Fatigue analysis - system parameters optimization

Hanna Markgren (hama0085@student.umu.se)

September 4, 2018

Abstract
For a mechanical system exposed to repeated cyclic loads fatigue is one of the most common reasons for the system to fail. However, fatigue failure calculations are not that well developed. Often when fatigue calculations are made they are done with standard loads and simplified cases.

The fatigue life is the time from start of use until the system fails due to fatigue and there does exist some building blocks to calculate the fatigue life. The aim for this project was to put these building blocks together in a workflow that can be used for calculations of the fatigue life.

The workflow was built so that it should be easy to follow for any type of mechanical system. The start of the workflow is the load history of the system. This is then converted into a stress history that is used for the calculations of the fatigue life. Finally, the workflow was tested with two test cases to see if it was possible to use.

In Algoryx Momentum the model for each case was set up and then the load history was extracted for each time step during the simulation. To convert the load history to stress history FEM calculations was needed, this was however not a part of this project so the constants to convert loads to stress was given. Then with the stress history in place it was possible to calculate the fatigue life.

The results from both test cases were that it was possible to follow every step of the workflow and by this use the workflow to calculate the fatigue life. The second test also showed that with an optimization the system was improved and this resulted in a longer lifetime.

To conclude the workflow seems to work as expected and is quite easy to follow. The result given by using the workflow shows the fatigue life, which was the target for the project. However, to be able to evaluate the workflow fully and understand how well the results can be trusted a comparison with empirical data would be needed. Still, the results from the tests are that the workflow seem to give reasonable results when calculating fatigue life.

Advisor at Algoryx: Peter Norlindh
peter.norlindh@algoryx.se
Examiner: Martin Servin
martin.servin@umea.se

Examensarbete Civileingenjör Teknisk Fysik, 30 hp
ACKNOWLEDGMENT

This project was completed at Algoryx Simulations AB as a thesis for the degree of master of science in engineering physics. First I want to thank my advisor Peter Norlindh at Algoryx who came with the idea for the workflow. He was also available to help me whenever I ran into problems during my work.

I would also like to thank my supervisor Martin Servin at Umeå University. He has helped with this thesis, and I am very thankful for his valuable comments on the report.

Finally, I would also want to thank my family who has supported me during this project. They have also supported me and helped me throughout the education.
Contents

1 Introduction 1

2 Theory 3
  2.1 Stress 3
  2.2 Fatigue theory 3
    2.2.1 S-N curve 3
    2.2.2 Basquin relation 4
    2.2.3 Palmgren-Miner cumulative damage rule 4
    2.2.4 Rainflow counting 4
    2.2.5 Hot Spot Method 5

3 Method 7
  3.1 SpaceClaim 7
    3.1.1 Algoryx Momentum 7
  3.2 WAFO 7

4 Fatigue analysis Workflow 8
  4.1 Assumptions 8
  4.2 Workflow to calculate fatigue 8
  4.3 Use of the Workflow 9
    4.3.1 Create model and extract loads 9
    4.3.2 Create submodel and FEM 10
    4.3.3 Load history to stress history 10
    4.3.4 Rainflow counting 11
    4.3.5 Basquin relation, Palmgren-Miner and Fatigue life 11

5 Test cases 12
  5.1 Case 1 12
  5.2 Case 2 14

6 Results 16
  6.1 Case 1 16
  6.2 Case 2 16

7 Discussion 18
  7.1 Case 1 18
  7.2 Case 2 18
  7.3 Conclusions and continued work 19

References 20
1 Introduction

One of the most common reasons for mechanical failure is due to fatigue. Despite this fact there is no simple workflow to use for fatigue calculations. The methods commonly used for fatigue calculations are often based on estimations. The aim in this project is to create a simple workflow for fatigue calculations. The workflow is designed to be used for fatigue calculations where there are many repeated loads acting on the system. The workflow should be easy to understand and implement and should also combine the methods for analysis that are used today.

Today fatigue analysis is often made with a combination of Finite element methods, FEM, and calculated, empirical or standard loads. The different cases are often simplified which can lead to large errors in the analysis. Due to these reasons there is a great need of a more accurate and more practical method for fatigue analysis.

Currently there does not exist a simple workflow for fatigue calculations and the main task for this project is to create a workflow for this. Since there exist a number of different methods for fatigue calculations the main idea is to combine a few of them to get a workflow that is easy to understand and follow. The main steps of the workflow is to convert load history to stress history and then to use the stress history to calculate the fatigue life.

There is a linear relation between load and stress. With a stress history it is quite easy to calculate the fatigue life. So if a history of loads is known it is possible to calculate the fatigue life time. This project was limited to create the workflow and extract load history, not to create a conversion of load history to stress history. Therefore the constants used to convert load to stress was given from another project. The stress constants had been calculated with FEM.

The most common way to calculate the fatigue life is to make use of so called standard loads. These standard loads are not available for every single case so to make calculations the case is matched to the most similar standard model. Then a S-N plot showing stress amplitude versus number of cycles to failure, for that standard case is used to retrieve parameters to use in the fatigue calculations.

It can be difficult to match the case with a standard model. This might lead to an inaccurate result in the fatigue calculations. This is the reasoning behind the decision in this project to simulate the whole model and that a submodel is chosen to calculate the loads and the fatigue life. The difference compared to using standard model method is that the real submodel is used and not a matched model. A sub model is chosen to shorten time since using the whole model for the calculation of fatigue life would take much longer time. Often it is possible to know where the model has its weak spots. For example in a model where there are welded objects there are often weak spots close to and at the weld.

The fatigue life varies exponentially with the stress-level. So the lifetime is very sensitive to the stress. With the standard method the data are taken from tables and it is hard to get accurate data. This leads to a large source of error and might be the reason why fatigue is the most common reason to mechanical failure. With the workflow
created in this project the aim is to get more accurate result. This is done by calculating the fatigue life instead of estimate it.
2 Theory

2.1 Stress

Stress is the force per unit area and to describe the stress in a single point the stress tensor can be used. The stress tensor is given by

\[
S = \begin{bmatrix}
\sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\
\sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\
\sigma_{zx} & \sigma_{zy} & \sigma_{zz}
\end{bmatrix}.
\] (1)

\(\sigma_{ij} = \sigma_{ji}\) for \(i, j = x, y\) or \(z\) and hence the stress tensor is symmetric.

The stress on a plane that is perpendicular to the \(x_i\) coordinate and in the direction of the \(x_j\) coordinate is represented by the \(\sigma_{ij}\) component[1].

The mean stress, \(\sigma_m\), for a stress cycle is calculated as

\[
\sigma_m = \frac{\sigma_{\text{max}} + \sigma_{\text{min}}}{2}
\] (2)

where \(\sigma_{\text{max}}\) and \(\sigma_{\text{min}}\) are the max and min values in the stress cycle. The stress amplitude, \(\sigma_a\)

\[
\sigma_a = \frac{\sigma_{\text{max}} - \sigma_{\text{min}}}{2}
\] (3)

2.2 Fatigue theory

Repeated load on a material can cause cracks to appear. This can happen on goods, welds, screws and so on. These cracks often leads to sudden and serious mechanical failures. This phenomenon is called fatigue and it is one of the most common reasons for mechanical failure. When and where the material fails depends on the material and also the load history. The life time until something breaks due to fatigue is called the fatigue life.

As mentioned fatigue is a very common reason for mechanical failure. Often when fatigue calculations are done they are made with so called standard loads. These standard loads are not available for every single case so to make calculations the case is matched to the most similar standard model. Then a S-N plot, stress amplitude versus number of cycles to failure, for that standard case is used to retrieve parameters to use in the fatigue calculations[2].

2.2.1 S-N curve

To describe fatigue in a material S-N curves often are used. They are also known as Wöhler curves and is a graph with stress amplitude versus cycles to fatigue failure, N. These curves are generated with samples of different material since different materials have different curves. If the stress amplitude for a material with known S-N curve is known it is possible to extract how many cycles the material would survive before fatigue failure from the curve.
2.2.2 Basquin relation

For many materials a linear relationship between the stress amplitude and number of cycles can be seen using a log-log plot. This relation is described by the Basquin relation and for zero mean stress it is

\[ \sigma_a = \sigma_f' (2N_f)^b \]  

(4)

where \( \sigma_a \) is the stress amplitude, \( \sigma_f' \) is the fatigue strength coefficient, \( N_f \) is number of cycles to failure and \( b \) is Basquin’s exponent[3]. Eq. (4) can be rewritten for non-zero mean stress to

\[ \sigma_a = (\sigma_f' - \sigma_m) (2N_f)^b \]  

(5)

where \( \sigma_m \) is the mean stress[3]. If the mean and amplitude stresses are known it is possible to rewrite eq. (5) to be able to calculate \( N_f \). This leads to

\[ N_f = \frac{1}{2} \left( \frac{\sigma_a}{\sigma_f' - \sigma_m} \right)^{1/b} \]  

(6)

2.2.3 Palmgren-Miner cumulative damage rule

A stress history can be divided to several block were each block has cycles with the same stress amplitude. These blocks can then be used to calculate the total damage. This is done with the Palmgren-Miner damage rule which is described as

\[ D = \sum_i \frac{n_i}{N_{fi}} = 1 \]  

(7)

where \( D \) is the total damage, \( n_i \) is number of cycles for stress \( i \) and \( N_{fi} \) is number of cycles until failure for stress \( i \). When \( D = 1 \) the "system" breaks[3].

To use this rule three assumptions are made. The first is that the number of stress cycles for each stress amplitude as a percentage of all cycles gives the fraction of damage. The second assumption that is made is that the fatigue life is not affected by the order of the stress blocks. The last assumption is that when the sum of damage reach a critical value failure occurs. The critical value is 1 but is often set lower when calculating fatigue to get a safety marginal.

2.2.4 Rainflow counting

To divide the stress history into blocks with different stress amplitudes rainflow counting can be used[4]. The rainflow counting algorithm works like this:

1. The load history is reduced to a sequence of peaks and valleys.

2. "Turn" the load history so that the time axis points down. See example in fig. 1. The stress history can now be compared to a pagoda roof.

3. For each peak or valley a source of water is imagined. The water is imagined to drip down the end of the roof.
4. Then the number of half-cycles is counted by finding flow terminations. The terminations can be due to either:

- The end of the time is reached
- If the flow from an earlier peak or valley merges into the current flow

5. Calculate the amplitude for each half-cycle. Then count the number of half-cycles and full cycles.

![Rainflow counting example](image.png)

**Figure 1** – Rainflow counting example[5]

To use rainflow counting algorithm two key assumptions are made. The first is that the loads are independent and the second is that the order in which the loads have been applied does not matter[4].

### 2.2.5 Hot Spot Method

There exist different methods to evaluate fatigue and one of the methods is the hot spot method[6]. The point where a crack is expected to start is defined as a hot spot. Usually if there is a weld then the hot spot lies at the weld toe. The stress close to a weld toe is often non-linear due to the geometry of the weld and also dependent of the thickness of the sheet metal. The geometric stress at the hot spot is defined as the hot spot stress, $\sigma_{hs}$. To include the non-linear behavior $\sigma_{hs}$ is calculated by quadratic extrapolation.
If \( t \) is the thickness, in mm, of the sheet metal then the points that is used for the extrapolations is chosen to be 0.4\( t \), 0.9\( t \) and 1.4\( t \) from the weld toe. \( \sigma_{hs} \) can be calculated as

\[
\sigma_{hs} = 2.52\sigma_{0.4t} - 2.24\sigma_{0.9t} + 0.72\sigma_{1.4t},
\]  

(8)

where \( \sigma_{0.4t} \), \( \sigma_{0.9t} \) and \( \sigma_{1.4t} \) is the stresses at the different extrapolation points[6].
3 Method

3.1 SpaceClaim

SpaceClaim is a CAD software developed by ANSYS. It is used to create CAD-models, i.e. 3D models. For this project it was used to sketch the model for calculation of fatigue. In SpaceClaim different modules, so called add-ins, can be added. The program can also be used to read other CAD models[7].

3.1.1 Algoryx Momentum

One of the add-ins is Algoryx Momentum which specializes to bring dynamic simulation into CAD models. The tool is powerful and is developed by Algoryx. To handle the dynamic of multi-body systems it uses AGX Dynamics, which is a physics engine[8].

This add-in was used to add physics to the 3D model and with python scripting forces could be added to the model. It was also possible to obtain load histories in parts that were of interest for the fatigue calculations while running a simulation. Then by changing parameters some optimizing was possible to perform.

3.2 WAFO

For the rainflow counting the open-source python-module WAFO, Wave analysis for fatigue and oceanography, has been used. WAFO is from the beginning a Matlab-toolbox but has been extended to work with python also. The WAFO module has been developed at Lund University and is used to analyze loads and random waves[9].
4 Fatigue analysis Workflow

The main task in this project was to create a workflow to use for fatigue calculations. The workflow should contain the steps of how to use load history and convert it to stress to calculate a fatigue life.

4.1 Assumptions

For this workflow to be used some assumptions has been made. These assumptions comes from limitations in the Basquin equation and also the rainflow counting algorithm. The assumptions that were made are:

1. The number of stress cycles for each stress amplitude as a percentage of all cycles gives the fraction of damage.
2. When the sum of damage reach a critical value failure occurs.
3. The loads are independent.
4. The order in which the loads have been applied does not matter.

These assumptions were made but it was still assumed that the result from Basquins relation, rainflow counting and Palmgren-Miner are correct, even though they may not be completely correct. This since there are no better alternatives and the results should be better than from estimations. So to assume that they work seemed most reasonable.

4.2 Workflow to calculate fatigue

The workflow to calculate fatigue for a case that was developed and used for this project consisted of the following steps:

1. Create model and extract load history for the case with a simulation.
2. Choose and create submodel to use with with FEM.
3. Find the linear relation between stress and loads with FEM.
4. Convert load history to stress history with stress constants from the linear relation.
5. Use Rainflow counting algorithm to obtain stress amplitudes, mean stress and number of cycles.
6. Add material parameters $\sigma_f$ and $b$.
7. Use Basquins relation, eq. (6), to calculate fatigue life for each stress amplitude.
8. Use Palmgren-Miner rule, eq. (7), to calculate damage per cycle.
9. Calculate total fatigue life.

To use the workflow a program to simulate in is needed and also some program to calculate FEM with.
4.3 Use of the Workflow

The workflow was tested with two different cases and the steps are described below.

4.3.1 Create model and extract loads

The first step of the workflow was done with SpaceClaim and Momentum with python scripting. The model to calculate fatigue was chosen and either sketched or imported to SpaceClaim. Then joints was added to the model where they were needed to get the wanted behavior in the simulation. The joints were also used to extract the loads to use in the fatigue calculations. After this forces was added to the system with the sequence editor in Momentum.

For the chosen submodel two interfaces were also chosen. The interfaces were chosen at joints in the submodel since the loads could only be extracted at the joints. So these interfaces were used as the source to extract the load histories from.

During the simulation the loads for each time step and for each chosen interface was stored in a text-file. The file that contains the load history was structured as follows; Each row had six columns and the loads was written in the order $F_x$, $F_y$, $F_z$, $M_x$, $M_y$, $M_z$. The first row corresponded to the first time step and the next row to next time step and so on. The time for each step was also saved in another file since the time steps are needed when calculating the fatigue life.

![Figure 2](image)

**Figure 2** – This plot shows an example of a load history.

Fig. 2 shows one example of how a load history extracted from Momentum looks. In appendix A the script used for extracting the load history from Momentum can be seen. The script was used for the second model but is very similar for different cases. The only things that need to be changed are the joints to extract loads from.
4.3.2 Create submodel and FEM

The submodels of the models was chosen where it seemed possible to have large loads and therefore possibly points for the models to fail due to fatigue. Since the FEM calculations and the linear relation between stress and loads was not in focus for this project the stress constants were given.

4.3.3 Load history to stress history

A linear relation between stress and load can be found. Therefore to obtain the stress history from the load history this relation was used:

\[
\begin{bmatrix}
\sigma_{xx} \\
\sigma_{yy} \\
\sigma_{zz} \\
\sigma_{xy} \\
\sigma_{xz} \\
\sigma_{yz}
\end{bmatrix} = \begin{bmatrix}
C_{xxF_x} & C_{xxF_y} & C_{xxF_z} & C_{xxM_x} & C_{xxM_y} & C_{xxM_z} \\
C_{yyF_x} & C_{yyF_y} & C_{yyF_z} & C_{yyM_x} & C_{yyM_y} & C_{yyM_z} \\
C_{zzF_x} & C_{zzF_y} & C_{zzF_z} & C_{zzM_x} & C_{zzM_y} & C_{zzM_z} \\
C_{xyF_x} & C_{xyF_y} & C_{xyF_z} & C_{xyM_x} & C_{xyM_y} & C_{xyM_z} \\
C_{xzF_x} & C_{xzF_y} & C_{xzF_z} & C_{xzM_x} & C_{xzM_y} & C_{xzM_z} \\
C_{yzF_x} & C_{yzF_y} & C_{yzF_z} & C_{yzM_x} & C_{yzM_y} & C_{yzM_z}
\end{bmatrix} \begin{bmatrix}
F_x \\
F_y \\
F_z \\
M_x \\
M_y \\
M_z
\end{bmatrix}, \quad (9)
\]

The left hand side is a vector that contains all the elements in the stress tensor. On the right hand side there is a matrix that contains constants to convert loads to stress and there is also a vector that contains all different loads. \(F_x, F_y, \) and \(F_z\) are the forces and \(M_x, M_y, \) and \(M_z\) are the moments. Stress constants was not a part of this project but the files that was given with them was structured as matrices according to eq. (9). There was a matrix for each node to calculate the stress in. The matrices was lined up after each other row wise. There was one file with stress constants for each interface that loads were extracted from.

In the FEM calculations the weakest spot, the spot of the model where it would most probably break, was determined. Then the stress constants were calculated on an area close to this spot. So the stress that was calculated then was one stress tensor for each node in that area. The stress tensors were calculated for each node in every time step.
Figure 3 - This plot shows an example of a stress history.

In fig. 3 an example of how the stress history looks like can be seen. The stress history is the $\sigma_{xx}$ history from one node. This stress history corresponds to the load history in fig. 2.

An example of the script used for converting load history to stress history and then calculating fatigue life can be seen in appendix B. The script given there is the one that was used with the second model.

4.3.4 Rainflow counting

When the load histories had been converted to stress histories for each node it was possible to use the rainflow counting algorithm which in this project was done in python with WAFO. For every stress history in each node it was possible to obtain stress amplitudes, mean stresses and number of cycles.

4.3.5 Basquin relation, Palmgren-Miner and Fatigue life

The last steps of the workflow were also done in python. The material parameters $\sigma'_f$ and $b$ was added in the calculations to match the material used in the simulation. Since each stress amplitude now was known after the use of rainflow counting it was quite easy to use Basquin’s relation. So for each stress amplitude in all the nodes eq. (6) was used to calculate $N_f$.

When calculating the total fatigue life in one node the damage from all interfaces and also all stress components, $\sigma_{ij}$, was summed. This was due to the fact that all the stress components contributes to fatigue.

After this Palmgren-Miner rule, eq. (7), was used to calculate the total damage for all time steps in the simulation. Then to get the total fatigue life it was just to divide 1 with the damage for one simulation. This led to the fatigue life in number of time cycles.
5 Test cases

5.1 Case 1

The model used in the first test case was a simple T-shaped component made of steel. The main focus for this case was to test the workflow and see if it worked out as expected. The geometry was sketched in SpaceClaim and can be seen in fig. 4. The measurements of the component can be seen in table 1. This case was also done with another model that had the same measurements except that it had rounded edges with a radius of 3 mm on the sub model. The sub model with rounded edges can be seen in fig. 5. However the steps to calculate fatigue was the same for both versions of the component.

Figure 4 – Model without rounded edges that was used in the first case.

Table 1 – This table contains the measurements for the T-shaped component.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>[mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$</td>
<td>100</td>
</tr>
<tr>
<td>$L_2$</td>
<td>29</td>
</tr>
<tr>
<td>$H$</td>
<td>70</td>
</tr>
<tr>
<td>$h$</td>
<td>14</td>
</tr>
<tr>
<td>$t_1$</td>
<td>5</td>
</tr>
<tr>
<td>$t_2$</td>
<td>6</td>
</tr>
<tr>
<td>$d$</td>
<td>5</td>
</tr>
</tbody>
</table>

Extracting values of the loads needed the model to be divided into four bodies. This was because the loads could only be extracted at joints. One body was the sub model which was chosen as the middle part of the component and it can also be seen in fig. 4 and 5 for the two different models. The other three bodies were attached to the sub
model with lock joints. The left joint and the right joint was chosen as the interfaces to extract load data from.

To simulate the T-shaped component and get a load history a force, $F_0$, in the x-direction was added at the center of mass of the left body. The x-direction is pointing to the right in fig.5. The force that was added was

$$F_0 = 5 \times 10^6 \sin(50\pi t) + 3 \times 10^6 \sin(37\pi t)$$

(10)

where $t$ is the time. The load history was extracted from the left and right joints during the simulation and stored in a text-file the time steps were also saved in a text-file. Momentum with python scripting was used to extract the times and loads from the simulation. The material parameters that was used in the simulation and also the fatigue calculations can be found in table 2.

Table 2 – This table contains the material parameters that was used in the simulation and the fatigue calculations with the T-shaped component.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma'_f$</td>
<td>756 MPa</td>
</tr>
<tr>
<td>$b$</td>
<td>-0.13</td>
</tr>
<tr>
<td>$\rho$</td>
<td>7800 kg/m$^3$</td>
</tr>
<tr>
<td>$E$</td>
<td>205 GPa</td>
</tr>
<tr>
<td>$\nu$</td>
<td>0.29</td>
</tr>
</tbody>
</table>

After the simulation was done the fatigue life was calculated with given stress constants, one file for the left interface and another file for the right interface. Then the calculations proceeded according to the steps in the workflow.

The fatigue calculations were a bit different for the two models of the component but the simulation time was 4 seconds for both models. For the component without rounded corners the hot spot method was used. The size of the weld was set to 10 mm. For the component with rounded edges the damage was calculated in all nodes given with the stress constants. Then the node with the shortest fatigue life was used in the result. The nodes used for the calculations were close the rounded edges since that is the most
probable place for the component to break.

5.2 Case 2

For the second test case a crane made of steel that should lift a log into a container, see fig. 6a, was used. The main reason to do this second case was to test the workflow with a model that had more complex geometry, more joints and that was exposed to loads from different sources. This case was used to ensure that the workflow could be used also when the system to be tested is more complex. With this case the workflow was both tested with a more complex model and it was also done some analyzing of the system. The analyzing was done to see how the movement of the crane affected the fatigue life.

The model consisted of three different CAD-models that was put together in SpaceClaim. One of the models was a crane with hydraulics[10]. The second model was a grapple that was used to pick up trees in the simulation[11]. The third model was a rotator that was put between the crane and the grapple to be able to move and rotate the grapple[12].

For the fatigue calculations the sub model that was chosen can be seen in fig. 6b. The analyzing that was done for this test was to change the movement of the grapple of the crane. First the grapple was lowered so it crashed into the ground before picking up the log. Then to extend the lifetime of the crane the grapple was only lowered enough to pick up the log. For this model the grapple did not touch the ground.
To be able to simulate the crane joints were added to the model where they were needed. The joints were added between all parts that should be able to move around each other. The joints that controlled the grapple, the rotator, the hydraulic cylinders and the rotation at the base of the crane motors were added to be able to move the crane in the desired way. Then the sequence editor in Momentum was used to get the wanted movement of the crane for one simulation. So to optimize the system some parameters in the sequence editor was changed so that the grapple did not crash into the ground. For both simulation the time of the simulation was 15 seconds.

The extraction of the loads for the two chosen interfaces was done as in the first case. Then the fatigue calculations was done according to the steps in the workflow. In table 3 the constants that was used for the fatigue calculations and in the simulation can be seen.

Table 3 – This table contains the material parameters that was used in the fatigue calculations with the crane.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma'_f$</td>
<td>168 MPa</td>
</tr>
<tr>
<td>$b$</td>
<td>-0.13</td>
</tr>
<tr>
<td>$\rho$</td>
<td>7870 kg/m$^3$</td>
</tr>
<tr>
<td>$E$</td>
<td>205 GPa</td>
</tr>
<tr>
<td>$\nu$</td>
<td>0.29</td>
</tr>
</tbody>
</table>

The method used for fatigue analysis for both the simulation with the crashing grapple and the more smooth simulation was the hot spot method. The size of the weld for this case was set to 5 mm.
6 Results

For both cases it was possible to go through all the different steps in the workflow and it was possible to get a fatigue life. The desired result of the fatigue life was achieved.

6.1 Case 1

The fatigue life of the T-shaped component was $2.561 \cdot 10^9$ time cycles with the hot spot method and $6.539 \cdot 10^7$ time cycles for the model with rounded edges. One time cycle was 4 seconds so both models had very large fatigue lifes. The points were the two different versions had the shortest fatigue life are shown in fig. 7a and 7b and marked as red circles. Fig. 7a shows the result with the hot spot method and fig. 7b shows the result for the other method.

![Figure 7](image)

(a) The sub model used with the hot spot method is shown here and the node with shortest life time due to fatigue is marked with a red circle. (b) The sub model with rounded edges is shown here and the node with shortest life time due to fatigue is marked with a red circle.

6.2 Case 2

The result of this case with the hot spot method was that with the smooth simulation the crane would survive around 4000 time cycles. One time cycle was one lift with the crane so it would survive 4000 lifts with the same cyclic loads. For the simulation with the grapple that crashed into the ground before lifting it would survive around 760 lifts.
Figure 8 – The node with the shortest fatigue life on the crane sub model is marked with a red circle.

The node where the fatigue life was shortest is shown in fig. 8 with a red circle. The node was the same for both the smooth simulation and the simulation with the crashing grapple.
7 Discussion

The main task for this project was to create a workflow to be used for calculations of fatigue life. The outcome shows that it was possible to use the proposed workflow in both cases where it was applied. Of course the lack of analytic data is a problem in the evaluation of the workflow, there may be effects or factors that have not been taken into account that would impact the results if tested on a physical model. However, the results from the test cases shows results that are reasonable and corresponds to the expectations which makes it possible to conclude that the model is working based on the assumptions made. In addition the model has been easy to follow which also was a part of the purpose of the project. With these results the work can continue to improve and refine the model.

Looking into the assumptions that were made for the different parts in the workflow this is, as mentioned, something that could have affected the results. To start with the order in which the loads have been applied is something that could affect the result. The reason for this would be that if there is a lot of large loads after each other the material would probably be worn out earlier. It is likely that larger loads cause more damage to the construction and if these heavier loads are applied closer in time than in a fully randomized order the damage would become apparent in an earlier stage.

Another thing that also made it a bit hard especially in the beginning to get results was that the python scripting in Momentum is still developing. So I encountered some problems with it. But in the end I had a version that was possible to work with and that I could extract the values that I wanted.

Since the fatigue life is calculated with the workflow and not estimated as with the standard model the result should be more accurate. The fatigue life is exponentially dependent on the stress and therefore very sensitive to small errors in the stress so the result should be more accurate with the workflow than with estimations of the stress.

It was assumed that the assumptions needed for the different building blocks did not affect the results. But since there does not exist any alternatives to use in the workflow this method should be the best solution. With the standard method the values was estimated but with this workflow the stress is actually calculated. So the result should at least be better than with the standard method.

7.1 Case 1

For the T-shaped component there is not much to say about the result since I do not have anything to compare with. The nodes where the component breaks for both methods also seems reliable since the thickness is smaller on the top part, $t_1$, than on the lower part, $t_2$.

7.2 Case 2

Since the simulations were done with a test case that was made up it is hard to know if the results are reliable. I do not have any analytic data that I can compare the results with. However I think it feels quite good to get a result were the crane would survive
more than one lift. Because I had a lot of problem with load histories that had large peaks when some force was added. But in the end I was able to reduce those peaks by adjusting the min and max values of the force in the joints. This made the motors to accelerate over a few time steps to the given value instead of going to the given value in one time step.

The result that the crashing simulation survives for a shorter time than the more smooth simulation also seems reasonable. I think this is reasonable since the loads should be larger for the case when the grapple crashes into the ground than the smooth lifting.

7.3 Conclusions and continued work

To conclude the workflow is possible to use for fatigue calculations. It is easy to follow and gives the expected result. However since I do not have any empirical data to compare the results with it is hard to say how accurate the results are. But one of the advantages for this method compared to the standard loads is that the stress is calculated and not estimated. This would lead to more accurate results.

If this project were to be continued the next step with this workflow would be to test it with a case where there exist empirical data of the stress or load history. This to see how accurate the calculations really are. After this, if needed, some adjustments in the workflow could be made to get even more accurate results.
References


To extract the load histories from Momentum python scripting was used. Below is the script that was used with the crane. The script for the T-shaped component was very similar.

```python
import v1
import numpy as np

# Access the simulation using:
sim = v1.Simulation()

fir_jo = sim.getJoint("end")  # First joint to extract loads from
sec_jo = sim.getJoint("basecyl")  # Second joint to extract loads from

i = 0
tlength = 15
definite = 1000
tsteps = tlength*freq
times = np.zeros((tsteps, 1))
load_1 = np.zeros((tsteps, 6))
load_2 = np.zeros((tsteps, 6))

def OnStart(time):
    pass

def OnStep(time):
    global i

    # Append values to lists
    times[i] = time
    load_1[i, 0:3] = [fir_jo.getBodyForce(1).x(), fir_jo.getBodyForce(1).y(),
                      fir_jo.getBodyForce(1).z()]
    load_1[i, 3:6] = [fir_jo.getBodyTorque(1).x(), fir_jo.getBodyTorque(1).y(),
                      fir_jo.getBodyTorque(1).z()]
    load_2[i, 0:3] = [sec_jo.getBodyForce(1).x(), sec_jo.getBodyForce(1).y(),
                      sec_jo.getBodyForce(1).z()]
    load_2[i, 3:6] = [sec_jo.getBodyTorque(1).x(), sec_jo.getBodyTorque(1).y(),
                      sec_jo.getBodyTorque(1).z()]

    i += 1

A.I
def OnStop(time):
    np.savetxt('C:/Users/algoryx/Documents/Momentum/crane_load1.txt', load_1)
    np.savetxt('C:/Users/algoryx/Documents/Momentum/crane_times.txt', times)
    np.savetxt('C:/Users/algoryx/Documents/Momentum/crane_load2.txt', load_2)
Appendix

The fatigue life calculations was done with python. For the crane the hotspot method was used and below is the code that was used for these calculations. It consists of three parts. The first parts loops over the points to calculate the hot spot stress in.

```python
from hotspot import *

# This file is used to calculate the fatigue life for a crane with the hotspot method

# Read data from files
times = np.loadtxt('crane_times.txt')  # Time steps
loads_l = np.loadtxt('crane_load1.txt')  # Forces and torque at left interface
loads_r = np.loadtxt('crane_load2.txt')  # Forces and torque at right interface
nodes = np.loadtxt('nodes_hs.txt')
no_p = 30  # Number of points to calculate the hot spot stress in
x = np.array([0.23452])
dtheta = 2*np.pi/(no_p-1)
rd = np.array([0.036676, 0.040646, 0.044616])  # 5 mm weld

for j in range(no_p):
    theta = j * dtheta
    y = np.array([2.14794+rd[0]*np.sin(theta), 2.14794+rd[1]*np.sin(theta),
                  2.14794+rd[2]*np.sin(theta)])
    z = np.array([-2.29742+rd[0]*np.cos(theta), -2.29742+rd[1]*np.cos(theta),
                  -2.29742+rd[2]*np.cos(theta)])
    tf_h[j, :] = hotspot_method_crane(times, loads_l, loads_r, nodes, x, y, z, 'C_hs.txt')

m = tf_h[:, 0].argmin()
print("Estimated lifetime (in cycles):", tf_h[:, 0].min())
print("Coordinates: ", nodes[int(tf_h[m, 1])], nodes[int(tf_h[m, 2])],
       nodes[int(tf_h[m, 3])])
```

B.III
The second part, which can be seen below, was used to calculate the hot spot stress.

```python
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~
from fatigue/rfc import *
# ~~~~~~~~~~~~~~~~~~~~~~~~~~

def hotspot_method_crane(time, load_l, load_r, nodes, x, y, z, C_file):
    
    """
    Fatigue Life estimation with rainflow and hot-spot method
    """
    Parameters
    ----------
    time : array with time steps
    load_l : matrix with force and torque history on the left interface,
        each line should contain [Fx Fy Fz Mx My Mz]
    load_r : matrix with force and torque history on the left interface,
        each line should contain [Fx Fy Fz Mx My Mz]
    nodes : matrix with coordinates of the nodes
    x : array with x-value for the three points used with hotspot
    y : array with y-values for the three points used with hotspot
    z : array with z-values for the three points
    C_file : name of the file that contains the constants for the weld
    Returns
    -------
    tf: array containing fatigue lifes and nodes for extrapolation
        """
    tf = np.zeros((1, 4))
    a = np.array([x[0], y[0], z[0]])
    b = np.array([x[0], y[1], z[1]])
    c = np.array([x[0], y[2], z[2]])

    # Find nodes closest to the extrapolation points
    n04 = np.sum(np.abs(nodes - a), axis=1).argmin()
    n09 = np.sum(np.abs(nodes - b), axis=1).argmin()
    n14 = np.sum(np.abs(nodes - c), axis=1).argmin()

    sigma04 = load_to_stress_node(load_l, load_r, C_file, n04)
    sigma09 = load_to_stress_node(load_l, load_r, C_file, n09)
    sigma14 = load_to_stress_node(load_l, load_r, C_file, n14)

    sigma_hs = 2.52 * sigma04 - 2.24 * sigma09 + 0.72 * sigma14
    tf[0, 0] = fat_rfc(sigma_hs, time)
    tf[0, 1] = n04
```

B.IV
tf[0, 2] = n09
tf[0, 3] = n14

return tf

The last part was used to calculate fatigue life and this function was used for both the crane and the T-shaped component.

```python
import matplotlib.pyplot as plt
import numpy as np
import wafo.objects as wo

def fat_rfc(sig_nom, time, sig_fail=33.096E6, b_const=-0.13):
    
    """
    Fatigue Life estimation with rainflow counting algorithm
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Parameters
    ----------
    sig_nom : array with stress history in all directions
              (one column for each direction)
              ((in the order: Sxx Syy Szz Sxy Sxz Syz))
    time : array that contains the time steps
    sig_fail : scalar, optional
               fatigue strength, material parameter.
    b_const : scalar, optional
               Basquin coefficient, material parameter.
    Returns
    -------
    tf : Estimated life time in cycles
    """

    no_dir = sig_nom.shape[1]
    D = 0.0
    for j in range(no_dir):
        # Add time and sig_nom to a matrix
        sig_mat = np.column_stack((time, sig_nom[:, j]))
        # Extract Time series and turning points from data
        ts1 = wo.mat2timeseries(sig_mat)

    """
    Turning points for rainflow-filtered data
    B.V
```
try:
    sig_tp = ts1.turning_points(h=-1, wavetype='astm')
except ValueError:
    sig_tp = None

if sig_cp is not None:
    try:
        sig_cp = sig_tp.cycle_astm()
    except AttributeError:
        print('Could NOT use cycle_astm')
        sig_cp = None

# Damage calculations
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Calculate damage with:
# Basquins equation - \( \sigma_a = (\sigma_f - \sigma_m) \times (2N_f)^b \)
# where \( \sigma_a \) = stress amplitude, \( \sigma_f \) = fatigue strength(mat. prop.)
# \( \sigma_m \) = mean stress, \( N_f \) = Fatigue number
# \( b \) = Basquins coefficient (material property)
# \( => N_f = 0.5 \times (\sigma_a / (\sigma_f - \sigma_m))^{1/b} \)
# With Basquins equation combined with P-M we get:
# \[ D = \text{sum} \left( \frac{n_i}{0.5 \left( \frac{\text{sig}_a}{(\text{sig}_f' - \text{sig}_m)} \right)^{1/b}} \right) \]  
# \[ \Rightarrow D = 2 \times \text{sum} \left( \frac{n_i}{\left( \frac{\text{sig}_a}{(\text{sig}_f' - \text{sig}_m)} \right)^{1/b}} \right) \]  
# Where \( D \) in this case is the total damage for this time period,  
# To get the damage per second we divide \( D \) with the length of  
# the time period then to get the time for failure we divide \( 1 \)  
# with the damage per second  

```
bc = 1.0/b_const
N_f = 0.5*np.power(sig_cp[:, 0]/(sig_fail-sig_cp[:, 1]), bc)
for k in range(len(N_f)):
    if np.isnan(N_f[k]):
        N_f[k] = 1.0/np.power(10, 10)
D += np.sum(sig_cp[:, 2]/N_f)

tf = 1.0/D
return tf
```