

MFAT – A BASIC FATIGUE MODULE FOR μ ETA-POST

¹Anders Jonsson*, ²Martin Sjöberg, ²Johnny Grenwald

¹DynaMORE Nordic AB, Sweden, ²BAE Systems Hägglunds AB, Sweden

KEYWORDS – Fatigue, Python, rainflow count, S-N-curve

ABSTRACT

A basic fatigue post-processing module for μ ETA-Post has been developed, consisting of Python scripts and an easy-to-use interface in the form of a User toolbar. The fatigue analysis is based on a simplified critical plane approach, where the critical direction is taken as the direction of the absolute maximum principal stress. Alternatively, the user may specify a projection direction. The stress history can be obtained either from a sequence of states, or by linear superposition of states scaled by curves. A standard routine for rainflow count is used for obtaining amplitude and mean stresses, and cycle counts. Fatigue strength data is input in the form of S-N-curves, in an easy-to-edit text-file-format. The fatigue material data is assigned to the elements via the property definitions of the corresponding keyword file. Linear damage accumulation (Miner's rule) is used for computing the fatigue damage. For each processed element, the maximum principal stress and amplitude stress, and calculated fatigue damage, are stored as scalar results, which can be visualized as fringe plots. This means that further post-processing of the fatigue analysis results is easily performed using the other built-in tools of μ ETA-Post. The module was developed using the built-in scripting tools of μ ETA-Post, which with its online help and powerful script editor offers an efficient programming environment.

1. INTRODUCTION

Fatigue strength in general is an important part of the customer experienced quality of a product. While a failure due to a dramatic overload or crash loading may be accepted by the customers, since then the connection between the loading and resulting failure is immediate and obvious, fatigue failure is caused by a very high number of small loadings, which in general are experienced as highly normal loadings by the customer. The coupling between the loading and the fatigue failure is not obvious to the everyday customer, see Figure 1. The scatter in loading and material properties also add a stochastic side to the nature of fatigue failure, which may further increase customer irritation. Many examples through the history, for example the DeHaviliand Comet airplanes and the Eshcede ECE train disaster, show that fatigue failure also can have catastrophic consequences, including loss of lives.

This makes it is important to incorporate fatigue analysis in an efficient and user friendly way in the design flow, especially so in the automotive and generally in the vehicle industry. This is a segment of the manufacturing industry where ANSA and μ ETA already are the preferred tools for general pre- and post-processing work. μ ETA-Post offers a powerful platform for general post-processing, and by the use of session files, Python scripts and User toolbars, a high level of customization can be achieved. By integrating the fatigue analyses into μ ETA, a consistent work flow is facilitated, where the fatigue analysis can be performed in the same way by different CAE engineers.



Figure 1 – Left image: in a crash or overload situation, the relation between the loading and the resulting deformation is often obvious to the everyday customer. Right image: fatigue failure often comes unexpectedly to a customer, without any obvious coupling to the current loading. (Failure of a bicycle handlebar stem).

2. FATIGUE ANALYSES AT BAE SYSTEMS

The customer requirements for all terrain vehicles, see Figure 2 for an example, are in general extremely high, but with different focus for different customers (military, rescue service, energy distributors etc.). One common requirement is however the demand for high reliability in service, under rough conditions. A fatigue failure causing a vehicle stop could potentially have fatal consequences in a combat situation.

At BAE Systems, fatigue analyses are incorporated as a central part of the design work of every development project. The fatigue analyses are based on experience from testing (ranging from materials testing via component testing to full-scale field tests) and also use in practical operations, including feedback from customers. At BAE Systems Hägglunds, μ ETA-Post is the premier post-processing tool, and incorporating fatigue analysis capabilities will help streamline the analysis work further.



Figure 2 - The BvS 10 MK II all terrain vehicle from BAE Systems Hägglunds.

2. DEVELOPMENT OF THE MFAT TOOLBAR

The development work (coding, de-bugging, verification) of mFAT was done using the built-in tools for customization of μ ETA – Post (1): the Toolbar Designer and the Script Editor (2). The work was done in close cooperation with the users at BAE Systems. The flexibility of μ ETA-Post, by the use of toolbars, session commands, and scripts, makes it very easy to customize in accordance with the user's requirements. In mFAT, communication with the user is mostly done explicitly via user actions (buttons, checkboxes, file textboxes etc.) but partially also implicitly, based on what information is present in the current μ ETA database. For example, one requirement was that mFAT should be able to compute fatigue damage either from a time history composed by a sequence of loaded states, or from a synthetic stress history defined by linear superposition of loaded states scaled by curves. If curves exist in a 2d plot window with a specific name, linear superposition is performed, otherwise a sequence of state is processed. There is no button or checkbox controlling what type of analysis is performed, it is simply the name of a 2d plot window that decides.

Crating the GUI using the Toolbar Designer

The GUI of mFAT, see Figure 10, was created using the Toolbar Designer of μ ETA-Post (available from User Toolbars > Toolbar Designer). User toolbars in general are very convenient for coupling scripts or session files to a GUI, providing numerous ways of interaction between the script and the user, and the Toolbar Designer makes it very easy to create user friendly GUIs, including tooltips and default values. Even though the user toolbars are aimed at interacting with μ ETA session files, it is easy to interact also with Python scripts by the session command "script execute ...".

More advanced GUI:s can be created using BCGUI, which also has a built-in design tool in μ ETA-Post, accessible from the Script Editor > Tools > Designer, see Sections 4.5 and 4.6 of Ref. (2).

Python programming using the Script Editor

With some previous experience from programing, but not in Python, it is still fairly easy to create quite complicated Python scripts after only a short learning period. Python programming help is available from many internet resources, the foremost is perhaps www.python.org (3). Also, basic programming help for Python is available in the μ ETA – Post documentation (2). The built-in help functions of the Script Editor on the μ ETA – Python API is invaluable for obtaining guidance regarding how to access the μ ETA-Post database, and help is provided to the user in a very efficient way. Almost every function comes with a small script primitive exemplifying how to use it, which is very helpful for users just starting to write scripts for μ ETA, see Figure 3.

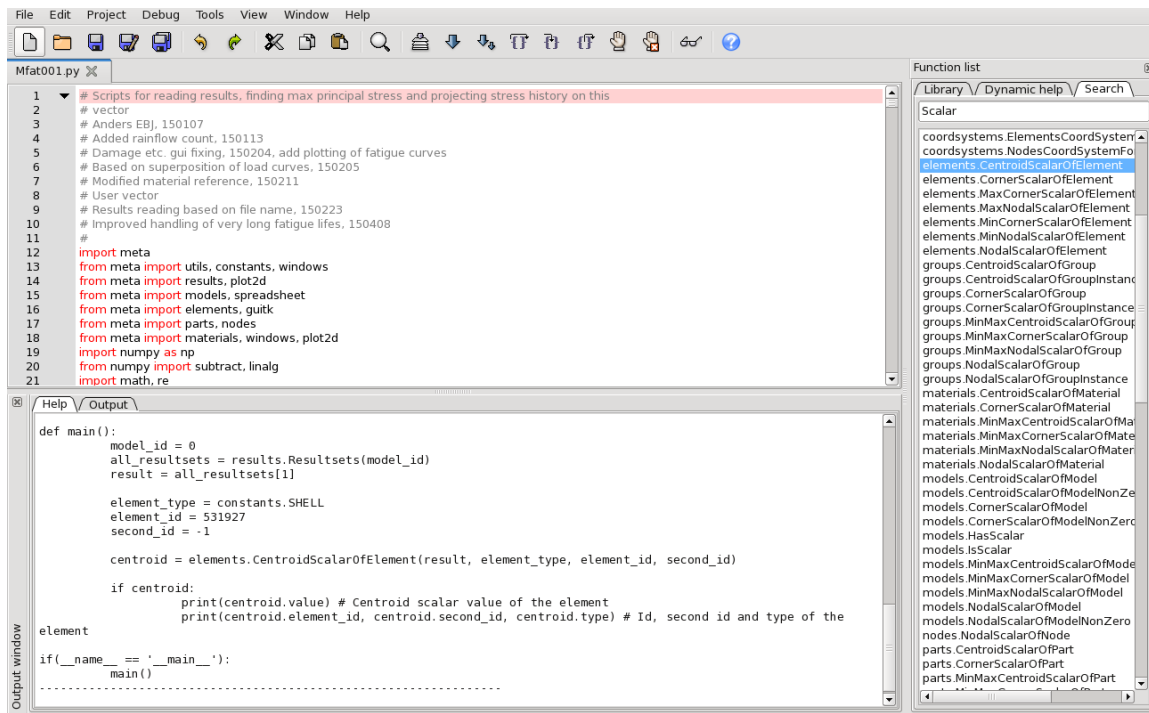


Figure 3 – The Script Editor programming environment. The help window shows a script primitive for using CentroidScalarOfElement.

The possibility to use other available Python modules, such as the math and the linalg library, etc. also saves time, since many functions are already available to the user. For example, in mFAT, the function `linalg.eig()` was used for obtaining principal stresses as eigenvalues of the stress tensor. The possibility to “communicate” with μ ETA-Post using session commands from within scripts (using the `utils.MetaCommand`) provide possibilities for alternative solutions. When coding the mFAT package, some effort was put into handling unexpected situations in a user-friendly way, providing feedback to the user in the form of warning or error messages. This is conveniently handled in Python, the exception handling being one of the strongpoints of the programming language.

The Python script of mFAT is built-up in a function-based way, the object-oriented functionality of Python is in general not utilized. A coarse flowchart of the main fatigue analysis procedure is shown in Figure 4.

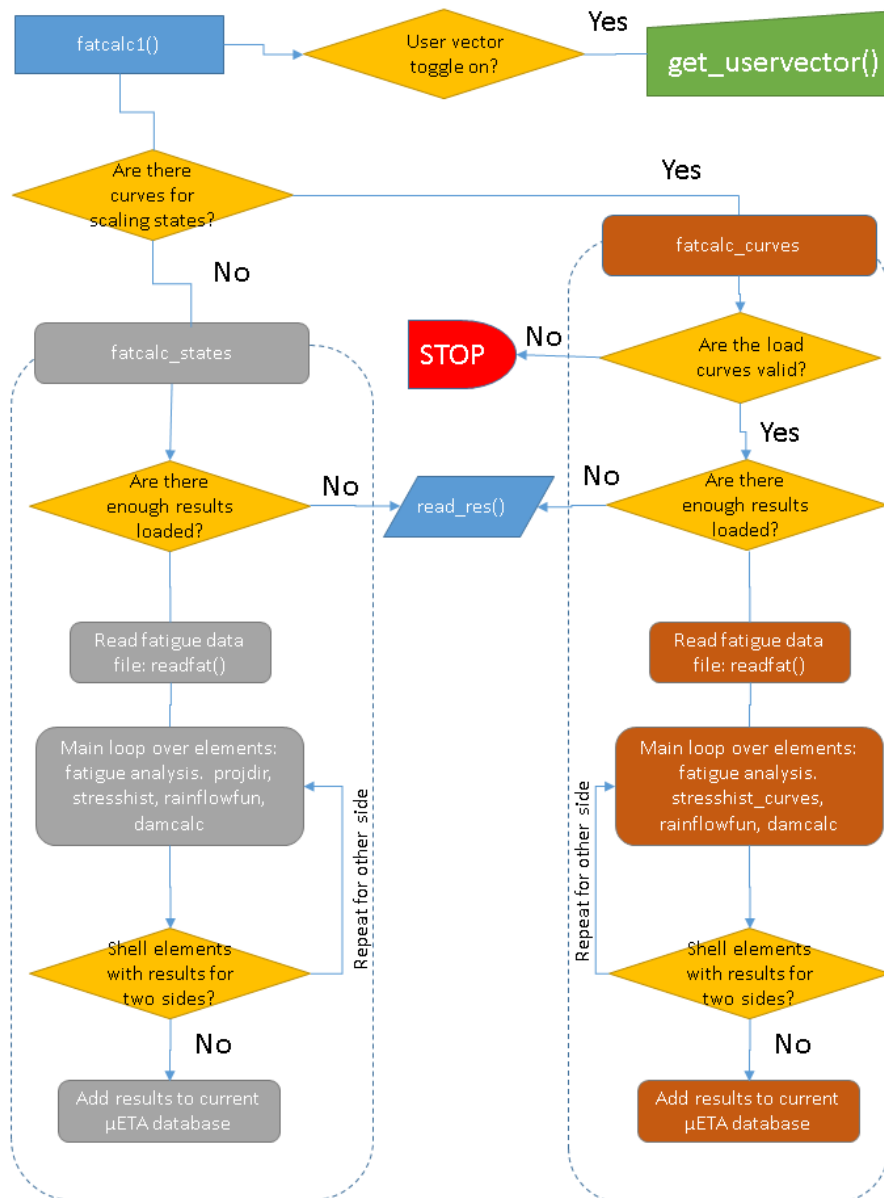


Figure 4 - Coarse flowchart of the fatigue analysis routine.

De-bugging and verification

The subroutines were to a large extent first tested and verified separately, and in later stages the functionality of the complete script was tested by a simple example. Python as a programming language provides quite good feedback to the programmer regarding syntax errors or unexpected events (such as trying to access a list beyond its last element) in a stable way (avoiding segmentation faults or similar). Results were compared to hand calculations and previous implementations in FORTRAN and Matlab/Octave. De-bugging of the GUI and general functionality of mFAT was done in close cooperation with the users at BAE Systems.

3. BRIEF THEORETICAL BACKGROUND

In this section, some of the theoretical background to the implemented fatigue analysis algorithm is presented. A good overview of fatigue and fatigue analysis in general is presented in Refs. (6), (7). For details on Python programming, the reader is referred to Ref. (3).

Principal stresses and stress projections

In order to obtain a scalar stress measure, a very basic critical plane approach is used in mFAT. It is assumed that the normal to the critical plane, for each element respectively, is given by the direction corresponding to the maximum absolute value of the principal stress (this will also be the direction that maximizes the normal stress). This means that the stress history, for each visible element, must be processed twice, sequentially:

1. The stress history (either from a sequence of states, or by linear superposition of states scaled by load curves) for element k is processed, and the principal stresses σ_i^k and corresponding directions \mathbf{e}_i^k are computed for each time step. When all time steps are processed, this will give the maximum absolute principal stress σ_{\max}^k and corresponding direction \mathbf{e}_{\max}^k .
2. Process the stress history for element k again, now projecting the stress tensor on the direction \mathbf{e}_{\max}^k . This will give a time history of the normal stress, relative to a fixed direction. On this time history, rainflow count (see Section **Error! Reference source not found.**) is applied.

Rainflow count

Once the stress history is obtained, it can be subjected to cycle counting, in order to obtain the corresponding amplitude and mean stresses. In mFAT, this is done using rainflow count. The method of “rainflow counting” for determining load cycles from a general load history was first mentioned by Endo (8). The algorithm of the present mFAT implementation is based on Refs. (9), (10), and follows the ASTM standard (12) for rainflow counting. The present implementation of mFAT (in Python) was verified against previous implementations in FORTRAN and Matlab/Octave, and also against other publically available implementations, for example Ref. (11).

A short description of the rainflow count method follows (see also Figure 5). First, the signal is cleaned, so that only the local extremes are kept. The remaining history is rotated with the time axis downward, and the linear curve segments are treated as they would form a “pagoda roof”. The cycle counting is done by tracing an imaginary rain drop dripping down these roofs, following several rules.

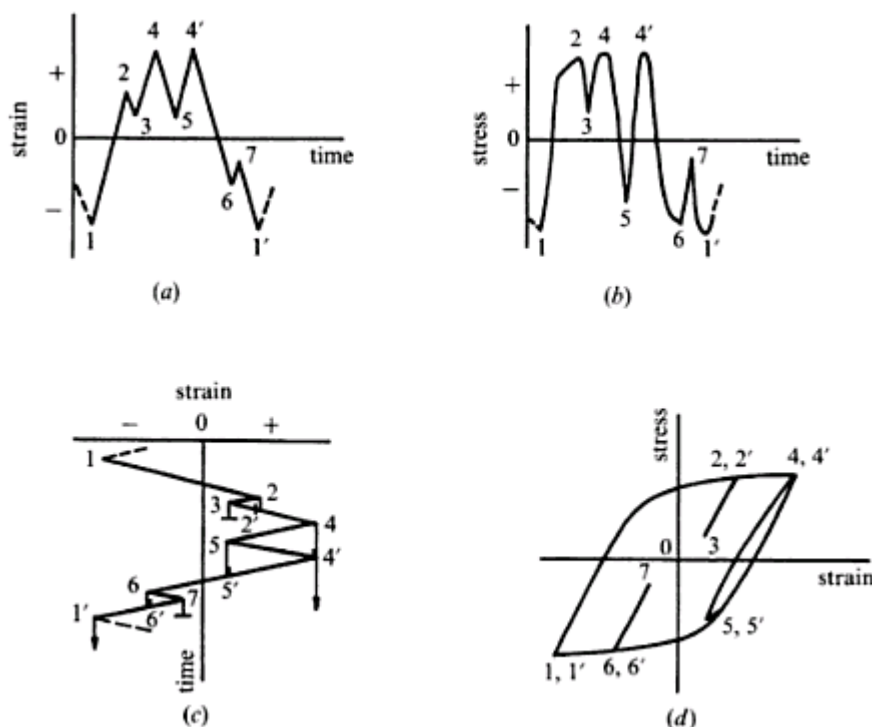


Figure 5 - Rainflow counting of cyclic loading. From Ref. (7).

It shall be noted that by use of the rainflow counting algorithm, it is implicitly assumed that the order of the loadings does not matter to the fatigue life of the structure, thus all sequence effects are neglected.

Linear damage accumulation

Once the amplitude and mean stresses, with corresponding cycle counts are obtained from the stress history via the rainflow counting procedure, the fatigue damage for each element k is computed using linear damage accumulation according to Palmgren-Miner:

$$D^k = \sum_{i=1}^l \frac{n_i^k}{N_i^k}$$

where n_i^k is the cycle count from the rainflow count (either 0.5 for a half cycle or 1.0 for a full cycle¹) and N_i^k is the allowed number of cycles at the current loading for the material of element k , computed from the S-N curve using Basquin's equation, as

$$N_i^k = \begin{cases} \left(\frac{C_1^k}{\sigma_{a,i}^i} \right)^{m_1^k}, & \sigma_a^i \geq \sigma_u \\ \left(\frac{C_2^k}{\sigma_{a,i}^i} \right)^{m_2^k}, & \sigma_a^i < \sigma_u \end{cases}$$

where $\sigma_{a,i}$ is the stress amplitude of cycle i and C^k and m^k are fatigue strength parameters for the material of element k (also compare Figure 6). No mean stress correction is currently implemented in mFAT, but it would be easy to add also such a feature, according to for example Goodman or Gerber.

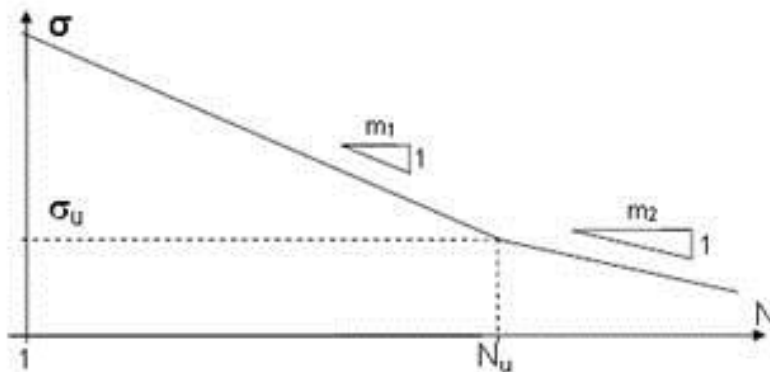


Figure 6 -S-N parameters.

4. USING THE MFAT TOOLBAR

In this Section, a quick overview of how to use the mFAT - package is given. Numbers in the list refer to annotations in Figure 10.

1. Load the geometry of the FE-model to be analyzed: Read Results > Geometry. Select the keyword file (.key or .k for LS-DYNA, .inp for Abaqus).
2. Open the mFAT – toolbar from User Toolbars. Load the stress results. This can be done manually, but is most conveniently done using the “Read stresses” – button, but to some extent automatic reading of stress results is implemented. Results will only be read for the visible parts.
3. Load fatigue material data, using the Fatigue data file – input filed. The S-N curves can be visualized by the “Plot fatigue curves” – button, see

¹ Bucket sorting is not done, each loading (σ_m , σ_a) is treated separately, either as a full cycle or a half cycle.

Figure 7.

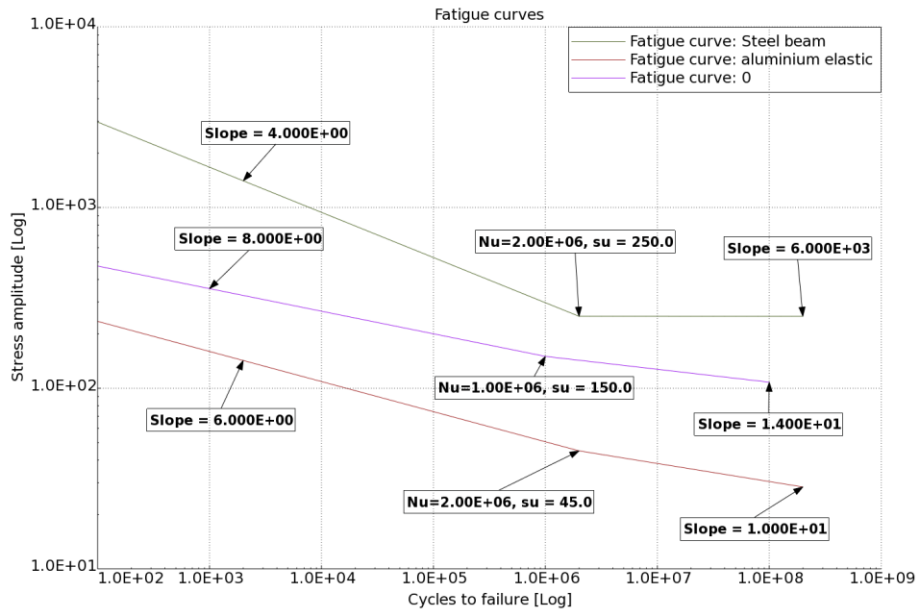


Figure 7 - Example of plotted fatigue curves, with annotations showing slopes.

4. Perform the fatigue analysis, by clicking on the “Calculate fatigue damage” - button. The fatigue analysis will only be performed on visible shell and solid elements.
 - a. By default, a fatigue analysis of the stress history defined by the loaded states is performed.
 - b. It is also possible to perform a fatigue analysis of a stress history defined by linear superposition of loaded states. For this option to be active, a 2d Plot window with the title “LoadcaseScalefactors” must exist (this window can be created from the toolbar), and contain load curves defining the scale factors of the states, see Figure 8.

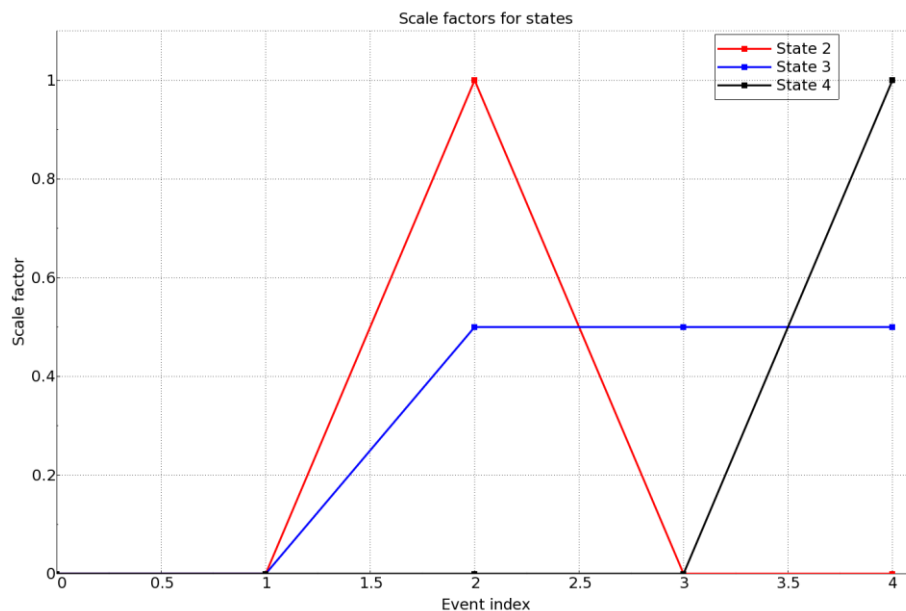


Figure 8 - Example of load curves for scaling loaded states.

The curves can be linked to the states, either by their ID (for example curve ID 5 will scale loaded state number 5) or by a spreadsheet table (if it exists) named “LoadcaseScalefactors”, see Figure 9.

	A	B	C	D
1	1	2	curve ID 1 will scale state number 2	
2	2	3	curve ID 2 will scale state number 3	
3	3	38	curve ID 3 will scale state number 38	

Figure 9 - Example of spreadsheet for association of load curves to states. Only columns A and B are read by the script, leaving space for comments.

5. A state “Fatigue damage of states/curves” will be created, containing the following scalar results: Fatigue damage, Max stress amplitude and Absolute Max principal stress. These results can be fringe plotted as any other scalar results in μ ETA-Post. Further post-processing of fatigue analysis results, such as identification of hot-spots or critical areas, and report generation, can be performed using the built-in functionality of μ ETA-Post. It is also possible to save the results for further post-processing at a later stage in .metadb – format.

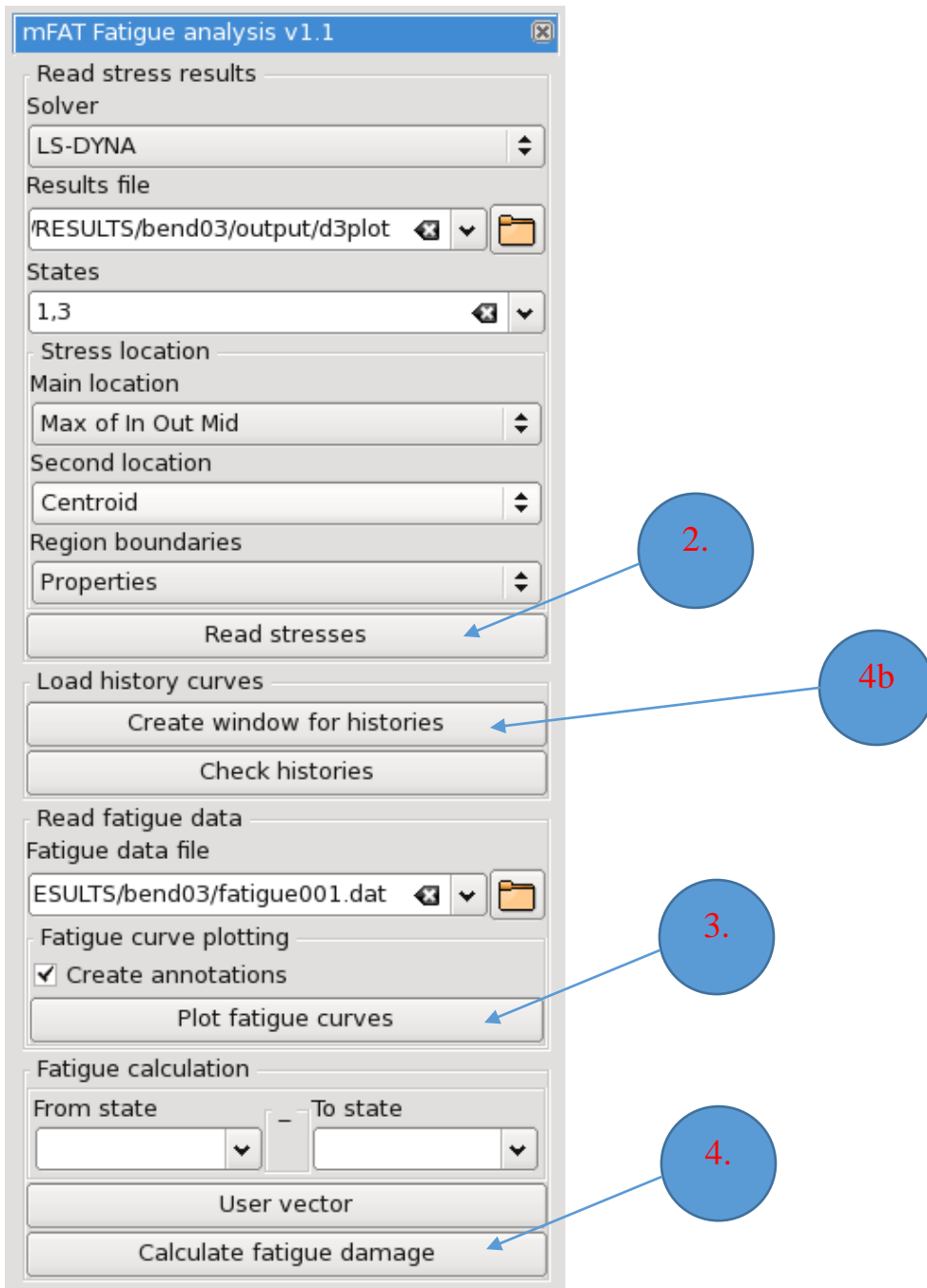


Figure 10 – GUI of the mFAT – toolbar. Numbers refer to points in the list above.

5. NUMERICAL EXAMPLES

In order to demonstrate the capability of the mFAT tool, a set of fatigue analyses of a crank shaft assembly² was studied. The crankshaft was meshed with 267000 2nd order tet elements. Elastic material behaviour was assumed for all included components. First, a basic stress analysis of the crank shaft only was performed (using LS-DYNA implicit), applying Y and Z-loadings to the four different cranks (totally 8 load steps), see Figure 12. Then, a fatigue analysis corresponding to superposition of the Y- and Z-loadings, according to a firing sequence of 1-3-4-2, was set up using simplified cylinder pressure curves, see Figure 13. A uniform sampling was used, resulting in totally 21 synthetic states. An assumed S-N curve according to Figure 11 was used for the fatigue analyses. For the complete crankshaft, the required analysis time was about 2.6 hours, see Table 1, using a normal work station with Intel Xeon(R) CPU E31240 running at 3.30GHz. If the element count is limited to 5577, by focusing on the interesting radii of the crankshaft, the analysis time is reduced to about 2.5 minutes. This illustrates that if the critical areas are well known, significant savings in analysis time can be made if the user focuses the fatigue analysis to these areas. Fatigue analysis results are presented in Figure 14. Identical fatigue damage results in the critical radii are obtained, corresponding to a fatigue life of 917000 cycles.

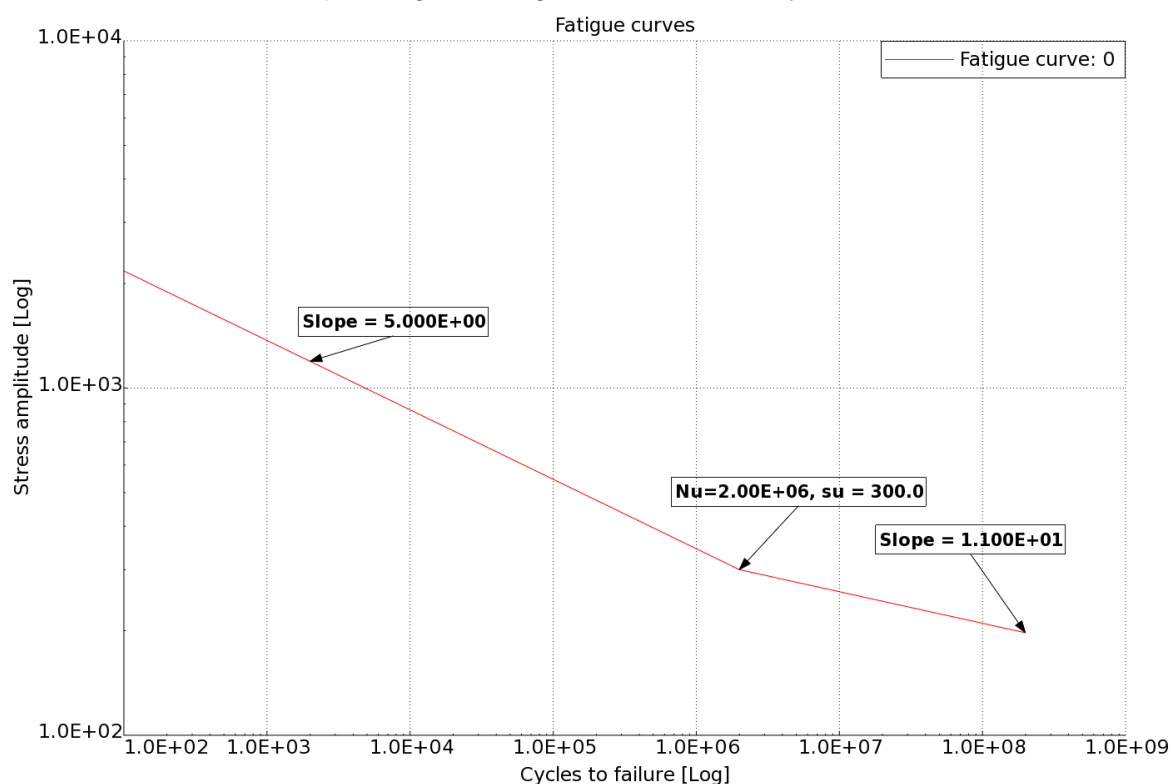


Figure 11. Assumed S-N curve for the crank shaft material.

Table 1 – Performance comparison for different mFAT analyses

Case	Element count	Number of states	Time / s
Superposition of states	267000	21	9219
Superposition of states	5577	21	149
Sequence of states	318509	8	525

² Based on the ANSA tutorial "Solid_mesh_assembly.pdf". It shall be emphasized that this is merely a numerical example, not an attempt at making a proper crankshaft system fatigue analysis.

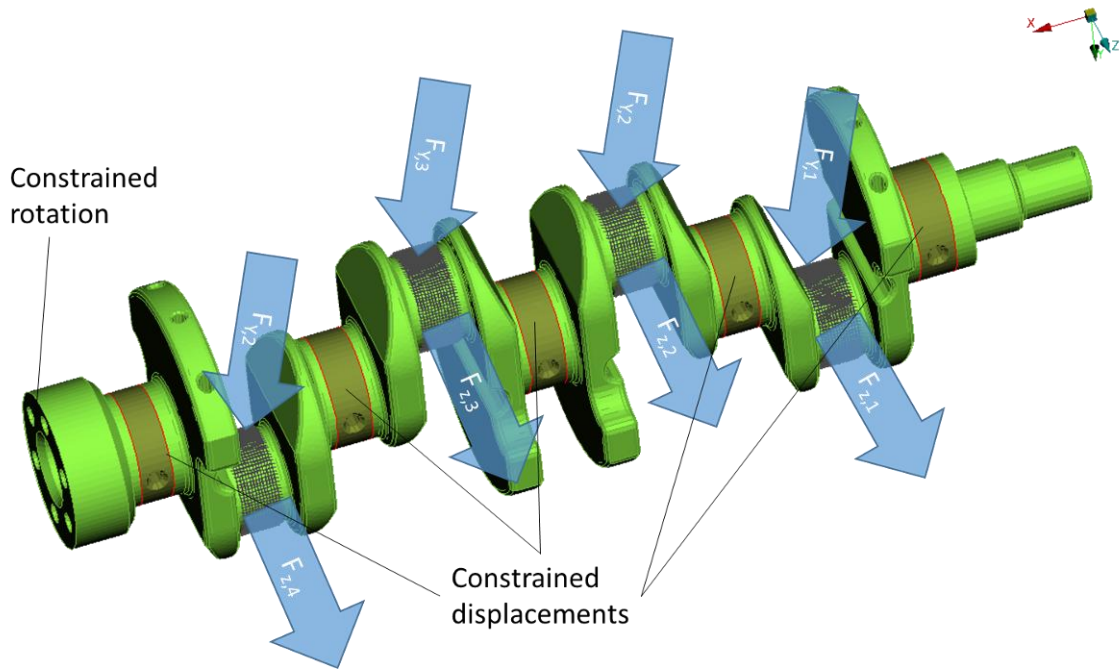


Figure 12 – Set up of the unit load cases for the crank shaft. The (Y, Z) displacements are constrained at the surfaces indicated by dark green, and the crankshaft rotation is constrained at the end towards the flywheel.

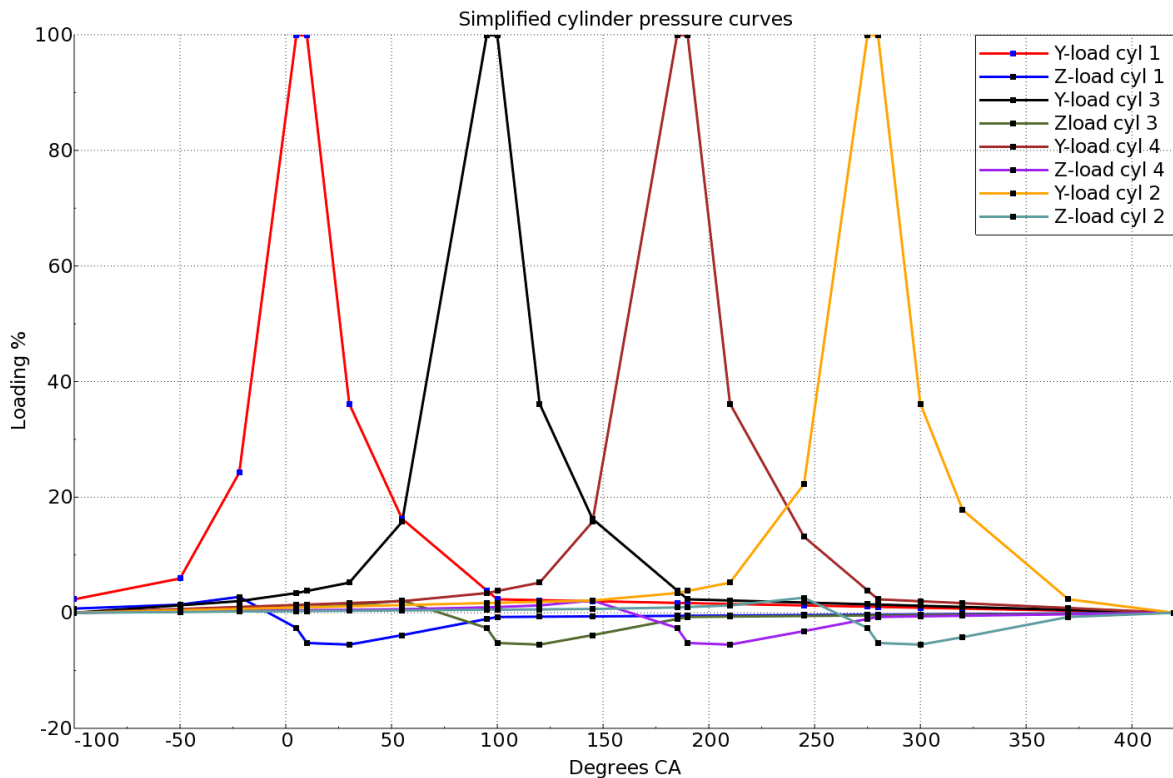


Figure 13 - Simplified cylinder pressure curves for Y and Z loading of the crankshaft.

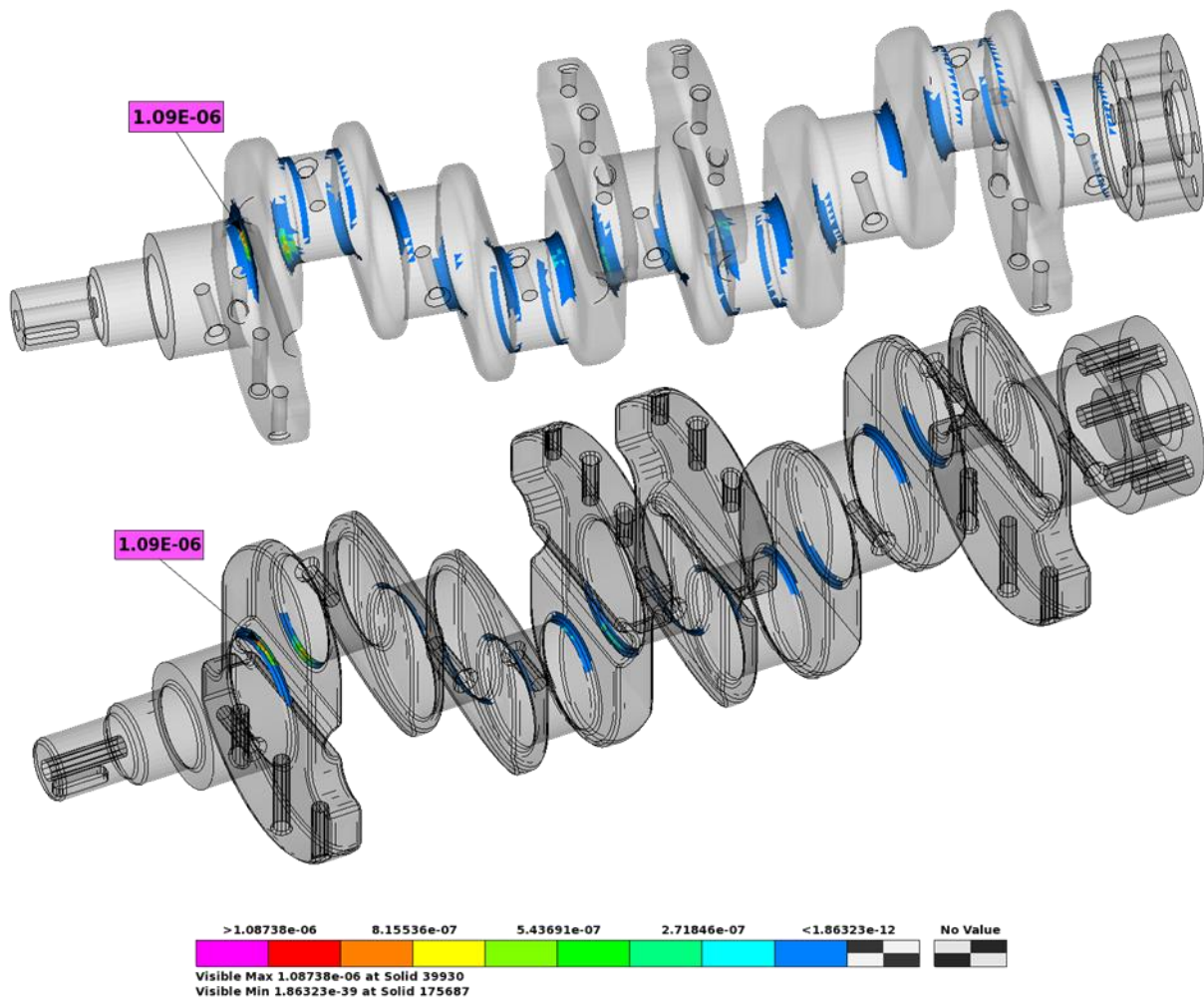


Figure 14 - Calculated fatigue damage for the synthetic stress history of the crankshaft. The top image shows results for the complete analysis, while the bottom image shows results for the reduced analysis (radii only). Identical results are obtained, with a maximum fatigue damage of $1.09E-6$.

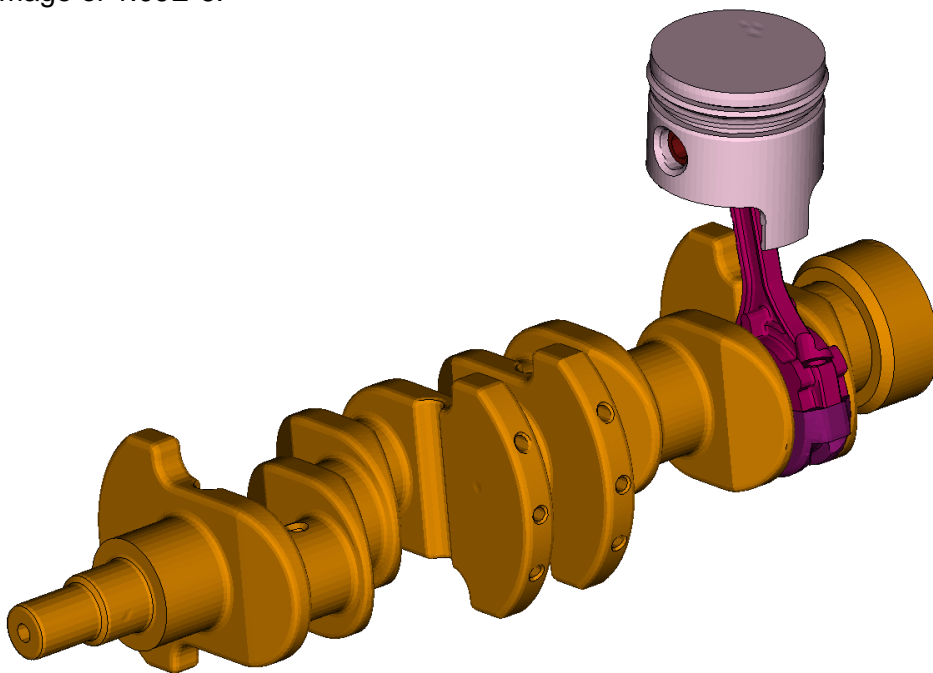


Figure 15 – Geometry of the crankshaft assembly.

In a second example, an assembly consisting of the crankshaft, conrod, piston pin and piston (see Figure 15) was analysed, for a sequence of 8 different cylinder pressures and crank angles. Contacts are modelled for all bearing surfaces. Prescribed boundary conditions were used to rotate the crankshaft. The cylinder pressure was applied as a distributed loading to the top face of the piston. The cylinder liner and main bearings were replaced by simplified boundary conditions. Also this case was solved by LS-DYNA implicit. For the fatigue analysis of this example, different fatigue curves were assigned to the parts by material name, see Figure 16. The fatigue analysis was performed based on a sequence of states, which means that the principal stresses computed by μ ETA-Post's hard-coded routines during results loading can be used. This saves significant time compared to the case when mFAT / Python computes the principal stresses of the synthetic stress history from load curves, compare Table 1. The fatigue analysis results are shown in Figure 17. The critical component with the assumed fatigue data is the piston, with a fatigue life corresponding to 530000 cycles.

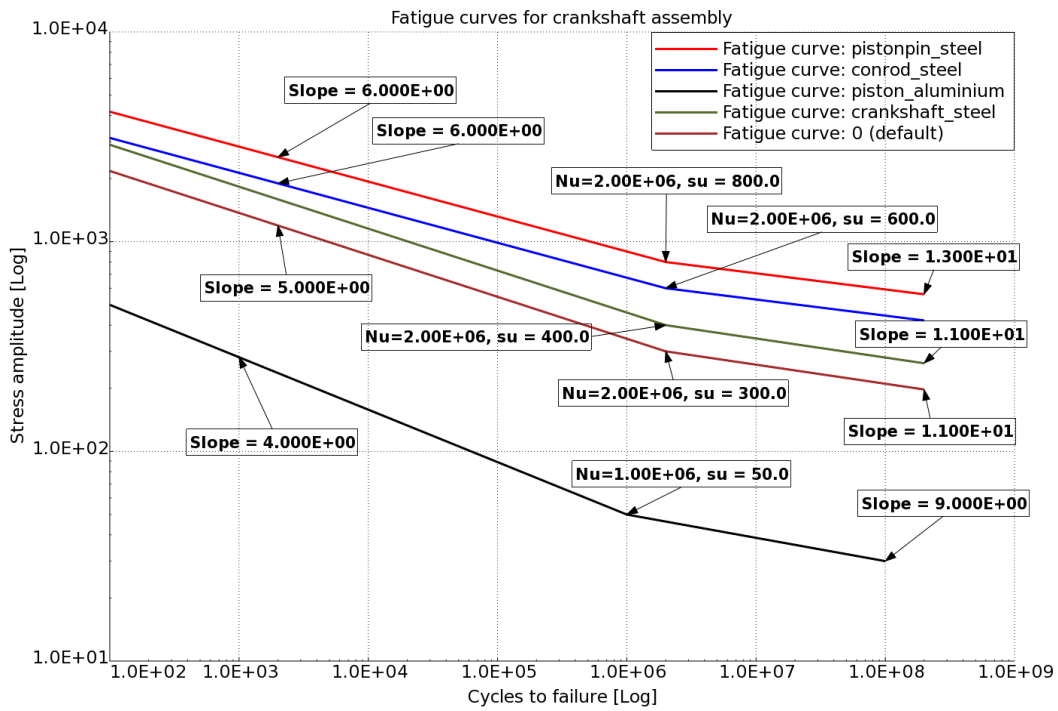


Figure 16 - Assumed S-N curves for the materials of the crankshaft assembly.

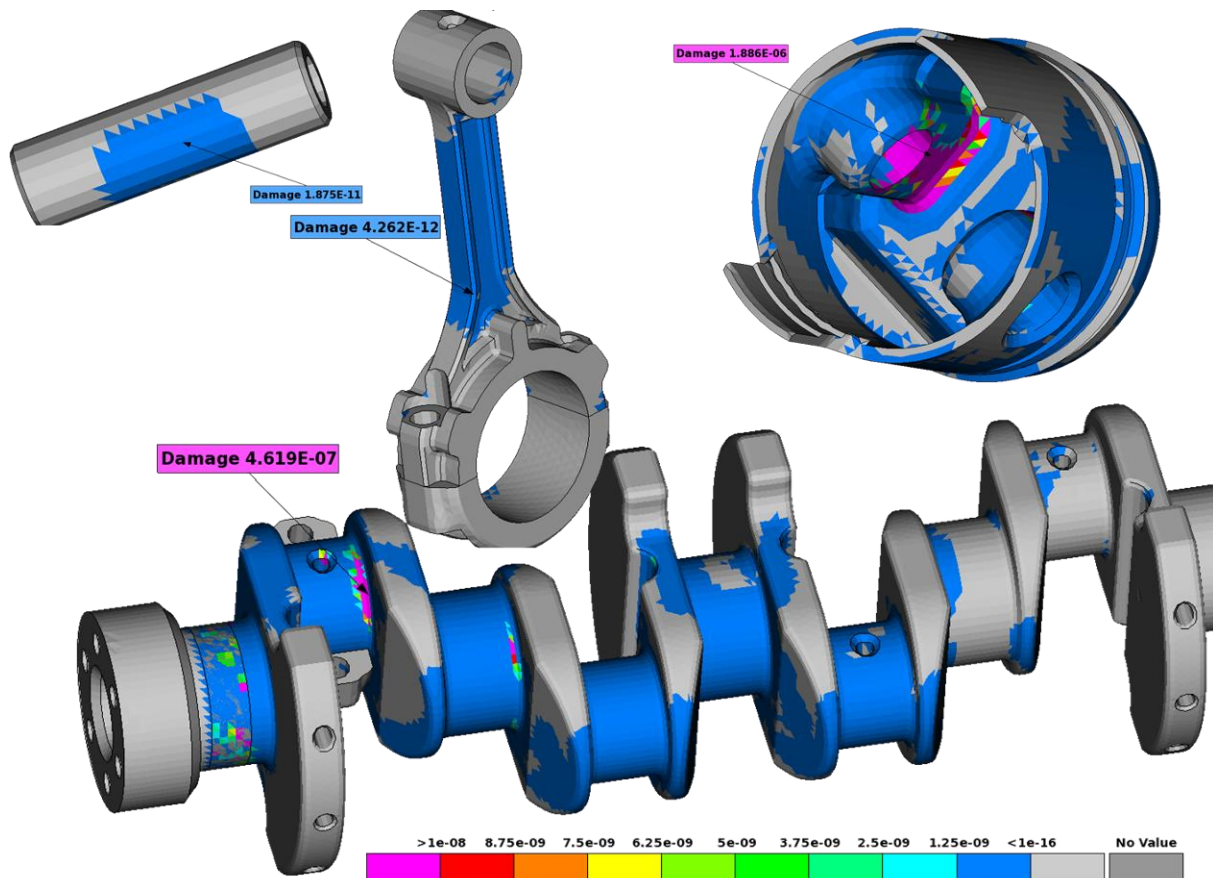


Figure 17 - Fatigue damage results for the crankshaft assembly.

6. CONCLUSIONS

An efficient and user friendly tool for fatigue post-processing within the framework of μ ETA-Post was implemented. Both Abaqus and LS-DYNA simulation results can be handled. The tool was developed using the built-in scripting tools of μ ETA-Post, which with its online help and powerful script editor offers an efficient programming environment. The numerical examples show that fatigue analyses of quite big models can be performed within reasonable time.

Currently, a rather basic fatigue model is implemented, but in the future this could be extended to account for mean stresses, according to for example Goodman or Gerber. Also, extensions towards accounting for multi-axial stress state by use of a more advanced fatigue model (for example the Dang-Van (13) criterion) could be worth considering.

REFERENCES

- (1) μ ETA PostProcessor v15.1.X User Guide (META_Post_V15.2.3_Users_Guide.pdf), help file as delivered with the BETA CAE Software package, BETA CAE, 2015.
- (2) μ ETA PostProcessor v14.1.0 Automation (META AUTOMATION Users GUIDE), help file as delivered with the BETA CAE Software package, BETA CAE, 2015.
- (3) Python 3.3.6 documentation, internet source: <https://docs.python.org/3.3/>
- (4) LS-DYNA Keyword User's manual version 971, Volume I – II, Livermore Software Technology Corporation (LSTC), Livermore CA, 2014.
- (5) Abaqus Analysis User's Manual, Dassault Systèmes Simulia Corp., Providence, RI, USA, 2011.
- (6) Socie, D. F., and Marquis, G. B., Multiaxial fatigue, SAE International, Warrendale 2000.
- (7) Suresh, S., Fatigue of materials, Cambridge University Press, Cambridge 1998.

- (8) Matsuishi, M., and Endo, T., Fatigue of metals subjected to varying stress, *Japan Soc. Mech Engineering*, 1968.
- (9) Niesłony A., Determination of fragments of multiaxial service loading strongly influencing the fatigue of machine components, *Mechanical Systems and Signal Processing*, Vol. 23(8), 2009, pp. 2712-2721.
- (10) Software package rainflow.zip, internet source:
<http://www.mathworks.com/matlabcentral/fileexchange/3026>
- (11) Matlab/Octave script rainflow_bins.m, internet source
http://www.vibrationdata.com/matlab2/rainflow_bins.m
- (12) ASTM E 1049-85 (Reapproved 1997), Standard practices for cycle counting in fatigue analysis, in: *Annual Book of ASTM Standards*, Vol. 03.01, Philadelphia 1999, pp. 710-718.
- (13) Dang Van, K., et al., Criterion for high cycle fatigue under multiaxial loading, *Biaxial and multiaxial fatigue*, editors Brown, M.W. and Miller, K. J., pp 459 – 478, Mechanical Engineering Publications, London 1989,