt to name the plain text file. MLife will also create tables of lifetime DELs using the value of *RootName* followed by _Lifetime_Damage.txt to name the plain text file.

If *WrLifeXLS* is set to true, MLife will generate the same lifetime tables in an excel worksheet for the lifetime damage, time until failure, and a worksheet for the lifetime DEL calculations. MLife uses the value of *RootName* followed by *_life.xlsx to* name the excel workbook file.

```
-----  Fatigue  ----------------------------------------------------------------
6                nFatigueChannels    The number of rainflow channels.
0.0              FiltRatio           The fraction of the maximum range …
630720000        DesignLife          # of seconds in the design period.
1                 Availability        Fraction of the design life the …
true             BinCycles           Bin the rainflow cycles?
0.5              UCMult              Multiplier for binning unclosed cycles…
true             DoShortTerm         Compute short-term DELs and damage-rates.
true             DoLife              Do lifetime-related calculations?
2                WeibullShapeFactor  Weibull shape factor. If …
10               WeibullScaleFactor  Weibull scale factor. If …
3                WSin                Cut-in wind speed for the turbine.
21               WSout               Cut-out wind speed for the turbine.
31               WSmax               Maximum  wind speed value for the …
2                WSMaxBinSize        Maximum width of a wind-speed bin.
true             WrShortTermTxt      Write DELs to plain-text files?
false            WrShortTermXLS      Write DELs to an Excel workbook?
true             WrLifeTxt           Write lifetime results to plain-…
true             WrLifeXLS           Write lifetime results to Excel
1                EquivalentFrequency The frequency of the damage …
true             DEL_AsRange         true = report DELs as a range …
3                DEL_Type            1 = fixed mean, 2 = zero mean, 3 = both
2                GoodmanFlag         0 = no Goodman correction, 1 = use…
Channel#  NSlopes  SNslopeLst      BinFlag  BinWidth/Number  TypeLMF     LUlt
$YawBrMxp$    2       3 4            BN            20           WM       20250
$YawBrMyp$    2       3 4            BN            20         340.32     20250
$RootMxc1$    3       8 10 12        BW          236.34         AM       31875
$RootMyc1$    3       8 10 12        BW          367.72      5248.04     31875
$RootMxc2$    3       8 10 12        BN            40         774.87     31875
$RootMyc2$    3       8 10 12        BN            40        5252.84     31875
2                 nGroups
Name             NChannels            ChannelList
"Yaw"               2                      1 2
"Root"              4                      3 4 5 6
```

Figure 3. Example of the fatigue section of the settings file.

Input Files

Set the *FileFormat* parameter to 1 when using ascii-formatted FAST output files. Set *FileFormat* to 2 when using binary FAST output files. The remainder of the input files section is split up into three sets of design load cases (DLCs). The first set is reserved for input files which are representative of standard operating conditions. For lifetime calculations, the damage cycles from these files are weighted using the wind speed distribution and the *Availability* parameter. The second DLC represents parked or idling conditions. For lifetime calculations, the damage cycles from these files are weighted using the wind speed distribution and one minus the *Availability* (1-*Availability*) parameter. In the final DLC section, add input files representing discrete events that occur a user-defined number of times over the design lifetime. For lifetime calculations, the damage cycles from these files are weighted by the number of occurrences of the event.

The first line of the input files section is used to specify the number of files in the normal operation DLC, followed by a list of four partial safety factors (PSFs). This is followed by a line for each input file in the normal operation DLC. These lines contain the file names placed in double quotes. The PSFs are used when a PSF_Type was specified in the channels table.

The next line designates the number of files in the idling/parked operation DLC, followed by a list of four partial safety factors (PSFs). Next, there is a line for each input file in the idling operation DLC. These lines contain the file names placed in double quotes.

Next include a line containing the number of files in the discrete events DLC, followed by a list of four partial safety factors (PSFs). Next, there is a line for each input file in the discrete events DLC, containing the number of occurrences and the name of the input file. Figure 4 shows an example input files section.

```
-----  Input Files  ----------------------------------------------------------
1        FileFormat         1 = ascii format, 2 = binary format
2  1.1   1.3   1.5   1.7    (Weibull-Weighted Normal Operation: NumNormFiles…
"DLC1.2_1.out"
"DLC1.2_2.out"
3  1.1   1.3   1.5   1.7    (Weibull-Weighted Idling: NumIdleFiles, PSF1, …
"DLC6.4_1.out"
"DLC6.4_2.out"
"DLC6.4_3.out"
1  1.2   1.3   1.4   1.6    (Discrete Events: NumDiscFiles, PSF1, PSF2, …
56 "DLC4.1_1.out"
```

Figure 4. Example of the input files section of the settings file.

You may also replace the entire Input Files section and the final === *EOF* === line with a single reference to an include file. In this case, simply type a line starting with the @ symbol followed by the filename of the include file. For example,

```
-----  Input Files  ----------------------------------------------------
@myIncludeFile.txt
```

The contents of the include file should replicate the normal contents of the Input Files section of the settings file, including the trailing === *EOF* === line.

Limitations

MLife has the following limitations:

- All files must have the same channel layout.
- Computers must have sufficient virtual memory to contain the longest file's data. All time-series data are stored as four-byte, single-precision fixed-point numbers to save space.
- All files must have the same time step.
- Channel names and units may not contain any white space.

Future Work

- write out the input data after MLife applies scales and offsets, filters the data, and adds calculated channels.

Acknowledgments

Greg Hayman of the NWTC wrote MLife. This program is based on MCrunch which was written by Marshall Buhl. Jason Jonkman, Marshall Buhl, and Amy Robertson created the outline for the technical functionality of MLife and the associated fatigue equations. Rick Damiani performed a fatigue study using MLife and provided valuable feedback which has been incorporated into the software.

Feedback

If you have problems with MLife, please contact Greg Hayman. If he has time to respond to your needs, he will do so, but please do not expect an immediate response. Please send your comments or bug reports to:

Greg Hayman
NWTC/3811
National Renewable Energy Laboratory
1617 Cole Blvd.
Golden, CO 80401-3393
United States of America

Web: http://wind.nrel.gov
Email: greg.hayman@nrel.gov

References

Buhl, M.L., Jr. (2008). *MCrunch User's Guide*. NREL/EL-500-43139. Golden, CO: National Renewable Energy Laboratory, 15 pp.

Downing, S.D.; Socie, D.F. (1982). "Simple Rainflow Counting Algorithms." *International Journal of Fatigue*; Vol. 4, [N.1], pp. 31–40.

Moriarty, P.J.; Holley, W.E.; Butterfield, S.P. (2004). Extrapolation of Extreme and Fatigue Loads Using Probabilistic Methods. NREL/TP-500-34421. Golden, CO: National Renewable Energy Laboratory, 32 pp.

Sutherland, H.J. (1999). *On the Fatigue Analysis of Wind Turbines*. SAND99-0089. Albuquerque, NM: Sandia National Laboratories.

The MathWorks. “The MathWorks – MatLab and Simulink for Technical Computing.” http://www.mathworks.com. Accessed March 28, 2008.

Veers, P.S.; Winterstein, S.R.; Nelson, D.V.; Cornell, C.A. (1989). “Variable Amplitude Load Models for Fatigue Damage and Crack Growth.” Potter, J.M. and Watanabe, R.T., eds. *Development of Fatigue Loading Spectra*, ASTM STP 1006, p. 172.

Appendix A – MLife Input Parameters

Parameter	Datatype	Dimensions	Description
settingsFile	string	scalar	filename of the input settings file
dataDirectory	string	scalar	path where input files are located
outputDirectory	string	scalar	path where output files are written
StatisticsOptions	structure	scalar	statistics-related options for configuring the statistics analysis
FatigueOptions	structure	scalar	fatigue-related options for configuring the fatigue analysis
FileInfo	structure	scalar	file-related options for configuring the analysis
inputData	single	nFiles, nChannels	matrix of input channel data

Appendix B – Input Data Structures

StatisticsOptions

Field	Datatype	Dimensions	Description
DoStats	boolean	scalar	true = compute full statistics
WrStatsTxt	boolean	scalar	output text-based results
WrStatsXLS	boolean	scalar	output excel-formatted results
SumStatChans	integer	1, nSummaryChannels	array of channel indices. A separate statistics output file will be created for each channel.

FatigueOptions

Field Name	Datatype	Dimensions	Description
nFatigueChannels	integer	scalar	number of channels on which to perform fatigue-related calculations
DesignLife	integer	scalar	lifetime period in seconds
FiltRatio	single	scalar	fraction of the maximum range of each channel will be the filtered out of peak finding results array
BinCycles	boolean	scalar	determines whether rainflow cycles are placed into load-range bins
UCMult	single	scalar	weight of unclosed rainflow cycles, [0,1]
DoShortTerm	boolean	scalar	compute short-term DELs and damage rates for each input file
DoLife	boolean	scalar	compute lifetime damage, time until failure, and lifetime DELs
DoAggregate	boolean	scalar	compute short-term DELs and damage rates using an aggregate of all input files
weibullShapeFactor	single	scalar	shape factor of Weibull distribution, 2 = Rayleigh
weibullScaleFactor	single	scalar	scale factor of Weibull distribution, used if weibullShapeFactor ~= 2
weibullMeanWS	single	scalar	average wind speed of the Weibull distribution, used if weibullShapeFactor = 2
WSin	single	scalar	cut-in wind speed of the turbine
WSout	single	scalar	cut-out wind speed of the turbine
WSmax	single	scalar	maximum wind speed of the highest wind speed bin
WSMaxBinSize	single	scalar	maximum width of a wind speed bins
WrShortTermTxt	boolean	scalar	write short-term DELs and damage rates to text files
WrShortTermXLS	boolean	scalar	write short-term DELs and damage rates to an Excel file
WrLifeTxt	boolean	scalar	write lifetime results to text files
WrLifeXLS	boolean	scalar	write lifetime results to an Excel file
EquivalentFrequency	single	scalar	frequency of the damage-equivalent loads in Hz
ChanInfo	structure	1, nFatigueChannels	defines fatigue-related channel properties
DEL_AsRange	boolean	scalar	Output DEL values a ranges or amplitudes
DEL_Type	integer	scalar	1 = fixed-mean load, 2 = zero fixed-mean load, 3 = both

GoodmanFlag	integer	scalar	0 = do not use the Goodman correction, 1 = use Goodman, 2 = compute with and without
nGroups	uint32	scalar	number of fatigue groups for output tables
Groups	structure	1, nGroups	defines each fatigue group

FileInfo

Field	Datatype	Dimensions	Description
RootName	string	scalar	text which prepends most output filenames (see Appendix D)
TitleLine	integer	scalar	location of title (1 = start of file)
NamesLine	integer	scalar	location of channel names
UnitsLine	integer	scalar	location of channel units
FirstDataLine	integer	scalar	location of the start of channel data
TimeChan	integer	scalar	index of channel representing time
WSChan	integer	scalar	index of channel representing wind speed
nFiles	integer	scalar	number of files used
FileName	cell array	nFiles,1	list of file names
FileFormat	integer	scalar	format of input files. 1 = ascii format, 2 = binary
UserNames	Boolean	scalar	true if names and/or units are provided in settings file or input parameter, false if they are to be read from a time series input file
Names	cell array	1, nChannels	list of channel names
Units	cell array	1, nChannels	list of channel units
nChannels	integer	scalar	number of channels
Title	cell array	1, nFiles	list of file titles
nSamples	integer	1, nFiles	datapoints contained in each file
RealFmt	string	scalar	printf format string for floating point numbers
CalcChan	structure	1, nCalcChannels	information about derived (calculated) channels
nLoadRoses	integer	scalar	number of load roses tables
LoadRoses	structure	1,nLoadRoses	information about the load roses tables
PSFtype	integer	1,nChannels	PSF index for DLCs(i).PSF array, if you set an element to 0, then no PSF is applied.
DLCs	structure	1,3	information of design load cases (DLC) and associated files
DLC_Occurrences	integer	1,DLCs(3).NumFiles	number of times the files in the third DLC group occur over the design lifetime

FatigueOptions.ChanInfo

Field	Datatype	Dimensions	Description
Chan	integer	scalar	index of this fatigue channel in the main channel list
NSlopes	integer	scalar	number of slope values provided for the SN curve
SNSlopes	single	1,NSlopes	list of Wohler exponent values
BinFlag	string	scalar	'BN' means that the number of bins is specified by the user, 'BW' means the bin width is specified by the user
BinVal	single	scalar	either, number of load-range bins, or width of bin
TypeLMF	string	scalar	Determine the Load mean. 'WM','AM', 'value' where 'value' is a number specifying the magnitude of the fixed mean, 'WM' = weighted mean, 'AM' = aggregate mean
LMF	single	scalar	fixed mean
LUlt	single	scalar	ultimate load

FatigueOptions.Groups

Field	Datatype	Dimensions	Description
name	cell	1,1	name of the fatigue group
channelIndices	integer	1, nGroupChannels	indices of the fatigue channels to be included in this group indices are in the range [1, nFatigueChannels] not [1,nChannels]

FileInfo.LoadRoses

Field	Datatype	Dimensions	Description
Name	string	scalar	name of the load rose
Channel1	integer	scalar	index of first orthogonal channel
Channel2	integer	scalar	index of second orthogonal channel
nSectors	integer	scalar	number of sectors to compute

FileInfo.DLCs

Field	Datatype	Dimensions	Description
NumFiles	integer	scalar	number of input files for this DLC
PSF	float array	1,4	partial safety factors (PSF) for this DLC
DLC_Name	string	scalar	name of the DLC group

FileInfo.CalcChan

Field	Datatype	Dimensions	Description
Name	string	scalar	name of the channel
Units	string	scalar	units for the channel
Eqn	string	scalar	MatLab-compatible equation which will be used to create this channel's data

Appendix C – Output Data Structures

Statistics

Field Name	Datatype	Dimensions	Description
Minima	single	nFiles, nChannels	minimum value
MinInds	uint32	nFiles, nChannels	index of minimum value
Maxima	single	nFiles, nChannels	maximum value
MaxInds	uint32	nFiles, nChannels	index of maximum value
Range	single	nFiles, nChannels	maxima - minima
Means	single	nFiles, nChannels	mean value
StdDevs	single	nFiles, nChannels	standard deviation
Skews	single	nFiles, nChannels	skewness
Kurtosis	single	nFiles,nChannels	kurtosis
AggMinima	single	nChannels	minimum value across all input files
AggMinInds	uint32	nChannels	index of minimum value
AggMinFileNum	uint32	nChannels	index of input file where aggregate minimum is located
AggMaxima	single	nChannels	maximum value across all input files
AggMaxInds	uint32	nChannels	index of maximum value
AggMaxFileNum	uint32	nChannels	index of input file where aggregate maximum is located
AggRange	single	nChannels	aggregate range
AggMeans	single	nChannels	aggregate mean
AggStdDevs	single	nChannels	aggregate standard deviation
AggSkews	single	nChannels	aggregate skewness
AggKurtosis	single	nChannels	aggregate kurtosisn

Fatigue

Field Name	Datatype	Dimensions	Description
Channel	structure	1, nChannels	channel-related fatigue results
File	structure	1, nFiles	file-related fatigue results
lifetimeEquivalentCycles	single	scalar	number of equivalent cycles during design lifetime
RFPerStr	string	scalar	lifetime period expressed as a string
NumPeaks	double	nFiles, nChannels	number of peaks for a given channel in a given file
Peaks	cell array	nFiles, nChannels	variable length peaks arrays for each channel on a per file basis

Fatigue.Channel

Field Name	Datatype	Dimensions	Description
lifetimeDamage	double	1, nSlopes	accumulated damage over design lifetime. One result for each Wohler exponent
timeUntilFailure	double	1, nSlopes	time in seconds until failure. One result for each Wohler exponent
lifetimeDamage_NoGoodman	double	1, nSlopes	accumulated damage over design lifetime, based on uncorrected fatigue cycles. One result for each Wohler exponent
timeUntilFailure_NoGoodman	double	1, nSlopes	time in seconds until failure, based on uncorrected fatigue cycles. One result for each Wohler exponent
lifetimeDEL_FixedMean	double	1, nSlopes	lifetime DEL about the fixed mean load. One result for each Wohler exponent
lifetimeDEL_ZeroMean	double	1, nSlopes	lifetime DEL about a men load of zero. One result for each Wohler exponent

Fatigue.File

Field Name	Datatype	Dimensions	Description
channel	structure	1, nChannels	channel-related fatigue results

Fatigue.File.Channel

Field Name	Datatype	Dimensions	Description
DEL_FixedMean	double	1, nSlopes	short-term DEL about fixed-mean. One result for each Wohler exponent
DEL_ZeroMean	double	1, nSlopes	short-term DEL about a zero fixed-mean. One result for each Wohler exponent
DEL_NoGoodman	double	1, nSlopes	short-term DEL about a zero fixed-mean, without the Goodman correction. One result for each Wohler exponent
DamageRate	double	1,nSlopes	damage-rate for each Wohler exponent
DamageRate_NoGoodman	double	1,nSlopes	damage-rate for each Wohler exponent without the Goodman correction
binnedCycleCounts_FixedMeans	integer	1,nBins	the cycle counts for the binned fixed-mean corrected load ranges
binnedCycleCounts_ZeroMeans	integer	1,nBins	the cycle counts for the binned zero fixed-mean corrected load ranges
binnedCycleCounts_NoGoodman	integer	1,nBins	the cycle counts for the binned un-corrected load ranges

Appendix D – Output Files

Output File Naming Convention	Description	Settings File Parameter
*data_file_name*_Statistics.txt	data file statistics	WrStatsTxt
*channel_name*_Statistics.txt	channel summary statistics	NumSFChans > 0
*root_name*_Statistics.txt	aggregate statistics	WrStatsTxt
*root_name*_Statistics.xlsx	all statistics	WrStatsXLS
*root_name*_Summary_Statistics.xlsx	all summary statistics	NumSFChans > 0
*root_name*_Short-term_DELs.txt	all short-term DELs	WrShortTermTxt
*root_name*_Short-term_Damage_Rate.txt	all short-term damage rates	WrShortTermTxt
*root_name*_Short-term.xlsx	all short-term results	WrShortTermXLS
*root_name*_Lifetime_Damage.txt	lifetime damage and time until failure results	WrLifeTxt
*root_name*_Lifetime_DELs.txt	lifetime DELs	WrLifeTxt
*root_name*_Lifetime.xlsx	all lifetime results	WrLifeXLS

Appendix E – DataTable License

Appendix F – Rainflow License