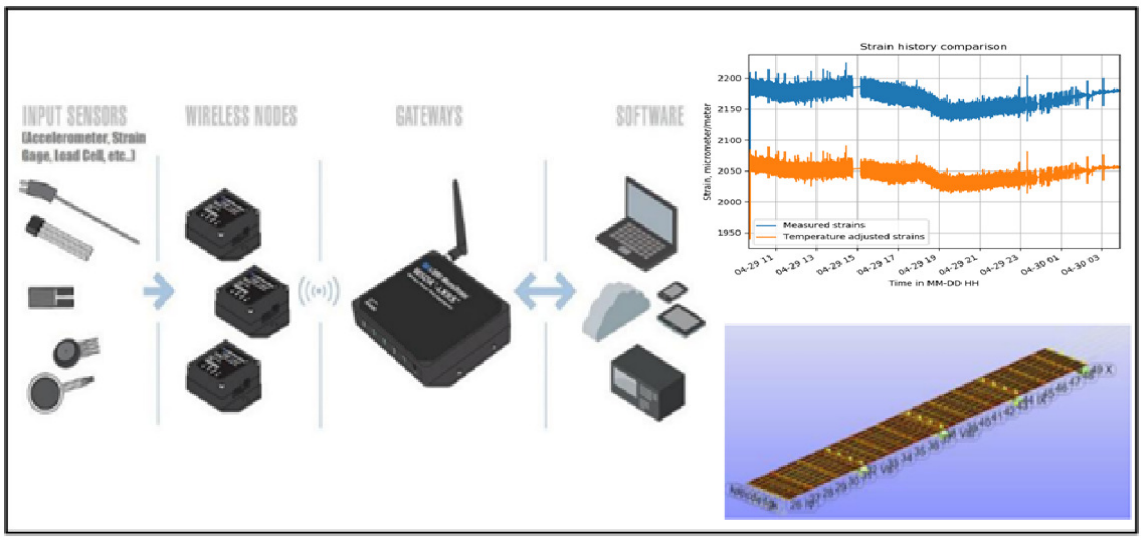
CHALMERS
UNIVERSITY OF TECHNOLOGY



Improved fatigue evaluation of Göta Älv bridge using monitoring data

Master's thesis in Structural Engineering and Building Technology

MAREK KORZENIOWSKI
SIMON LARSSON

Department of Architecture and Civil Engineering
Division of Structural engineering
Steel and Timber Structures
CHALMERS UNIVERSITY OF TECHNOLOGY
Master's Thesis ACEX30-18-13
Gothenburg, Sweden 2018

MASTER'S THESIS ACEX30-18-13

Improved fatigue evaluation of Göta Älv bridge using monitoring data

Master's Thesis in the Master's Programme Structural Engineering and Building Technology

MAREK KORZENIOWSKI
SIMON LARSSON

Department of Architecture and Civil Engineering
Division of Structural engineering
Steel and Timber Structures
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2018

Improved fatigue evaluation of Göta Älv bridge using monitoring data

Master's Thesis in the Master's Structural Engineering and Building Technology

MAREK KORZENIOWSKI

SIMON LARSSON

Examensarbete ACEX30-18-13
Institutionen för bygg- och miljöteknik,
Chalmers tekniska högskola 2018

Department of Architecture and Civil Engineering
Division of Structural engineering
Steel and Timber Structures
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Cover:
Figure presenting various stages in the use of a monitoring system. Left: collection and transfer of measurement data using a wireless sensor network; Upper-right: recorded strain response on the lower flange of a main girder in the bridge; Lower right: finite element model of a part of the bridge.

Department of Architecture and Civil Engineering
Göteborg, Sweden, 2018

Improved fatigue evaluation of Göta Älv bridge using monitoring data.

Master's thesis in the Master's Program Structural Engineering and Building Technology

MAREK KORZENIOWSKI
SIMON LARSSON
Department of Architecture and Civil Engineering
Division of Structural engineering
Steel and Timber Structures
Chalmers University of Technology

ABSTRACT

The Göta Älv Bridge was constructed during the initiation of the welding era and was completed in the year 1939. During its service life, the bridge has endured a substantial amount of traffic load, which have resulted in fatigue induced cracks in some locations. The cracks have prompted the installation of a monitoring system, which in part consists of strain gauges. Due to practical reasons, some of the strain gages could not be located close to the critical locations. In this study, a fatigue evaluation of the Göta Älv bridge is carried out using the data collected by the strain gauges, as well as historical data. This is done in three main steps.

The first step is the calibration of the FE-model. The calibration's goal is fine-tuning the model so that the computed stresses match those collected at the bridge. The calibration is done both by manual adjustment of model parameters and by an automated script.

The second step is the development of a method to transform the stresses collected at one location to another. This is needed to solve the problem that the strain gauge could not be placed at the critical location. The Spatial Adjustment Factor (SAF) concept is used based on the original idea given by Liu (2010), but with substantial modifications to account for the effect of moving traffic loads. The method is based on the calculation of a quotient between the equivalent stresses for the two locations. The SAF can then be used to transform the stress spectrums to the desired location. The SAF concept is evaluated for a number of differently spanned beams and different load cases. In general, the method yields better results the more similar the influence lines at the two locations are to each other.

The final step is the fatigue assessment, in which two details are evaluated. For the evaluation, both historical (past) traffic data and measured data are used. For the fatigue damage due to past traffic, traffic data in terms of vehicle's weight and a number of passages are used in a conventional cumulative damage calculation as in EN 1993. For the damage evaluation due to measured stresses, the recorded data is transformed to stress spectrums and equivalent stresses by filtering and processing in a computer script which also utilizes the SAFs to evaluate the equivalent stresses in the critical details.

The report describes a method by which the SAF can be computed and used for the transformation of stresses. It also gives some suggestions for further improvements to the concept.

Keywords: Fatigue assessment, Structural Health Monitoring (SHM), Finite element model calibration, Spatial Adjustment Factor (SAF), Steel bridges.

Contents

Preface

This master's thesis has been produced as a result of a project conducted at ÅF Infrastructure AB with a collaboration of researchers from the Chalmers University of Technology.
The authors would like to express the greatest thanks, to all of people involved in work with this thesis. Especially thanks are due for supervisor Farshid Zamiri who always granted us kindly with his time and Mohammad Al-Emrani who often provided engaging brainstorming discussions, helping to better understand project's issues.

Göteborg, June 2018
Marek Korzeniowski and Simon Larsson

List of notations

$\Delta\sigma$	Stress range
$\Delta\sigma_C$	Reference range value of the fatigue strength
$\Delta\sigma_{eq}$	Equivalent stress
$\Delta\sigma_{FLM}$	Stress range due to the fatigue load model
ΔK	Intensity factor range
σ_{max}	Maximal stress
σ_{min}	Minimal stress
σ_{nom}	Nominal stress
σ_{mean}	Mean stress range
$\sigma_{amplitude}$	Stress amplitude
γ_{Ff}	Partial safety factor for fatigue loading
γ_{Mf}	Partial safety factor for fatigue strength
Φ_2	Dynamic factor
λ_i	Damage equivalent factor
A_i	Area of a cross-section
a_0	Initial crack size (depth or length)
a_c	Final or critical crack size (depth or length)
C	Constant of the Paris law
D	Damage accumulation factor
K_t	Stress concertation factor
m	Slope of fatigue strength curve
n	Number of loading cycles
N_i	Number of stress cycles within initiation phase
N_p	Number of stress cycles within propagation phase
N_t	Total number of cycles (fatigue life of the detail)
R	Stress ratio
Y	Product of various multipliers which account for the geometry
FLM	Fatigue load model
ULS	Ultimate limit state
SHM	Structural health monitoring

1 Introduction

1.1 Background

Many existing steel bridges, especially those erected during the inception of the welding technique, during the first half of the twentieth century, are aging and often encounter problems of metal fatigue nature. The aging effect takes the form of fatigue process. The bridges that were constructed long time ago have endured many stress cycles which have exerted certain damage to the bridge. The fatigue damage is exacerbated by the fact that the loads acting on the bridges have increased over the bridges' lifespan. Furthermore, many of the old bridges have not been designed with fatigue life in mind, using poor detailing, steel quality, and welding practices. These aspects also make the problem worse with fatigue in existing steel bridges.

The established conventional method for evaluating fatigue relies on simplified models of the structure, combined with simplified but conservative estimations for the loading and structural response. This results in conservative estimations for the structures' lifespan with regard to fatigue. However, due to the manner of how the results are determined, the calculated fatigue life may differ from the structures' actual fatigue life and thus the fatigue life is usually underestimated.

1.2 Motivation

The conservative estimation might unnecessarily recommend the bridge be torn down, whereas a more accurate evaluation of the fatigue life might show that the bridge has sufficient remaining fatigue life. A determination of the fatigue life based on data from the strain measurements would increase the accuracy of the determined fatigue life. The more accurately determined fatigue life would provide a better basis for the actions that should be performed on the bridge. The increased accuracy might thus result in a better allocation of public resources and funds.

1.3 Aim and objectives

The aim of the thesis is to examine the concept of using measured data as the basis for advanced fatigue evaluation of a case study bridge which is the Göta Älv Bridge (Swedish: Götaälvbron) in Gothenburg. The purpose is to evaluate fatigue damage caused to the structure. Another part of this thesis will be a validation of the FE model using data collected at the bridge. An additional purpose of this thesis is to develop and investigate an innovative method, Spatial Adjustment Factor (SAF), which would allow for the transformation of stress measurements from one location to another.

1.4 Method

A Wireless Sensor Network (WSN) is installed on the bridge for Structural Health Monitoring (SHM). The data collected by the monitoring system is processed by a series of computer scripts created by the authors. The collected data is used for two purposes: firstly, to calibrate an existing FE model of the bridge; and secondly, to evaluate the fatigue damage caused to the bridge due to road and rail (tramway) traffic. The calibrated model serves for the generation of reliable influence lines, which is later

used for fatigue evaluation of the chosen critical details of the bridge. The fatigue life evaluation is based on the damage accumulation method. Furthermore, for the development and proof of SAF concept, the generated influence lines from the FE model of the Göta Älv Bridge, as well as from simple multi-span beams, are used for finding the correlation between equivalent stresses at two points of interest.

1.5 Limits and scope of the project

This thesis is limited to fatigue evaluation based only on the damage accumulation method using nominal stresses. Other methods used for fatigue assessment (local stresses methods, fracture mechanics) are beyond the scope of this study. Furthermore, the concept of SAF is only checked and proved empirically for simple cases presented in the thesis.

1.6 Introduction to the Göta Älv bridge

The Göta Älv Bridge is located in the center of Gothenburg, it crosses the Göta river (Swedish: Götaälv) and connects the city center of Gothenburg on the south side to the Hisingen island on the north side. The bridge is the only tramway connection between the northern part of the city (Hisingen) and the rest of the city. The bridge was completed in the year 1939 after three years of construction. The bridge is about 950 meters long and can be divided into three parts, the southern "viaduct", the river part and the northern "viaduct", see Figure 1-1 for an illustration. One of the spans in the central part of the bridge is openable (leaf bridge), see Figure 1-2.

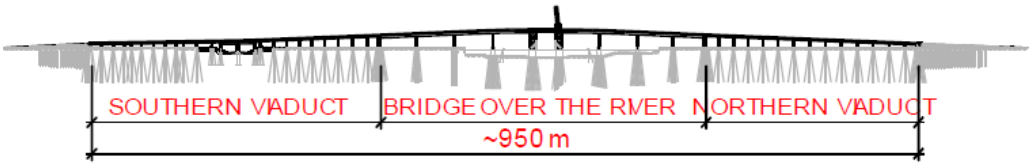


Figure 1-1 Side view of the bridge, Zamiri (2018).

Figure 1-2 Openable part of the bridge, Leander (2015).

Currently, the bridge has six lanes for traffic (Figure 1-3). The two centremost ones are reserved for public transport (tramway and buss) with a restriction of 10 tons per axle weight and 16 tons per bogie. The adjacent lanes are restricted to a maximum axle weight of 8 tons and 12 tons per bogie. The outermost lanes are restricted to vehicle weight of 3,5 tons (Olsson, 2015). Outside of the outermost traffic lanes there exists a pedestrian and bicycle lane, which has its own load carrying system (Leander, Trillkott, & Kullberg, 2015).

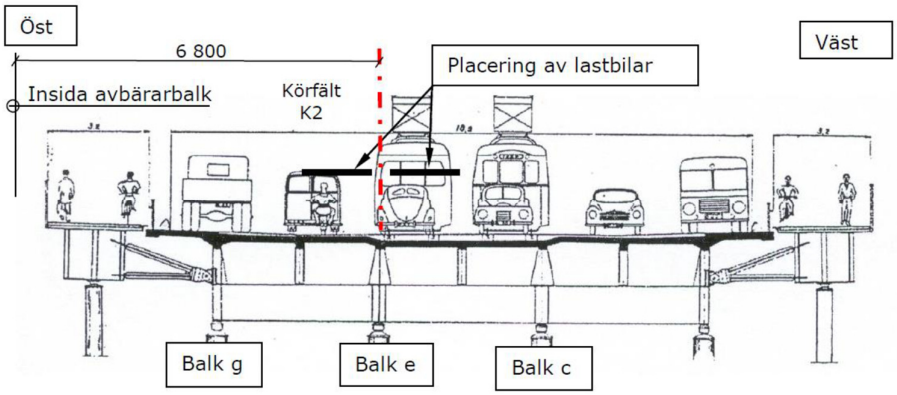


Figure 1-3 Cross-section of the bridge and lanes' division, Olsson (2015).

The bridge is constructed of steel as the main material for the load carrying system, with a concrete slab placed on top (Leander et al., 2015). Shear connectors between the concrete deck and the steel girder exist, but their spacing does not conform to the requirements of modern Eurocodes and it is not clear if the complete composite action exists between steel and concrete. The bridge has seven main longitudinal I-girders. For the viaduct parts, the main longitudinal beams are cut above the support in the place where they are connected to the continuous crossbeams (Figure 1-4). The continuity of the longitudinal beams is provided by welded cover plates to the top and bottom flanges (Figure 1-5). From fatigue point of view, this type of connection is considered a vulnerable detail. This was the case when during regular inspections in 1999, two cracks in the top cover plates were detected in the northern viaduct (Figure 1-6).

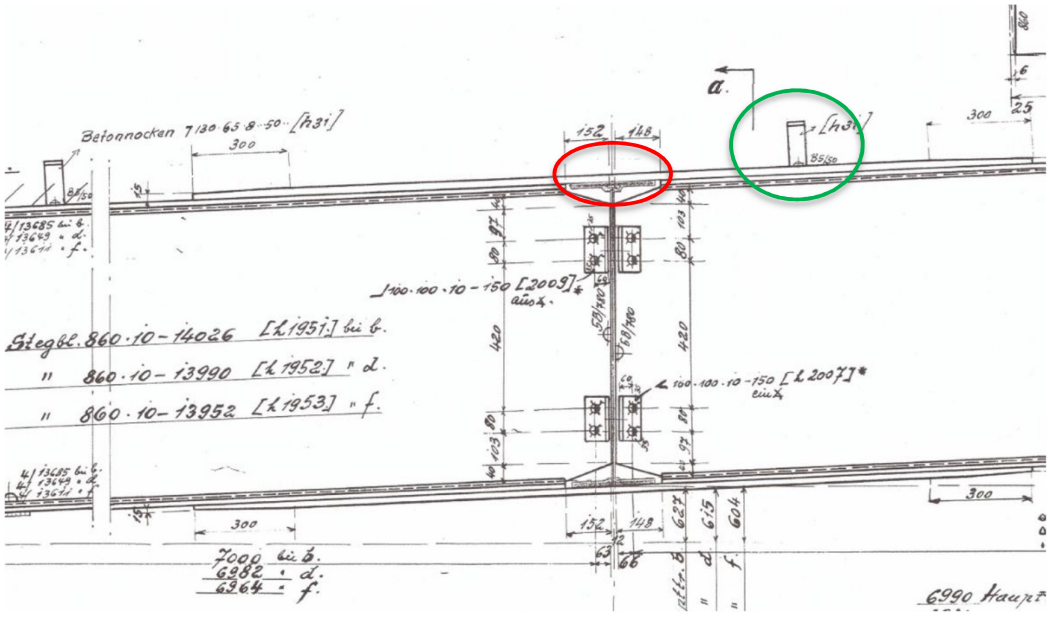


Figure 1-4 *A side view of the connection between the longitudinal beam and the cross-beam. Marked on a green color– shear connector, marked on the red color – critical location, vulnerable to crack, Olsson (2015).*

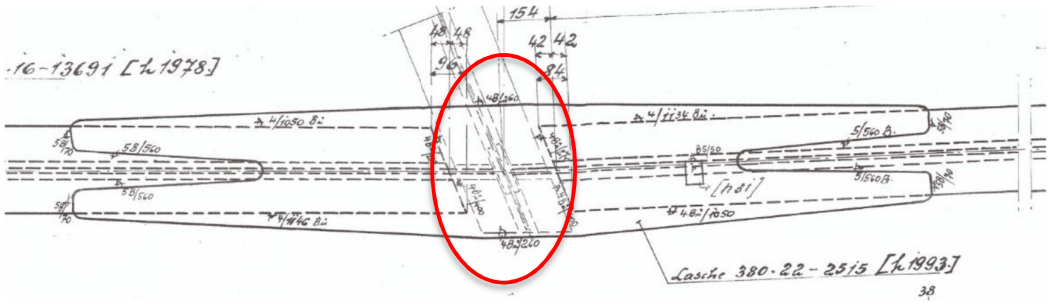


Figure 1-5 *A view from above the connection between the longitudinal beam and cross beam. Marked on a red color – critical location, vulnerable to crack, Olsson (2015).*

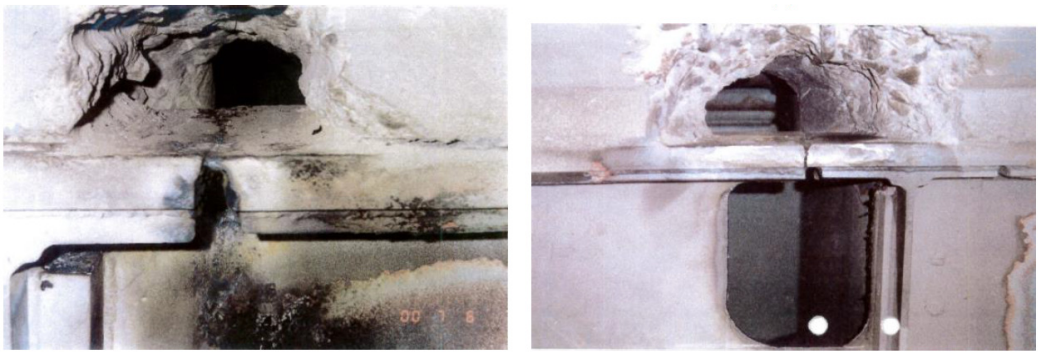


Figure 1-6 Picture of the crack location above the support in the northern viaduct, Zamiri (2018).

In the viaduct parts, the transversal beams have almost the same dimensions as the longitudinal ones. They are located over the support and in the mid-span. The river part of the bridge continues with seven longitudinal I-girders, however, four of them are considerably larger than the remaining three: 2.4 m height for the main girders versus 1.1 m for secondary girders. The main girders are continuous over several spans, whereas the smaller ones are simple beams with riveted connections at the ends. The transversal beams in the river part are located above the support as well as being distributed in the span with a distance of 6 to 7 meters. The transversal beams are smaller than the main girders, with a height of about 1.3 meters. (Olsson, 2015). A general overview of the bridge cross sections in the viaduct part and the river part is presented in Figure 1-7.

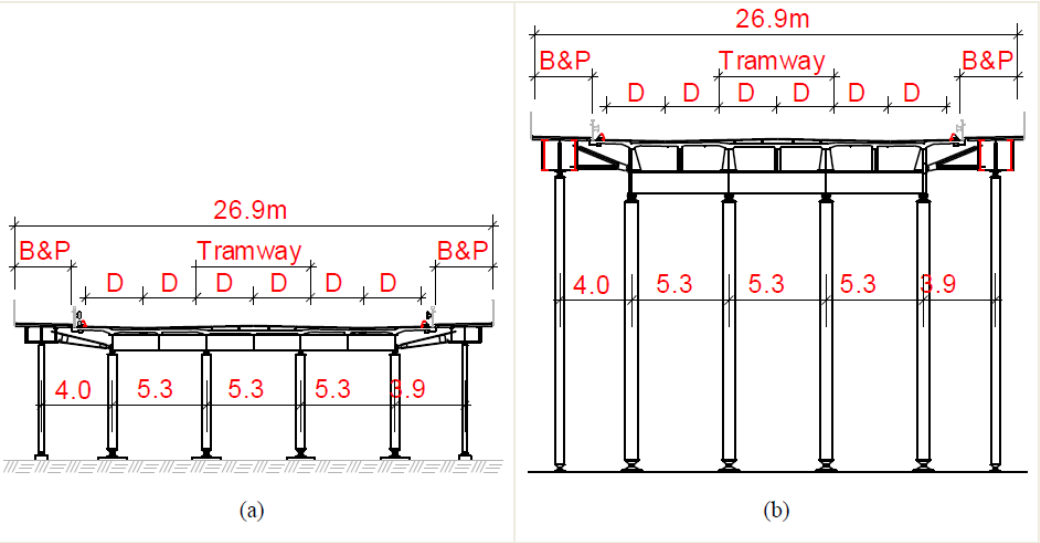


Figure 1-7 Comparison of the viaduct (a) and river part (b), Zamiri (2018).

The concrete slabs estimated strength class is K40 (equivalent to C28/35), with a thickness that varies between 180 and 210 mm. The steel quality used for the production of the girders and crossbeams is St.44 for the viaducts and St.52 for the river part, which today they are represented by S275 and S355 in Eurocode (Olsson, 2015).

2 Theory and literature study

2.1 What is fatigue?

Material fatigue is the process by which damage, and potential failure, occurs as a result of cyclic loading, despite the stresses being lower than the strength of the component under cyclic loading. The low stresses cause small localized damages to accumulate within a limited region of the component, usually one with high local stresses. The small local damages within the region will eventually combine into a larger fatigue crack. As the crack grows, the local stresses will increase at a faster rate, finally resulting in sudden failure (Al-Emrani & Åkesson, 2013).

The fatigue life of a detail within a structure, N_t, consists of two phases, the initiation phase, N_i, when a fatigue crack forms, and the propagation phase, N_p, when the crack is propagating. If a crack already exist due to local defects, then the fatigue process only consists of the propagation phase. The fatigue life can be expressed as:

$$N_t = N_i + N_p \tag{2.1}$$

The initiation phase is characterized by being erratic and slow regarding the formation of the small damages, whereas the propagation phase has the behavior of steady growth. Depending on the detail subjected to the fatigue loading, either of the two phases can be the dominating one. For undisturbed details, without sharp geometrical changes, welds or other stress raisers, the initiation phase is dominant. For disturbed details, the propagation phase is the dominating one (Al-Emrani & Åkesson, 2013).

2.1.1 Fatigue process

Consider a smooth steel bar as illustrated in Figure 2-1-a, which is subjected to axial cyclic loading.

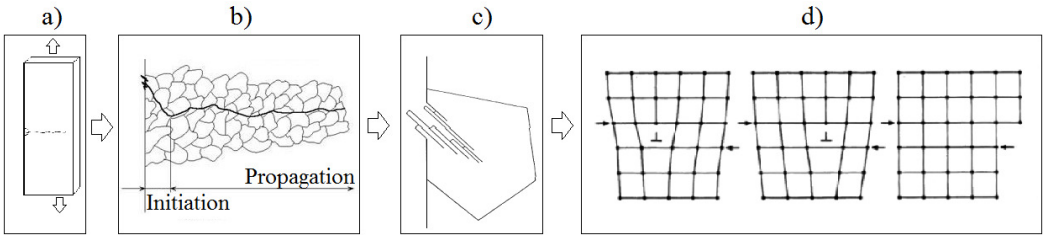


Figure 2-1 The process of fatigue crack growth on four different levels, to illustrate the process; a) the crack growth at the scale of the sample; b) the crack growth orientation in the initiation phase and in the propagation phase; c) the slip band formations due to the cyclic loading at material boundaries; d) the movement of the dislocation within a lattice structure, (Al-Emrani & Åkesson, 2013).

The steel material within this sample is comprised of grains separated by grain boundaries. Within these grains there exist defects such as dislocations in the crystal lattice structure. The grains that are located at the edge of the steel sample, are less restrained from deforming, compared to those further in. When the steel sample is subjected to cyclic loading, the dislocations within the boundary grains are moved

(Figure 2-1-d) and will eventually accrue to a slip band at the free edge (Figure 2-1-c), either as intrusions or extrusions. This process is driven by shear which causes these microcracks to propagate at an angle of 45 degrees. The movement of the dislocations and the slip band are both plastic deformations. During the cyclic loading, several of these microcracks are formed and will eventually connect to form one or more main cracks. When the main crack is formed, it will cause the crack propagation to change direction, being perpendicular to the direction of maximum principal stress. (Al-Emrani & Åkesson, 2013).

2.1.2 Factors that influence the fatigue of steel structures

Several factors affect the fatigue life of the component. These factors can be different depending on the stage in the fatigue life (initiation or propagation). Some of these factors are discussed here.

2.1.2.1 Detail geometry

The effect of geometry is somehow different during the two phases of fatigue damage. During the initiation phase, the geometry of the detail is a determining factor which influences the initiation location of the crack. Assuming a uniform cross-section of a plate subjected to uniform tension, the stresses will be evenly distributed and equal to nominal stresses, σ_{nom}. If a geometrical disturbance, such as a hole, notch, or crack is introduced to the plate, the stress distribution will change across the cross section. The change in stress is defined by the Stress Concentration Factor (SCF) shown by K_t. Around the geometrical disturbance there will be a stress concentration, defined as: $\sigma_{max} = \sigma_{nom} \cdot K_t$. The degree to which this disturbance increases the stresses depends on the geometrical disturbance. In general, a more sudden change of geometry will cause a greater increase of stresses, whereas a smooth geometrical transition will cause a smaller stress increase. E.g. given two different cross sections, $A_1 and\ A_2$, a tapered transition is preferable over a sudden change (Al-Emrani & Åkesson, 2013). This is illustrated in Figure 2-2.

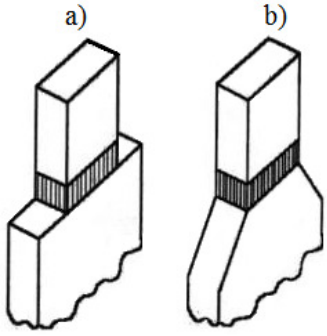


Figure 2-2 Two different geometrical transitions of a welded connection. a) sudden geometrical transition together with a weld, causing a large stress concentration in the transition region; b) Smoother geometrical transition in the welded area, which has a lower stress concentration factor and is beneficial for the fatigue life of the detail, (Al-Emrani & Åkesson, 2013).

During the propagation phase, the crack is already formed and the SCF no longer applies, since the elastic stress at the crack tip is singular (infinite). Another parameter, Stress Intensity Factor (SIF) is used in this phase to describe the stress state at the crack tip region. SIF is not only a function of the loading mode, the detail's geometry, and the boundary conditions, but also it is a function of the crack shape and crack's size. So, for the propagation phase, both the detail's geometry and crack geometry become influencing factors.

2.1.2.2 Loading

The factor that causes the fatigue is cyclic loading. In reality, a structure is rarely subjected to cyclic harmonic loading. Instead, it undergoes a stochastically distributed load spectrum, as is the case for the bridge structures. A simplified introduction to the cyclic loading is to take the special case of cyclic loading with constant-amplitude. This type of loading has the form of a sinusoid function, with a constant-amplitude.

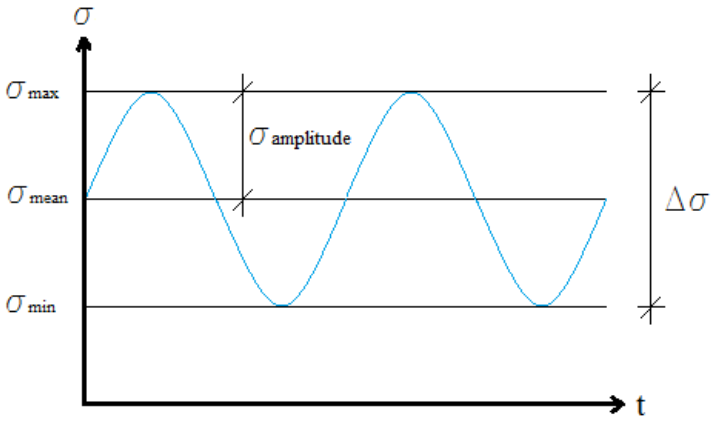


Figure 2-3 Constant-amplitude cyclic loading, including definitions for important loading parameters.

In Figure 2-3, definitions for important loading parameters for constant amplitude loading are presented. These are the stress range, $\Delta\sigma$, the mean stress, σ_{mean}, the stress amplitude, $\sigma_{amplitude}$. These are defined as:

$$\Delta\sigma = \sigma_{max} - \sigma_{min} \tag{2.2}$$

$$\sigma_{mean} = \frac{\sigma_{max} + \sigma_{min}}{2} \tag{2.3}$$

$$\sigma_{amplitude} = \frac{\sigma_{max} - \sigma_{min}}{2} \tag{2.4}$$

Where $\sigma_{\max}$ and $\sigma_{\min}$ are maximum and minimum applied stresses, respectively. Another important loading factor is the stress ratio, R:

$$R = \frac{\sigma_{min}}{\sigma_{max}} \tag{2.5}$$

The most influential of these factors is the stress range $\Delta\sigma$. But the mean stress, σ_{mean}, is also of importance for non-welded details.

The loading that many structures are subjected to in reality is variable-amplitude (VA) loading. As the term implies, the applied stress varies arbitrarily in time. In order to process the randomly occurring stresses, a stress histogram is used. The histogram is a conversion of the variable-amplitude stress history over time, into a series of constant-amplitude blocks, each defined by a constant stress range together with a corresponding number of cycles (Al-Emrani & Åkesson, 2013). The process is illustrated in Figure 2-4.

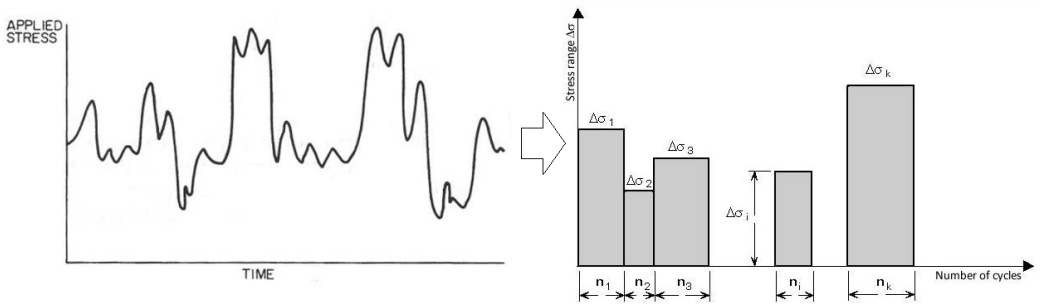


Figure 2-4 Conversion of the variable-amplitude loading history to a stress range histogram, (Al-Emrani & Åkesson, 2013).

2.1.2.3 Welding

As was seen in 2.1.2.1 the existence of geometric features, such as holes or attachments, leads to raising the stresses and therefore reducing the fatigue life of the details. When welds exist in the structure, usually those undesirable effects exist together with some other drawbacks. Welding introduces the risk of several problems occurring on the local level within the welded detail. The defects that occur in and around welds act as crack initiation locations, from which a fatigue crack can propagate more easily. Thus a weld defect shortens the fatigue life by considerably reducing the initiation phase. Some of the defects that can arise due to welding are weld start-stop points, weld ripples, lack of fusion, partial penetration, porosity, undercuts, and inclusions. (Al-Emrani & Åkesson, 2013). These defects are shown in Figure 2-5.

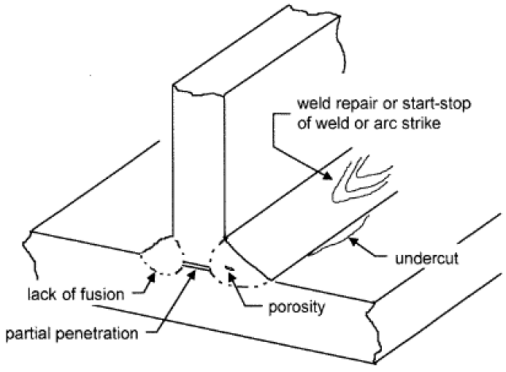


Figure 2-5 Some of the possible defects within welds, (Al-Emrani & Åkesson, 2013).

2.1.3 How fatigue resistance is quantified

The fatigue resistance of a certain detail is estimated by the concept of S-N curves. An S-N curve is a representation of how many cycles a detail can endure under constant-amplitude loading. The curve is obtained through testing of a large number of samples of a certain detail over a series of stress ranges, until failure and until a stress range is found when failure does not occur. The large number of test-samples gives widely distributed fatigue resistances, due to the variety of defects and irregularities that can exist in a detail. In addition to this, the data used for the S-N curves is a summary of testing over a long period of time and with different standards. Because of the dispersed results, a low percentile is chosen as the fatigue resistance due to safety reasons (Al-Emrani & Åkesson, 2013). The design resistance curves are illustrated in Figure 2-6. The 14 curves all have equal slopes in logarithmic scale and the slope is defined in two regions. Below 5 million cycles it is 3, and from 5 million up to 100 million it is 5. The curves are equally spaced in the logarithmic plot as well. The lowest curve is defined at 36 MPa at 2 million cycles, whereas the highest class is defined at 160MPa at 2 million cycles.

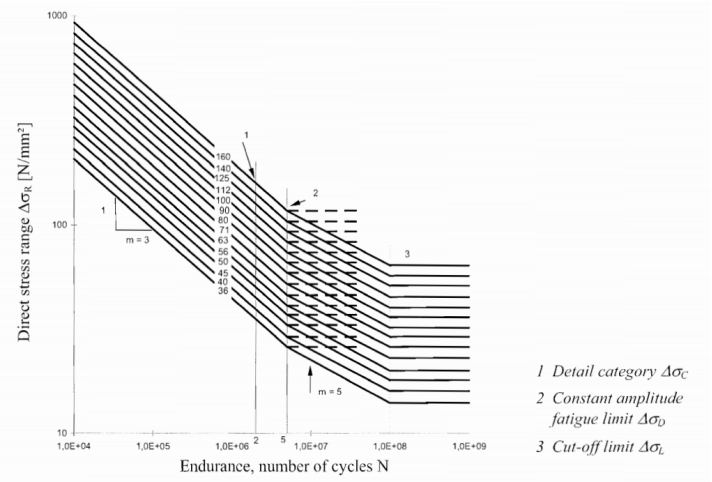


Figure 2-6 The S-N curves as they are defined in EN-1993-1-9. The three slope regions are discernable as well. The leftmost and steepest linear line is annotated by the slope m=3, the second and middle line is annotated by the slope m=5, the third and last line on the right-hand side is flat, i.e. the slope is m=∞.

2.2 Cycle counting methods

The most important loading parameters regarding fatigue are the stress range, the number of cycles and the mean stress, as stated in Section 2.1.2. It was also stated that the loading that the structure faces is not uniform and needs to be converted into a stress range histogram for the subsequent use in the fatigue assessment.
In order to arrive at a quantification of these parameters, different methods can be used. There exist several methods of deriving results from the variable-amplitude loading. Some of these methods are the level-crossing counting, peak counting, range-pair counting, simple-peak counting, reservoir counting and rainflow counting (ASTM International, 2017). The two most frequently used are the rainflow counting and

reservoir counting methods (Al-Emrani & Åkesson, 2013). These two, rainflow and reservoir, will be described here.

2.2.1 Transformation of stress data to the peak-valley dataset

Before the cycle counting can begin, the input data needs to be preprocessed into a dataset consisting of peaks and valleys, i.e. the data set should only contain local maximums and local minimums (Marsh et al., 2015). The conversion of the data into a dataset consisting of peaks and valleys is done through the process described in (Marsh et al., 2015), which is illustrated in Figure 2-7. The principle is that the value of a point, y_i, is compared to its two adjacent points, y_{i-1} and y_{i+1}. If the value of the point, y_i, is either smaller than or larger than both adjacent points, then a local extremum is found and the point is kept. If the check fails, then the point is removed from the dataset.

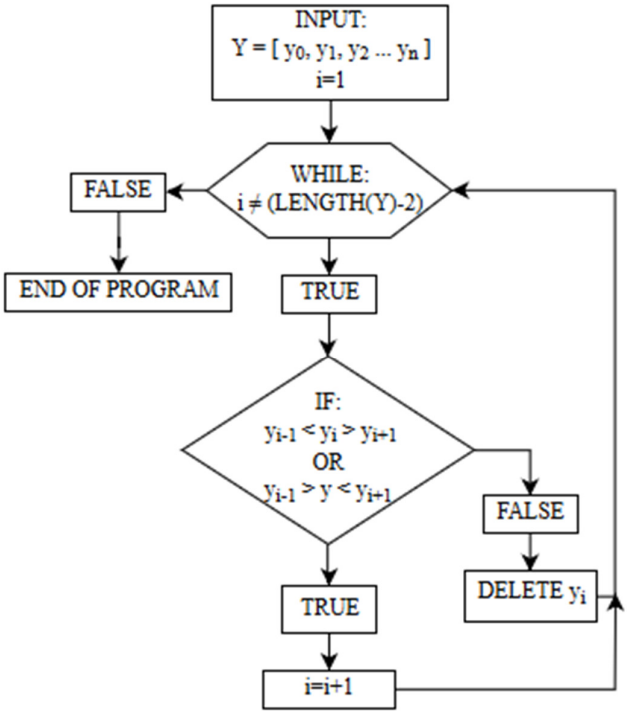


Figure 2-7 Flowchart of the algorithm for evaluating peaks and valleys in a dataset, such that only the peaks and valleys remain, flowchart based on (Marsh et al., 2015).

The result of the peak-valley algorithm is illustrated in Figure 2-8, in which a continuous function is reduced to the important parameters, namely the peaks and valleys, which are marked by the circular markers. The original function is the solid line and the dashed line is the resulting line between the peaks and valleys.

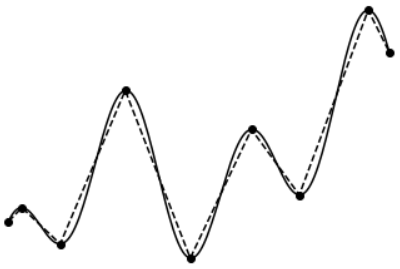


Figure 2-8 The result of applying the peak-valley algorithm. The solid line is the original function, the dashed line with circle markers is the resulting line which only contains peaks and valleys.

2.2.2 The rainflow cycle counting method

The rainflow counting can be done using either the four-point algorithm or the three-point algorithm. The original four-point algorithm was defined and outlined in (Matsuishi & Endo, 1968). The process for the algorithm described below is based on (Okamura, Sakai, & Susuki, 1979).
The rainflow cycle counting is described for the four-point algorithm, based on the steps outlined in (Amzallag, Gerey, Robert, & Bahuaud, 1994). The algorithm is illustrated in Figure 2-9.

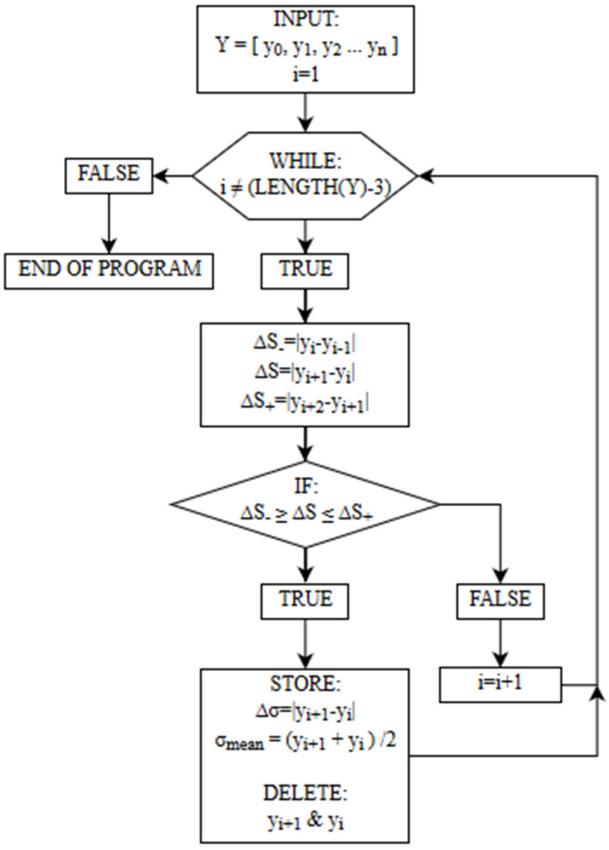


Figure 2-9 Flowchart of the algorithm for identifying full stress cycles within a dataset, flowchart based on (Amzallag et al., 1994).

The process illustrated in Figure 2-9 is here described by words. A full stress cycle is identified through the evaluation of four consecutive points, with the y-values defined as y_{i-1}, y_i. y_{i+1}, and y_{i+2}. These points form three connected lines. If the absolute value of the change in the y-direction for the middle line is smaller than or equal to the corresponding value for the adjacent lines, then a full stress cycle is found. The stress cycle is saved, and the two middle points are discarded. The operation is then repeated with the set of remaining points.

An example of the result of applying the rainflow cycle counting algorithm to the dataset from Figure 2-8 is shown in Figure 2-10. For this particular case, only one full cycle is found. The dashed line is the original line connecting the peaks and valleys and the dotted line is the remaining data after the rainflow cyclic counting is applied.

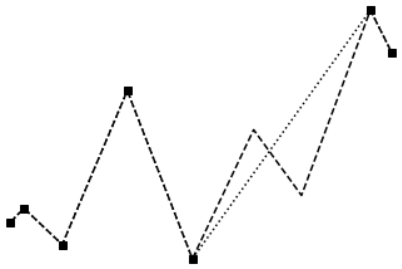


Figure 2-10 The result of applying the rainflow cycle counting. The dashed line is the original peak-valley dataset from Figure 2-8, the dotted line with square markers is the result of applying the algorithm. Only one full stress cycle was extracted for this particular function.

2.2.3 Residue stresses (half cycles)

The stress values that remain after the full cycle counting, correspond to values in such an order and of such a magnitude that they cannot form closed stress cycles, i.e. they cannot fulfill the condition stated in 2.2.2. There are some methods for accounting the residue cycles from the full cycle counting (Marsh et al., 2015). Two of these methods will be described here.

2.2.3.1 Half cycle counting

The principle of this method is that since no more full cycles can be found, the simplification is made that each pair of values forms a stress range. But because the two values do not form a whole stress cycle, their influence is reduced, by only attributing half a cycle to each counted pair of points (Marsh et al., 2015).

2.2.3.2 Simple rainflow counting methodology

The principle of this method is that the evaluated stress history should be representative of the loading of the structure. Thus, the reasoning followed that all values will at some point be repeated, this repeating of values should thus allow for the formation of full stress cycles. Therefore, if the cycles remaining after the full cycles are extracted are duplicated, then all the repeating cycles should disappear. Any residue remaining after this can be discarded (Marsh et al., 2015).

2.2.4 Reservoir cycle counting method

An alternative to the rainflow method and second most popular cycle counting method is the reservoir method. The basic concept of this method consists of comparing the signal from variable loading to a reservoir filled with the water. In the initial stage of analyzing stress-time history, water in the reservoir is restricted by two maximal peaks and troughs between them (Figure 2-11).

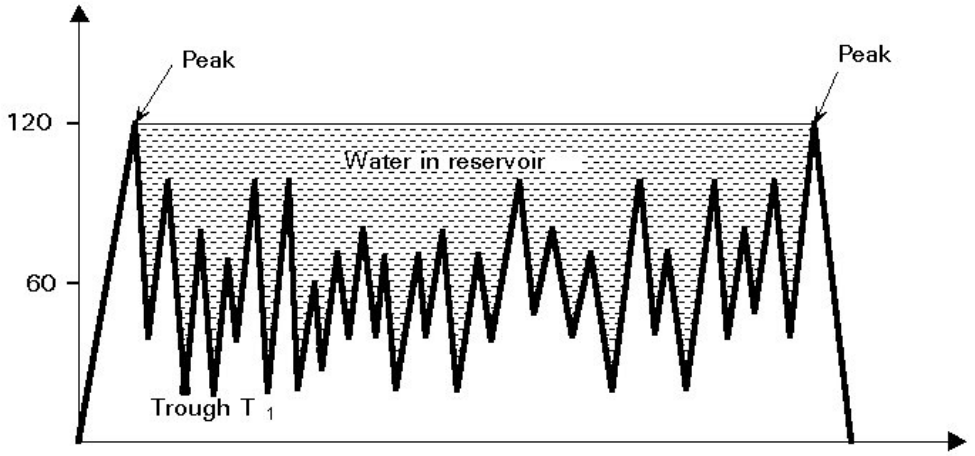


Figure 2-11 Water in the reservoir – initial stage for stress-time history, (Može, 2000).

A cycle counting in this method may be imaginarily presented as a gradual draining water out of the lowest point from the reservoir. Troughs become gradually empty, where one totally empty trough corresponds to the one cycle stress range, see Figure 2-12.

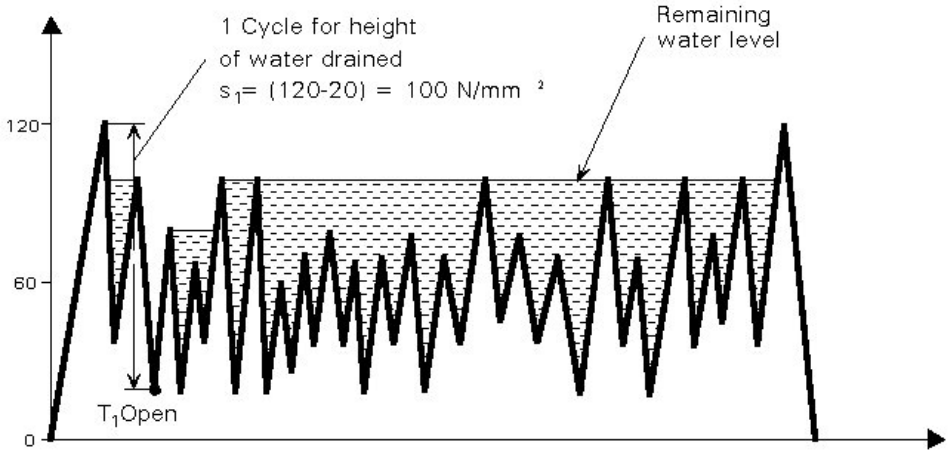


Figure 2-12 Position of the water after the first cycle, (Može, 2000).

Figure 2-12 illustrates how the procedure should proceed and presents an overall way of thinking about this method. Based on this example, the first cycle corresponds to stress range of 100 N/mm^2. In other words, totally empty trough T_1 matches to the first maximum cycle. Next steps consist of subsequent opening of following plugs (T_2, T_3, T_n) in the bottoms of the troughs and summing subsequent cycles (Figure 2-13).

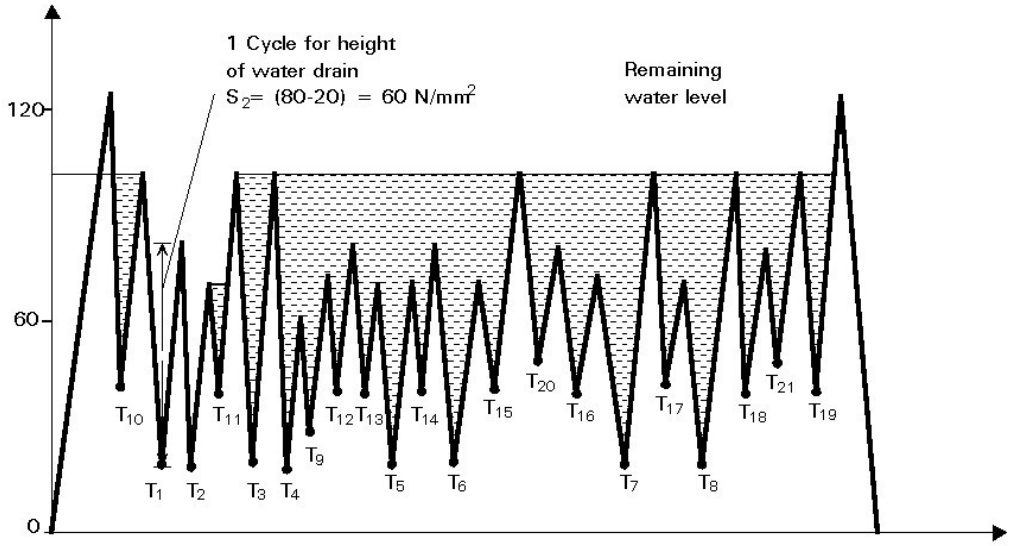


Figure 2-13 Following procedure of counting next cycles, (Može, 2000).

Similar to the rainflow method, a proper pre-processing of the signal is needed. The sample selected for cycle counting should start and end at the same stress level. It is essential to "fill the reservoir entirely" since the half cycles do not exist in this method. What is more, as the main advantage of this method is that cycles with higher stress ranges, which cause the biggest damage on the structure, will be calculated first, hence there is a smaller risk they will be omitted or treated as the half cycles (Maddox, 1991).

2.3 Fatigue design methods for bridges

According to Eurocode, two methods (λ-coefficient method and accumulative damage method) are approved for fatigue design and verification of bridge structures under variable-amplitude traffic loading. These two methods are briefly presented in this section.

2.3.1 Damage Accumulation Method

After transforming the variable-amplitude loading history into representative blocks of constant-amplitude loading, the damage accumulation method can be applied to the resulted stress histogram. For each individual stress range, there is a number of stress cycles that the detail can endure. The 'partial' damage caused by each of these stress ranges is the quotient between the number of cycles for this stress range, divided by the number of cycles that the detail can endure for this stress range. The accumulated damage is the sum of all the partial damages for each stress range block in the stress histogram. Thus:

$$D = \sum_i D_i = \sum_i \frac{n_i}{N_i} \tag{2.6}$$

where D is the accumulated damage parameter, D_i is the partial damage parameter for stress block i, and n_i is the number of loading cycles in block i obtained from the stress histogram (Al-Emrani & Aygül, 2013). N_i is the number of cycles to failure for the constant amplitude stress range $\Delta\sigma_i$ and is calculated as:

$$N_{i=}N_C \cdot \left(\frac{\Delta\sigma/\gamma_{Mf}}{\gamma_{Ff}\cdot\Delta\sigma_i}\right)^m \tag{2.7}$$

Where m is the slope of the S-N curve, $N_C = 2 \times 10^6\ cylces$, and γ_{Mf} and γ_{Ff} are partial factors for material strength and loading, respectively. In theory, it is assumed that the fatigue life of a specific construction detail will be exhausted when the accumulated damage D becomes equal or greater than 1.

2.3.2 Equivalent Stress Range

Another manner of evaluating damage caused by variable-amplitude loading is using the equivalent stress range. The equivalent stress range is defined as the constant-amplitude stress range which, if applied with the same number of cycles as the total number of cycles in the variable stress range, will cause the same damage to the structure. There are two cases regarding the equivalent stress range calculation, either all of the stress ranges are in one of the two slope regions, or they are within both of them. These two cases have different equations for the computation of the equivalent stress.

2.3.2.1 Equivalent stress range with one slope

In the case where all of the stress ranges are confined to one region, i.e. only in the $m = 3$ or $m = 5$ region, then the equivalent stress range can be obtained from the following formula:

$$\Delta\sigma_E = \left[\frac{\sum_{i=1}^{n} n_i \cdot \Delta\sigma_i^m}{\sum_{i=1}^{n} n_i}\right]^{\frac{1}{m}} \tag{2.8}$$

The expression was derived using the simplification of a constant slope of $m = 3$. Further details can be found in (Al-Emrani & Aygül, 2013).

2.3.2.2 Equivalent stress with a double slope

In addition to the equivalent stress range calculation defined in 2.3.2.1, the equivalent stress range can also be defined in the case where the stress ranges are not confined to only one region. The expressions for this conversion is not as compact as for the single slope conversion. The conversion can be done with either of two choices, either the equivalent stress range is calculated for the slope of $m = 3$ or $m = 5$. However, this choice does not matter for the end result. Below follows the derivations of the expressions, based on (Može, 2000).

Before the derivation, some additional context is provided. Consider a stress histogram, containing stress range blocks with their corresponding number of cycles. This data can be subdivided into three regions: the stress range blocks that fall under the cutoff limit,

the blocks within the $m = 5$ region, and the blocks within the $m = 3$ region. These regions are illustrated in Figure 2-6. The stress ranges in the cutoff region have infinite fatigue life (i.e. zero partial damage) and can thus be removed. Those within the $m = 5$ region will be denoted with the index j and those within the $m = 3$ region will be denoted with the index i.

Slope of 5

The damage caused to a detail is given by the summation of the number of cycles at a given stress range (n_i), divided by the number of cycles to failure in the said stress range (N_i). The desired end result of this conversion is the division of the total number of cycles, n_{tot}, with the number of cycles that the detail can resist at the given equivalent stress range, N_{eq}. If also combined with the expression of the resistance at a given stress range, the damage can be written as:

$$D = \Sigma \frac{n_i}{N_i} = \frac{\Sigma(n_i) + \Sigma(n_j)}{N_{eq}} = \frac{n_{tot}}{N_{eq}} = \frac{n_{tot}}{N_D \cdot \frac{\Delta\sigma_D^5}{\Delta\sigma_{eq}^5}} = \frac{n_{tot} \cdot \Delta\sigma_{eq}^5}{N_D \cdot \Delta\sigma_D^5} \tag{2.9}$$

The accumulated damage, without using the concept of equivalent stress is calculated as:

$$D = \sum \left(\frac{n_i}{N_i}\right) + \sum \left(\frac{n_j}{N_j}\right) \tag{2.10}$$

Converting this expression by substituting N_i and N_j with the expression for the resistance at a given stress range, it becomes:

$$D = \sum \left(\frac{n_i \cdot \Delta\sigma_i^3}{N_D \cdot \Delta\sigma_D^3}\right) + \sum \left(\frac{n_j \cdot \Delta\sigma_j^5}{N_D \cdot \Delta\sigma_D^5}\right) \tag{2.11}$$

Extracting the common factors gives:

$$D = \frac{1}{N_D \cdot \Delta\sigma_D^5} \left(\sum(n_i \cdot \Delta\sigma_i^3 \cdot \Delta\sigma_D^2) + \sum\left(n_j \cdot \Delta\sigma_j^5\right)\right) \tag{2.12}$$

Equating the two expressions:

$$\frac{1}{N_D \cdot \Delta\sigma_D^5} \left(\sum(n_i \cdot \Delta\sigma_i^3 \cdot \Delta\sigma_D^2) + \sum\left(n_j \cdot \Delta\sigma_j^5\right)\right) = \frac{\left(\Sigma(n_i) + \Sigma\left(n_j\right)\right) \cdot \Delta\sigma_{eq}^5}{N_D \cdot \Delta\sigma_D^5} \tag{2.13}$$

Removing the common factors on both sides of the equal sign gives:

$$\left(\sum(n_i \cdot \Delta\sigma_i^3 \cdot \Delta\sigma_D^2) + \sum\left(n_j \cdot \Delta\sigma_j^5\right)\right) = \left(\Sigma(n_i) + \Sigma\left(n_j\right)\right) \cdot \Delta\sigma_{eq}^5 \tag{2.14}$$

Finally, solving for the equivalent stress, $\Delta\sigma_{eq}$, gives:

$$\Delta\sigma_{eq} = \sqrt[5]{\frac{\sum(n_i \cdot \Delta\sigma_i^3 \cdot \Delta\sigma_D^2) + \sum\left(n_j \cdot \Delta\sigma_j^5\right)}{\left(\Sigma(n_i) + \Sigma\left(n_j\right)\right)}} \tag{2.15}$$

Which allows for the computation of the damage caused by the equivalent stress:

$$D_{eq} = \frac{\left(\Sigma(n_i) + \Sigma(n_j)\right) \cdot \Delta\sigma_{eq}^5}{N_D \cdot \Delta\sigma_D^5} \tag{2.16}$$

Slope of 3

The derivation of the equivalent stress, with the slope defined by $m = 3$, follows the same procedure that was outlined for $m = 5$. The derivation takes the following form, but a few steps have been omitted to make it more compact:

$$D = \frac{1}{N_D \cdot \Delta\sigma_D^3} \left(\sum(n_i \cdot \Delta\sigma_i^3) + \sum\left(n_j \cdot \frac{\Delta\sigma_j^5}{\sigma_D^2}\right)\right) \tag{2.17}$$

$$D = \frac{N}{N_{eq}} = \frac{\left(\Sigma(n_i) + \Sigma(n_j)\right) \cdot \Delta\sigma_{eq}^3}{N_D \cdot \Delta\sigma_D^3} \tag{2.18}$$

$$\frac{1}{N_D \cdot \Delta\sigma_D^3} \left(\sum(n_i \cdot \Delta\sigma_i^3) + \sum\left(n_j \cdot \frac{\Delta\sigma_j^5}{\sigma_D^2}\right)\right) = \frac{\left(\Sigma(n_i) + \Sigma(n_j)\right) \cdot \Delta\sigma_{eq}^3}{N_D \cdot \Delta\sigma_D^3} \tag{2.19}$$

$$\Delta\sigma_{eq} = \sqrt[3]{\frac{\left(\sum(n_i \cdot \Delta\sigma_i^3) + \sum\left(\frac{n_j \cdot \Delta\sigma_j^5}{\Delta\sigma_D^2}\right)\right)}{\Sigma(n_i) + \Sigma(n_j)}} \tag{2.20}$$

2.3.3 λ – method

The damage accumulation method requires the calculation of the stress spectrum due to the passage of the design traffic composed of various lorries or trains with different axle weights. Therefore, calculations fatigue assessment using damage accumulation method will be extensive. In order to provide the engineer with a simpler and shorter calculation method, the concept of damage equivalence (λ-method) given in Eurocode. The concept is based on approximating equivalent stress range for the examined detail by using only maximum stress range induced in the detail multiplied by a so-called "damage equivalent factor". In this meaning, equivalent stress range can be approximated by $\Delta\sigma_E$ or $\Delta\sigma_{E,2}$, where the latter is the equivalent constant amplitude stress range corresponding to 2 million cycles. The design or assessment in this method is reduced to a simple resistance control, which takes the form of:

$$\gamma_{Ff} \cdot \overbrace{\lambda \cdot \Phi_2 \cdot \Delta\sigma_{FLM}}^{\Delta\sigma_{E,2}} \leq \frac{\Delta\sigma_C}{\gamma_{Mf}} \tag{2.21}$$

Where:

γ_{Ff}	is a partial safety factor for fatigue loading
γ_{Mf}	is a partial safety factor for fatigue strength
λ	is the fatigue damage equivalent factor related to $2 \cdot 10^6$ cycles
Φ_2	is the dynamic factor
$\Delta\sigma_{FLM}$	is the maximum stress range due to the fatigue load model
$\Delta\sigma_C$	is the reference range value of the fatigue strength

The λ-coefficient is obtained as follows:

$$\lambda = \lambda_1 \cdot \lambda_2 \cdot \lambda_3 \cdot \lambda_4 \leq \lambda_{max} \tag{2.22}$$

Where

λ_1	is the span factor taking into account the length of the span and the structure type
λ_2	is the volume factor taking into account the traffic volume
λ_3	is the time factor taking into account the design life of the bridge
λ_4	is the lane factor taking into the account the traffic on more than one lane
λ_{max}	is the maximum damage equivalent factor taking into account fatigue limit

Further references to λ factors can be found in the "Fatigue design of steel and composite bridges" report by (Al-Emrani & Aygül, 2013).

2.4 Fatigue assessment of welds using local approaches

Besides using "detail categories" method which employs nominal stresses acting on the detail to assess its fatigue life, there are other methods which consider the local stresses close to the weld region to evaluate the fatigue life of the welded connection. These methods often incorporate FE models and help in more accurate manner assess stress prevailing in the element caused by local stress raiser. It is emphasized that both of the presented methods here are only applicable to welded structures.

2.4.1 Hot-spot stress method

The hot-spot method was initially used for designing tubular structures, pressure vessels, and welded ships. Nowadays, this method is also widely used in other structures, where accurate evaluation of stresses is important. The advantages of using this method can be easily seen in structures with complex geometry, where assessing nominal stress is very difficult e.g. in case of warping effect (Figure 2-14(a)), shear-lag effect Figure 2-14(b) or caused by curvature and flange curling effect (Figure 2-14(c)).

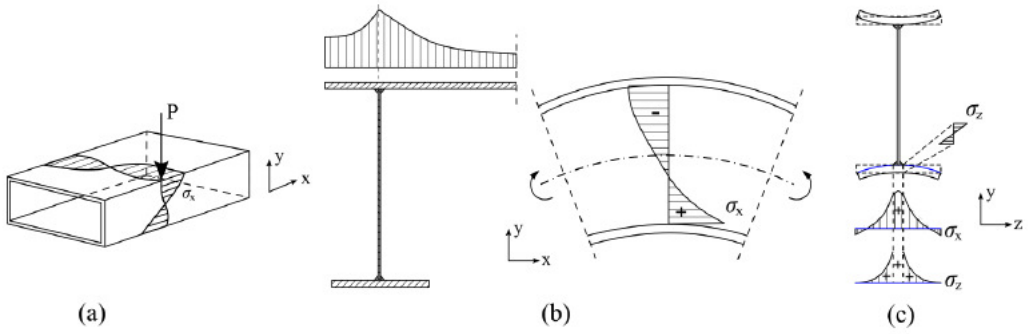


Figure 2-14 Examples of complex stress state. a) warping effect; b) shear-lag effects; c) curvature and flange curling effects, (Al-Emrani & Aygül, 2013).

This method is based on studying stress gradients at the point of a crack initiation – so-called hot-spot point. Using the hot-spot stress method is beneficial for the user since not only geometrical stress concentration is included but also a local redistribution of the loads is taken into consideration. The stresses computed in the hot spot method can be derived from a simple post-processing of the finite element analysis results.

Another reason, which makes the hot-spot stress method more attractive for designers, is the reduced number of S-N curves. In the hot-spot stress method, all stress raisers are accounted for on the load effect, in contrast to the method of nominal stress where geometrical changes are considered in resistance side (reducing a fatigue strength of the investigated detail). Therefore, methods incorporating nominal stresses need several S-N curves to address the fatigue resistance of the same detail having a range of different sizes (e.g. various plate thicknesses). Figure 2-15 presents a comparison of detail categories of the same connection for both the nominal stress method and the structural hot-spot stress (SHSS) method. As can be seen, a single detail category for SHSS method can replace several detail categories in nominal stress method (Al-Emrani & Aygül, 2013).

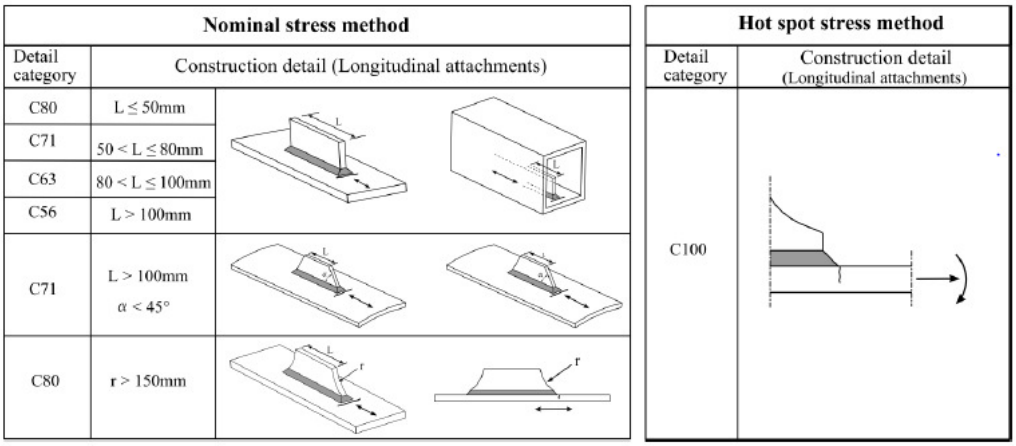


Figure 2-15 Recommended fatigue classes based on the nominal and the hot-spot stress method according to EN 1993-1-9:2005, (Al-Emrani & Aygül, 2013).

2.4.2 Effective notch stress method

Stress raisers like notches and geometrical discontinuities are commonly known for their negative influence on fatigue life for welded steel structures. The features are an almost inseparable part of the welded details. Welded joints are considered to be the places where stresses are affected by local geometry and shape of the weld itself (Figure 2-16).

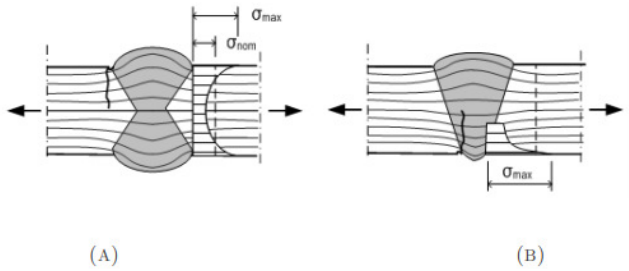


Figure 2-16 Stress concentration in (A) double-sided butt welds, (B) single-sided butt welds, (Al-Emrani & Åkesson, 2013).

The peak stress caused by the influence of local stress raisers and the geometrical change in the welded connection is defined as the notch stress. In most cases, notch stresses can reach substantially high values. The stress levels achieved in the notch depends on the "notch radius", i.e. sharpness of the notch. In theory, for a notch with a radius of zero, the elastic stress approaches infinity. However, that is not the case in reality and peak stresses larger than the yield stress of material will be redistributed due to the plasticity. In order to avoid the problems related to the singularity of stress field at the notches in the finite element analysis, the stresses are evaluated at a certain distance from the weld toe (singularity point).
The general idea of using effective notch stress consists of calculating the stress in the point of crack initiation with a prescribed notch radius. To execute a calculation like this, an accurate model must be provided. For these purposes, usually, 2D or 3D FE models are used, where all elements (welds, local discontinuity etc.) should be defined separately. With the created model, calculations can be performed. The next step is to assign related to this detail S-N curve and compare them with fatigue strength of detail (Al-Emrani & Aygül, 2013).

2.5 Traffic loads

As it is commonly known, traffic loads applied to the bridges are variable action and design based on real data is almost impossible. In the case when such a data is unavailable, there is a need to transform variable-amplitude loading into representative constant amplitude loading using one of the two methods earlier described in Section 2.2: rainflow cycle counting or reservoir method.

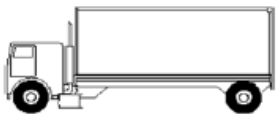
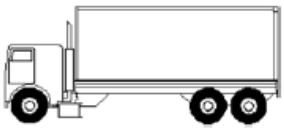
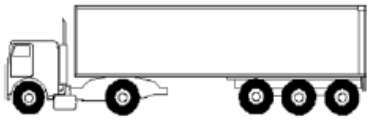
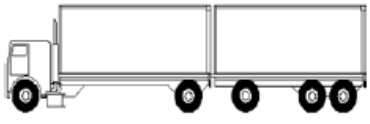
Furthermore, complex data like traffic load history is rarely available during the design process. Therefore, Part 2 of EN 1991 (2003) provides fatigue load models which can be used for fatigue design. These models can be used as a substitute to simulate traffic on the bridge in the future. For purposes of this thesis, only fatigue load model 4 and 5 (FLM 4 and FLM 5) will be discussed, further references to other models can be found in EN 1991-2: 2003. Thus, FLM 1, 2, and 3 are omitted from further explanation.

2.5.1 Fatigue load model 4, FLM 4

FLM 4 presented in Eurocode is a set of typical, most common lorries occurring on the European roads. From this set, 5 lorries can be distinguished with different geometries, loadings, and number of axles. FLM 4 was created based on the traffic measurements coming from Auxerre (France). The aim of this traffic model is to represent heavy

traffic existing on the roads. EN 1991-2:2003 provides also a variety of FLM 4 which corresponds to local traffic, medium-distance traffic and long-distance traffic. The difference between the traffic types is a percentage of each lorry used in the traffic model, see Table 2-1. In order to use FLM 4, the code provides a total annual number of lorries passing a bridge, called $N_{obs,}$ which allows establishing traffic volume on the bridge. What is more, it is important to mention that FLM 4 is recommended to use with damage accumulation concept and stress-time history incorporating cycle counting.

Table 2-1 Set of the lorries and their percentage in the traffic, EN 1991-2:2003.

Vehicle type				*Traffic type*		
				Lorry percentage		
Lorry	*Axle spacing [m]*	*Equivalent axle loads [kN]*	*Wheel type**	*Long distance*	*Medium distance*	*Local traffic*
	4,5	70 130	A B	20,0	40,0	20,0
	4,20 1,30	70 120 120	A B B	5,0	10,0	5,0
	3,20 5,20 1,30 1,30	70 150 90 90 90	A B C C C	50,0	30,0	5,0
	3,40 6,00 1,80	70 140 90 90	A B B B	15,0	15,0	5,0
	4,80 3,60 4,40 1,30	70 130 90 80 80	A B C C C	10,0	5,0	5,0

2.5.2 Fatigue load model 5, FLM 5

Fatigue load model 5 (FLM 5), presented in the EN 1991-2: 2003, is based on the real traffic records. From all presented models in Eurocodes, this one is the most accurate since it takes into consideration real traffic loads. What is important to mention, data coming from traffic measurements must be extrapolated into past and into future and supplemented by statistical calculations (if relevant) to properly assess the fatigue life of the bridge.

Detailed description and recommendations about FLM 5 may be found in Annex B in EN 1991-2: 2003.

3 Fatigue assessment of existing steel structures

In this chapter, the general procedure for fatigue evaluation of existing steel bridges will be briefly discussed. The aim of this chapter is to point out what actions can be undertaken if the bridge is nearing the end of its lifespan and the first symptoms of fatigue damage, i.e. small cracks, start to appear. The entire procedure is divided into 4 phases.

The first phase should start from the study of the documentation, include a site visit and simple calculation to check limit states of the structures. This step is made by the engineer himself. There is no need for support of any experts or use of sophisticated equipment from the laboratory. Visual inspection of the critical details in the bridge should be sufficient for the first phase.

In the case when simple calculation or visual inspection is not passed, the more advanced investigation should be considered. Phase II should be devoted to more accurate calculations and checks from Phase I. For these purposes measurements of real traffic can be done, instead of using models proposed by codes. Furthermore, some NDT can be carried out to better evaluate the real state of the bridge. Operations in Phase II are mainly conducted by the engineer alone, however, some activities may require the use of more advanced devices.

If the limits and assumptions made in Phase II were not fulfilled, more advance investigation is recommended. In Phase III experts in particular fields should be incorporated. Also, very advanced methods like fracture mechanics or probabilistic methods can be used.

Phase IV in this procedure can be called “remedial actions”. That means checks made in previous steps have not been passed and some interventions must be done if the bridge is to remain in use. In general, if the lifespan of the bridge is spent and limit states are not fulfilled anymore, then the structure should be demolished and replaced by the new one. However, this is often not the case. Due to intensive traffic and a high cost of erecting a new bridge, old structures are expected to stay in service as long as possible. Therefore, repairs or strengthening of the bridge can be accepted as one of the remedial actions. As an alternative to these two solutions, a reduction of the load can be done. It can be obtained by limiting or prohibiting entry of the heavy vehicles on the bridge. What is important and worth mentioning is that the new residual service life has to be evaluated. The last remedial action which can be used is to intensify monitoring of the bridge. Examples of intensified monitoring can be Structural Health Monitoring (SHM) system, which is partially a subject of this thesis. As an example, SHM, by means of strain gauges, gives a possibility to control strains and react if the values reach the critical values. Systems like this are often expensive devices, however, in comparison to the cost of the new bridge or to the strengthening of the structure, the costs are still relatively low. A more detailed description of particular phases will be provided in following sections. Figure 3-1 presents the overall procedure of all the phases discussed here.

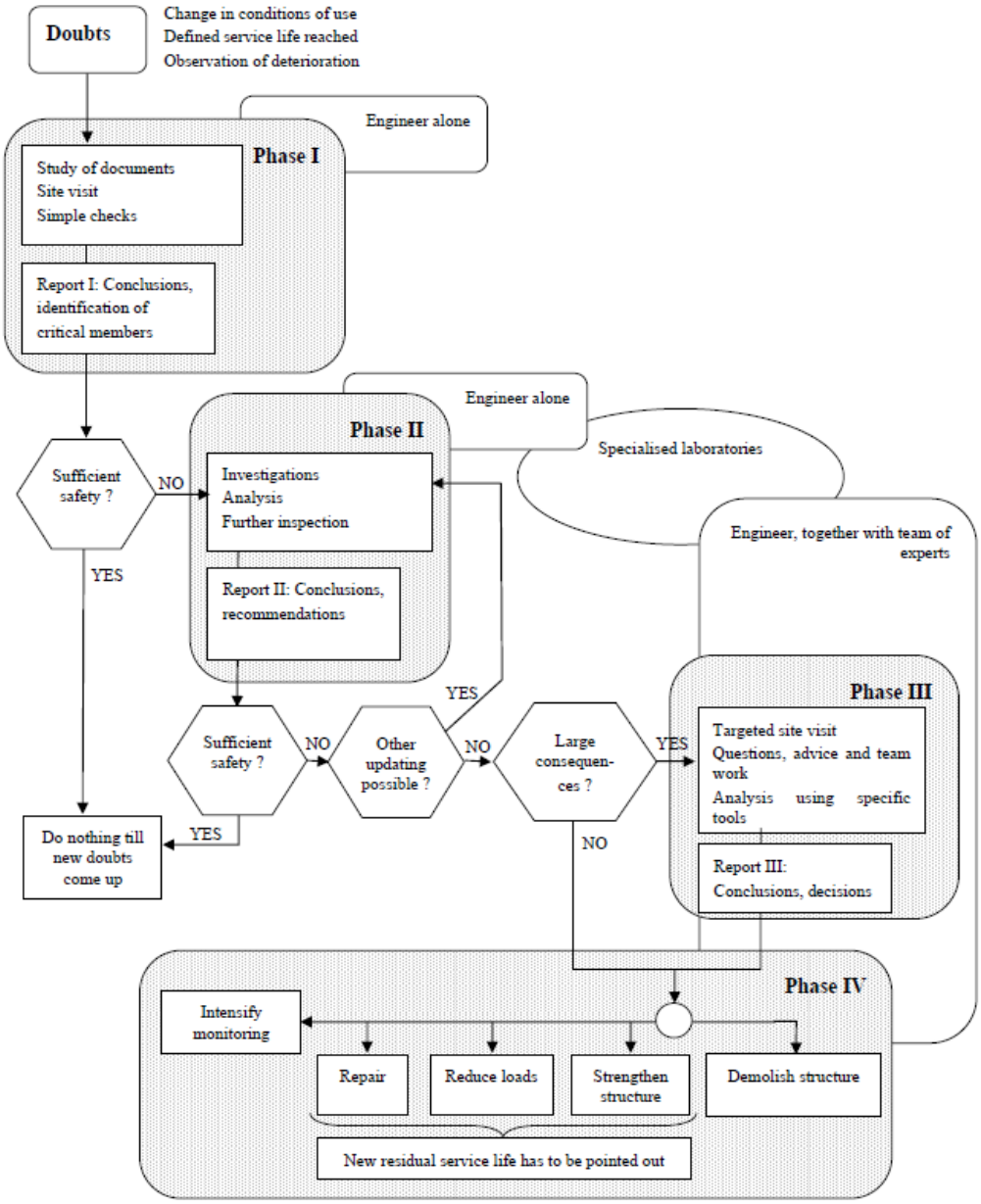


Figure 3-1 Flowchart of four phases of the described procedure for fatigue assessment of existing steel bridges, according to (Kühn et al., 2008).

3.1.1 Phase I – preliminary evaluation of a structure

Phase I of this procedure can be called preliminary evaluation. In this stage, basic methods are used and the engineer uses own knowledge to identify and assess the most critical details in the construction. Research should be started from a comprehensive study of the provided documentation. In some cases, particularly for old structures, fatigue documents are rarely available. An engineer in a situation like this should use his experience and support himself with currently valid state-of-the-art material and literature. Furthermore, the engineer should not only rely on technical drawings and

calculations attached to documentation but also take into consideration maintenance documents and any records of repairs in structures.

Regarding visual inspections, the investigation should start from looking for the most visible deteriorations like corrosion, damages in supports or expansion joints, and cracked welds. This check can be made without using additional equipment, no further investigation is needed in this phase.

For the preliminary assessment, the calculation methods proposed in the codes for design can be used. For these purposes, the methods presented previously in Chapter 2 can be used. For evaluation of the fatigue in the first phase, it is not recommended to use the hot-spot stress method or the effective notch stress method since they are too advanced for preliminary evaluation. However, if the simple checks from codes are not fulfilled, a list of priorities should be done and further steps or remedial actions should be recommended.

3.1.2 Phase II – a detailed investigation of a structure

Phase II, in other words, detailed investigation, deals with refined methods used in stage one. The decision about commencing the next step in investigation usually is made if the safety of structure was not proved by simple assessment. In this stage, experts or external specialized firms can be involved.

What is more, as soon as critical details of the construction are identified, remaining fatigue life can be estimated for details which fatigue life capacity has not been used up yet. This calculation can be performed by damage accumulation method (Palmgren-Miner), which was described in Chapter 2.

As a following step in this phase, more accurate load information there should be provided. As it was said before, traffic load is variable in time and using conservative models proposed in codes is convenient for the design purposes. However, in fatigue evaluation of existing structure, they are not reflecting the reality and the real state of the construction. Instead of using models coming from codes, data from real measurements should be utilized. Here it is important to emphasize that data coming from measurements has to be extrapolated into the past and at the same time into the future. Predictions like these are able to serve better as a loading occurring on the bridge.

With updated loading information, the engineer can go for a more exact calculation. To obtain this, the refined computational model should be provided since static models usually are too conservative and obtained stresses can be up to 40% higher than in the reality. For these purposes, 2D or 3D FE models are recommended utilizing structural elements using shell or solid elements. We should bear in mind that using simple calculation models omit secondary effects, out-of-plane deformations, distortions. That is why those models often overlook deformation-induced fatigue cracks.

As the last step of updates in phase II, information about resistance should be refined. The engineer facing fatigue evaluation of existing bridge is often placed in a situation where some information is lacking. Due to missing information, some assumptions must be made in phase I. Phase II is devoted to verification of these assumptions which

were put into question. To dispel all doubts, material tests on specimens coming from structures are recommended. This action is advisable e.g. if the type of used steel is not known and there is a risk of brittle fracture.

3.1.3 Phase III –Expert assessment and advanced methods

In the case that checks made in phase II were not fulfilled or there is a still a high risk of failure and remedial actions are not taken into consideration yet, experts should be incorporated into the investigation. Commonly used methods in this phase, like fracture mechanics or probabilistic methods, will be briefly discussed. Further description can be found in Kühn et al. (2008).

3.1.3.1 Fracture mechanics

Contrary to the classification method based on S-N curve, fracture mechanics is able to deliver prognostic information about the crack size or the reaming fatigue life if the crack was detected. With this information, inspection intervals can be easily established if other remedial actions will not be undertaken. However, there is a limitation of this method which should be noted. When it comes to evaluation of the "distortion-induced" cracking (caused by secondary stresses), it is rather complicated. Therefore, during employing this method in a case when secondary stresses are expected to be significant, high caution is recommended.

In linear elastic fracture mechanics (LEFM), the basic case of fracture mechanics, stress state in the area adjacent to the tip of the crack can be described by stress intensity factor(SIF) usually shown by K or in case of cyclic stress range, stress intensity factor range ΔK. It can be described as follows:

$$\Delta K = Y \cdot \Delta\sigma \cdot \sqrt{\pi \cdot a} \tag{3.1}$$

Where:

a	crack size (depth or length)
$\Delta\sigma$	applied cyclic stress range
Y	the product of various multipliers which account for the geometry of the crack, the geometry of the cracked body and (if necessary) the effect of non-uniform applied stress

Utilizing the Paris' equation, stress intensity factor can be related to the rate of crack growth as follows:

$$\frac{da}{dN} = C \cdot \Delta K^m \tag{3.2}$$

Where:

C	constant of the Paris equation
m	exponent of the Paris equation

By integration of the Paris equation over the crack size, the remaining fatigue life can be calculated:

$$N = \int_{a_0}^{a_c} \frac{da}{C \cdot \Delta K^m} \tag{3.3}$$

Where

a_0 initial crack size (depth or length)

a_c final or critical crack size (depth or length)

It is important to note that according to LEFM, small initial cracks are always assumed to exist before loading. Past research has shown that with regard to welded structures this assumption is correct (Kühn et al., 2008).

3.1.3.2 Probabilistic methods

Since variations exist in many of the parameters affecting the design equation, the theory of probability finds application in the advanced assessment of the fatigue. Using probabilistic methods "probability of failure" P_f or "reliability index" $\beta = -\Phi^{-1}(P_f)$ can be evaluated (where Φ is the standard normal cumulative distribution). The main difference between deterministic and probabilistic methods is that deterministic values (e.g. $a_0 = 0{,}2mm$) are replaced by statistical distribution in the probabilistic methods (e.g. $a_0 = N[0{,}2\ mm; 0{,}045\ mm]$). With data like this and having the limit state function (design equation simply written in probabilistic form), the assessment of the probability of failure is possible. The next step is to evaluate if obtained P_f is acceptable or not (Kühn et al., 2008).

3.1.4 Phase IV – remedial actions

If none of the assessment presented in the phases I-III have proven that sufficient safety is provided, the bridge should either be stopped from using for the sake of safety or remedial measures should be implemented.

It is important to note in this phase that damage which occurs to the bridge is really caused be fatigue. Otherwise, in many cases, methods mentioned in this chapter are not applicable. However, when the crack is detected as an effect of propagation of the fatigue damage, the common following inspections are possible:

- Control of the crack propagation
- Visual inspection of the crack mouth opening (under cyclic load)
- Using scanning electron microscope to inspect the surface of the crack

All the points mentioned above are valid if the crack is found. Therefore, the engineer should pay a lot of attention to areas in the structures which are susceptible to crack due to fatigue.

The most popular reason of cracking welded steel structures is often poor quality of welds like lack of the fusion, porosity or improper treatment after welding, for example avoiding grinding of the weld surface and the area adjacent to it. There are two typical places where the crack starts its propagation. The first one is the root of the weld. Cracks which start growing from the root of the weld are very dangerous for construction since it is almost impossible to detect them in early stages by visual inspections.

Therefore, during the designing process, it is important to select detail category in a manner to avoid this type of failure. As a rule-of-thumb, in this case, size of the fillet weld should be approximately equal to the thickness of the plate. What is more, special care should be taken to the quality of the weld in order to avoid lack of fusion and porosity.

The second typical starting point of a crack is in the toe of the weld. This type of crack is less detrimental in the fatigue since it gives the possibility to detect it before the total collapse of the structure occurs. Figure 3-2 presents the two typical initiation points of cracking in the welded connections.

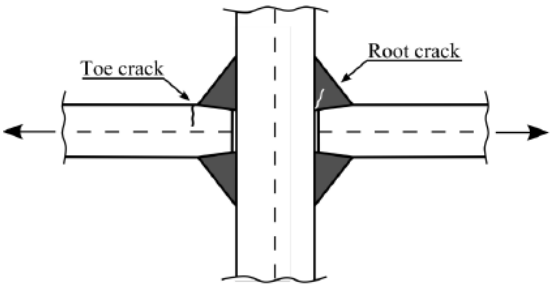


Figure 3-2 Two typical initiation points of cracking in the welded connections, (Al-Emrani & Åkesson, 2013).

A good practice to prevent or postpone the toe cracking is to get rid of an undercut. The undercut is a micro defect located along the weld in the base metal, which is the result of thermal contraction. However, this type of defect can be easily removed by grinding the weld and the area adjacent to it.

Other common factors that can affect fatigue failure include:

- Cold cracks caused by environmental conditions
- Restraint caused by geometrical imperfections, distortion or out of plane bending
- Vibration caused by wind, traffic or earthquake
- Web gaps
- Geometrical changes caused by place changes of cross-section or places of connections with perpendicular elements
- Web breathing caused by repeated web buckling (Kühn et al., 2008).

3.1.4.1 Repair and strengthening

After the inspection, when the crack has been detected and its origin has been established, retrofit measures are recommended to repair cracked element. Before selecting the appropriate method of repairing or strengthening, cause of the damage must be considered. The most common repair and strengthening methods are listed below:

- removal of crack by grinding
- re-welding

- surface treatments such as TIG dressing or hammer peening
- adding plates or adhering fiber reinforced polymer (FRP) sheets
- bolted splices using high strength preloaded bolts
- shape improvements (e.g. smoothening geometric transitions)
- modification of the connection detail

In Table 3-1 from Kühn et al. (2008), most typical failures are compiled with possible repair methods. Also, each of the combinations is assessed according to the grading scale presented at the bottom of the table.

Table 3-1 Applicable repairs and strengthening methods for fatigue failures in welded structures, (Kühn et al., 2008).

		Repair and strengthening method							
		Grinding	Re-welding	Surface treatment	Adding plates	Bolted splices	Shape improvement	Stop holes	Modification connection
Causes of fatigue	weld defects	G	G	N	G	E	G	N	G
	lack of fusion	F	G	G	G	E	E	G	E
	cold cracks	F	G	G	F	E	G	G	E
	restraint	F	F	G	G	E	G	G	E
	vibration	F	F	G	G	F	G	F	E
	web gaps	G	F	G	F	N	N	F	E
	geometrical changes	F	F	F	G	E	N	F	G
	web breathing	N	F	F	F	F	N	G	E
		E: Excellent		**G: Good**		**F: Fair**		**N: Not good**	

3.1.4.2 Other measures

Besides repairing and strengthening, other forms of intervention for the bridges exist. Following actions can be included in the last group of the remedial measures:

- Intensified monitoring
- Reduction of the traffic
- Demolition of the bridge

Intensified monitoring can be justified to use only in situations when safety is at the sufficiently high level and it is confirmed by one the expert methods (e.g. probabilistic fracture mechanics). Moreover, as additional conditions for using intensified monitoring, more frequent inspections on site must be conducted.

various intensified monitoring systems may be distinguished: Weight In Motion (WIM) systems which record axel weights as well as axel distances, simple vehicle counting devices, and continuous strain measurement systems.

For the choice of the most suitable system for monitoring, besides economical aspects, environmental conditions and surroundings should also be taken into consideration. For

example, not all systems may be appropriate for railway bridges because of the electromagnetic fields from the overhead wire.

Reduction of the traffic may be described as the cheapest method of remedial measure and still gives the possibility of prolonging the usage of the bridge. Reduction of the traffic must be carried out based on information gathered during Phase II (or Phase III).

Demolition may be said that is a more unpleasant necessity instead of remedial action. In a situation, when none of the presented above methods is able to prolong fatigue life of the bridge, demolition of the structure is the only reasonable way to prevent failure during service.

4 Theory of optimization

The problem of optimization is the search for the optimal solution to a problem, which takes the form of the maximization or minimization of a process or function. There are several methods that can be used when facing the problem of optimization, these can be categorized into following groups:

- Analytical.
- Graphical.
- Experimental.
- Numerical.

The names of these methods strongly imply how they function. The analytical one depends on classical differentiation of the function. The graphical method relies on the plotting of the function and visually determining the extrema. The experimental method works by experimentally changing parameters and evaluating as the process progresses. Finally, the numerical method uses iterative numerical evaluation to search for better solutions until an end condition is met.

For the process of optimization, a target function, f, must first be determined and defined as a function of one or more parameters, $n_1, n_2, \dots, and\ n_p$. The function value, F, of the function, f, should be scalar-valued. Thus, the target function used for the optimization is of the following form $F = f(n_1, n_2, \dots, n_p)$ (Antoniou, Murray, Wright, & SpringerLink (e-book collection), 2007).

Thus, the problem of optimization is the search of the extrema for a target function, be they maxima or minima, global or local. Generally, the point of interest is the global maximum or minimum. The optimization process can be performed with or without constraints. The target function can either be smooth and differentiable or non-smooth. For non-smooth functions, the gradient cannot be calculated, which limits the optimization procedures available. An issue regarding optimization is that many local extrema may exist, out of which only one is global. Thus, an optimization process that is started near a local extermum might result in a false result.

4.1.1 Optimization in relation to this thesis

For this thesis, the main focus will point to experimental and numerical optimization, due to the problematic nature of the optimization process entailed in the project. The optimization takes the form of the minimization of the error between the measured values, and the computed values from the FE model. Thus, the model is being calibrated to better represent reality, whereas it is the error function that is subjected to the optimization.

4.1.2 Experimental optimization

The basis of the experimental optimization is outlined in the short introductory part of this chapter. Namely, manually changing and updating parameters for the problem at hand and using an engineering mindset and judgment to evaluate the progress being made.

4.1.3 Numerical optimization methods

Numerical methods can be subdivided into gradient-based methods and gradient-free methods. The names imply their way of working. They will be further expanded upon in the following text.

4.1.3.1 Gradient-based optimization

The gradient-based methods work by computing the derivative, or gradient, at a given set of the parameters that the function depends on. Based on the computed derivative, or gradient, a step is taken in the direction of the computed gradient. The parameters that the function depends on are now updated in accordance with the step taken. A convergence check is computed and if conditions are met, the optimization process ends.

4.1.3.2 Gradient-free optimization

Unlike the gradient-based methods, and as the name implies, the gradient-free methods work even when the derivative, or gradient for a multivariable function, is unavailable. In addition to this, gradient-free functions also work when the function being evaluated is non-smooth, non-continuous or noisy. There are several methods for utilizing gradient-free optimization, some of them are Nelder-Mead Simplex, Genetic Algorithm, Particle Swarm, Simulated Annealing and Divided Rectangles Method (Hicken, Alonso, Farhat, & Rajnarayan, 2012). In this study, the focus will only be on the Nelder-Mead Simplex optimization, which is specified below.

Nelder-Mead simplex method

For a function of n variables, $f(x_1, x_2 \dots x_n)$, a simplex of $n + 1$ variables is formed around an initial guess, illustrated in Figure 4-1 for a two and three variable function.

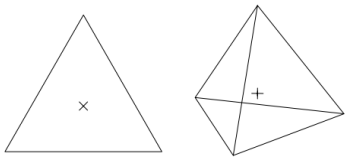


Figure 4-1 Two simplexes, one for two variables and one for three variables.

Based on the function values for the points defining the simplex, the method attempts to replace the point which has the worst function value, with a new point which has a better function value. The new point defining the function value is created from one of three actions, reflection, expansion and, inside or outside, contraction. The point that the method attempts to create is created along a line which passes through the worst point and the centroid of the remaining points. Should none of these three actions succeed in finding a new better point to replace the worst point, the entire simplex is shrunk towards the best point, such that only the best point remains in the new simplex (Nocedal & Wright, 2006).

These actions are illustrated in Figure 4-2, in which a simplex of two variables is shown. Subfigure I defines the points, P_1, P_2 *and* P_3 defining the triangle, P_C defines the centroid of all the points excluding the worst, P_H represents the highest value and P_L

represents the lowest value. Subfigure II illustrates the reflection of the worst point, subfigure III illustrates the expansion of the reflected point, subfigure IV illustrates the outside contraction of the reflected point, subfigure V illustrates the inside contraction and subfigure VI illustrates the shrinkage of the simplex. In all subfigures, the dotted lines represent the original simplex and the dashed line represents the line along which the new point is formed.

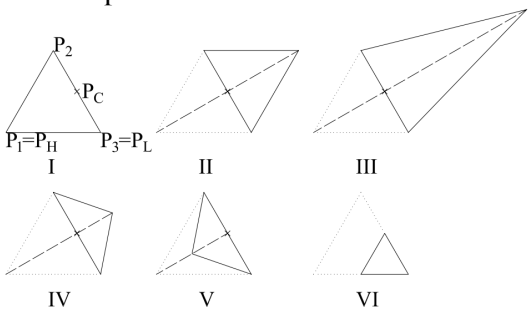


Figure 4-2 *(I) the starting conditions for the simplex, with P_1, P_2 and P_3 defining the simplex, with $P_3=P_L$ being the lowest point, $P_1=P_H$ being the highest point, and P_C being the centroid of all points excluding the worst. The figure also illustrates different actions that the Nelder-Mead method can take per iteration. (II) illustrates the reflection. (III) illustrates the expansion of the reflected point. (IV) illustrates the contraction of the reflected point, i.e. the external contraction. (V) illustrates the internal contraction. (VI) illustrates the shrinkage of the simplex.*

The algorithm by which the Nelder-Mead optimization works is illustrated in Figure 4-3. The figure is based on the algorithm outlined and devised by the authors, in (Nelder & Mead, 1965).

The process described in Figure 4-3 is also described here but in words.
The method first tries the reflection of the worst point.
If this action proves fruitful, i.e. giving a better value than the currently lowest function value, the methods attempts to expand further in this direction.

> If the reflected expansion is further fruitful, then the method accepts this as a solution for the iteration, replaces the worst value with the reflected expanded value and starts over.
>
> If the reflected expanded value is not lower than the reflected value, then the method accepts the reflected value as a solution for the iteration and replaces the worst value with the reflected value and starts over.

If the reflected value is not lower than the best value, but still better than some of the other values, then the method accepts the reflected value as a solution to the iteration and replaces the worst value with the reflected value and starts over.
If the reflected value, which from the beginning was the worst value, is still the worst value, the method evaluates if the reflected value is worse than its old value.

> If it is better than its former value, the method replaces the old value with the reflected and proceeds to the next step.
>
> If it is worse than its former value, the method proceeds to the next step.
>
> I.e. these two if statements lead to the same path, but with different input values.

The method now updates and calculates the contracted point, be it internal or external, and its function value. It then evaluates if the contracted point is better or worse than the originally worst point.

If it is better, then the method accepts this as a solution to the iteration and replaces the old worst point with the contracted.

If it is worse, then the method resorts to its last option, it shrinks the simplex towards the best point.

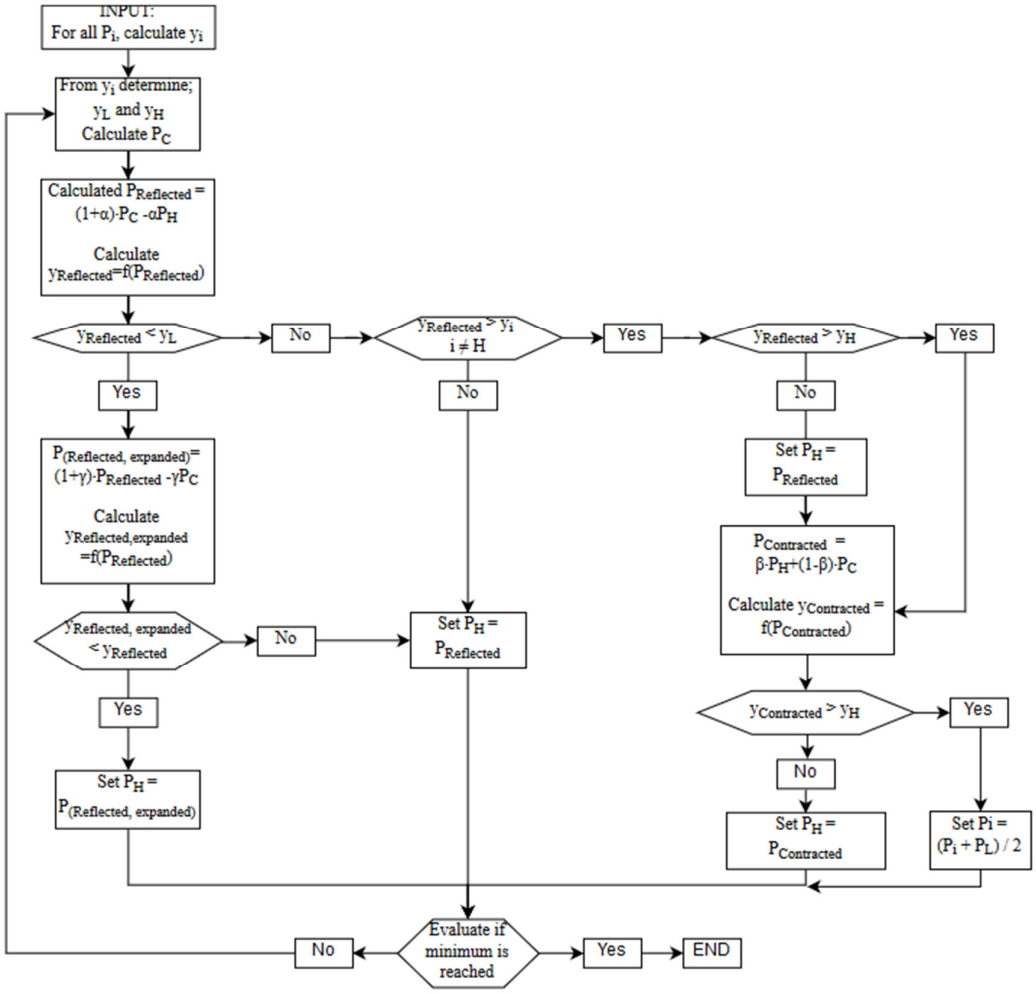


Figure 4-3 Flowchart of how the Nelder-Mead optimization method operates. Based on a figure in (Nelder & Mead, 1965). The notation P_i defines a point on the simplex, whereas the notation y_i defines the function value at the point P_i.s

4.2 Optimization evaluation

During the optimization process, the FE-model is to be modified so that it better represents the bridges actual behavior. In order to evaluate the accuracy of the model and how closely it represents reality, some method of quantifying the discrepancy between the two is needed. This discrepancy is crystallized in the comparison between

the collected stresses from the bridge, and the ones calculated from the FE-model. This comparison takes the form of an error function, which serves as the target function which is to be minimized, to find the model configuration with the smallest error.

4.2.1 Different error functions

There are several ways to measure the errors. Some of these methods are the mean bias error (MBE), the mean absolute error (MAE) and the root mean squared error (RMSE), aside from these there are several others but these will not be expanded upon. The ones mentioned here have their own advantages and disadvantages, as will be briefly expanded on in their respective chapters below.
In general, the error is defined as:

$$e_i = x_{measured} - x_{predicted} \tag{4.1}$$

4.2.1.1 Mean bias error

The mean bias error, MBE, is the simple summation of all the differences between the measured and predicted values, e_i, followed by the division by the number of measurements, n.

$$MBE = \frac{\sum_{i=1}^{n} e_i}{n} \tag{4.2}$$

The method does show if the prediction is biased, meaning that it will show if the prediction gives results that are over- or underestimations, as compared to the measured (J. Willmott & Matsuura, 2005). The method also has the inherent problem that positive and negative errors might negate each other. However, this what gives it the ability to show model bias.

4.2.1.2 Mean absolute error

As the name implies, the mean absolute error, MAE, is concerned with the absolute value of the error. Thus, the calculation of the MAE is the summation of the absolute value of each error, divided by the number of measurements.

$$MAE = \frac{\sum_{i=1}^{n} |e_i|}{n} \tag{4.3}$$

This method of quantifying the error of the prediction has the advantage that it gives the magnitude of the error, i.e. how large the error is regardless of sign (J. Willmott & Matsuura, 2005).

4.2.1.3 Root mean squared error

The root mean square error, RMSE, like the mean absolute error is concerned with the absolute value of the error. However, this method is more sensitive to large errors, due to the squaring of each individual error. Thus, if large errors are more important for the evaluation, then this method provides a good way of quantifying the error (J. Willmott & Matsuura, 2005).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}} \tag{4.4}$$

4.2.2 Mean value comparisons

In order to easily compare the final results derived from the optimization process, there are several ways of quantifying the mean values. Three will be described here, for a set of values, x_i, along with the total number of values, n.

4.2.2.1 Mean value

The simplest comparison, which is given by the summation of all the values, followed by the division of the total number of points evaluated, is given by:

$$MV = \frac{\sum_{i=1}^{n} x_i}{n} \tag{4.5}$$

4.2.2.2 Mean absolute value

A slight variation of the former, with the modification being implied by the name. Instead of just summarizing all of the values, the summation is instead of the absolute values, which eliminates the problem of negative and positive values negating each other. As given by:

$$MAV = \frac{\sum_{i=1}^{n} |x_i|}{n} \tag{4.6}$$

4.2.2.3 Root mean square value

The root mean squared value is what the name implies, namely that the value is calculated by squaring each error, adding them together, dividing the sum with the number of compared values and then taking the square root of it. As given by:

$$RMSV = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}} \tag{4.7}$$

This way of computing the mean value gives a higher sensitivity towards larger values.

5 Calibration of the provided FE-model

This chapter describes the process of the calibration of the provided FE model. This part could be described as the experimental optimization of the model, with regard to the minimization of the error. The finite element model was created in Robot Structural Analysis (RSA). The primary goal of the original model was to evaluate stresses in Ultimate Limit State (ULS) under the movement of the supports caused by vibration during the erecting of a new bridge near the existing bridge. For the purposes of accuracy with regards to ULS, the original model was sufficient, however, for fatigue evaluation more advance validation was needed.
The model was calibrated by adjusting parameters so that the computed stresses better fit the measured stresses, which were generated during the passage of two trucks with a known weight. The collected stresses could be illustrated in the shape of influence lines. The measurements were performed Kungliga Tekniska Högskolan (KTH) in 2015 (Leander et al., 2015).

5.1 Description of the measurements program

The bridge was equipped with 23 strain gauges and 4 accelerometers. The installed equipment was divided between two areas, see Figure 5-1. The measuring equipment on the southern part of the viaduct was located between supports S13 and S14, shown in Figure 5-2, and in case of the river part, between support III and IV, illustrated in Figure 5-3. Figure 5-4 present a photo of the measurement area between support III and IV.

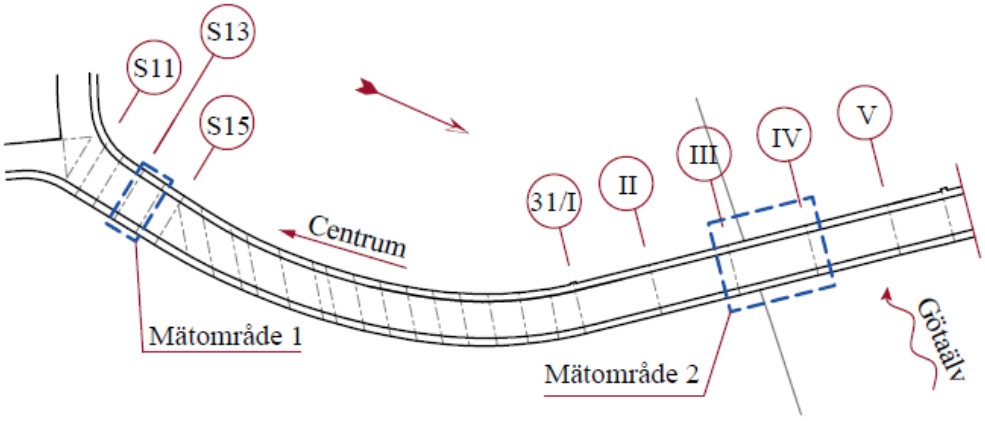


Figure 5-1 *Overview of the southern part of the bridge, showing the two measuring areas. Mätområde 1 means measurement area 1, Mätområde 2 means measurement area 2, Centrum refers to the City Center, (Leander et al., 2015).*

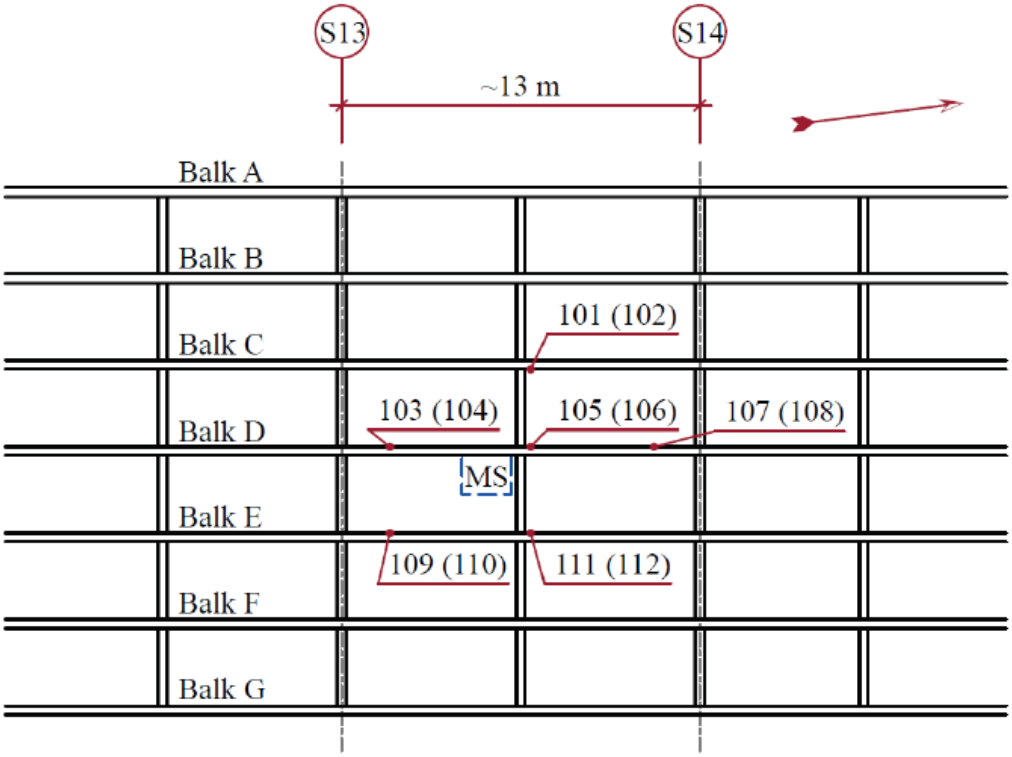


Figure 5-2 Location of the strain gauges between supports S13 and S14 (southern viaduct). Even numbers correspond to strain-gauges installed to bottom flanges,(Leander et al., 2015).

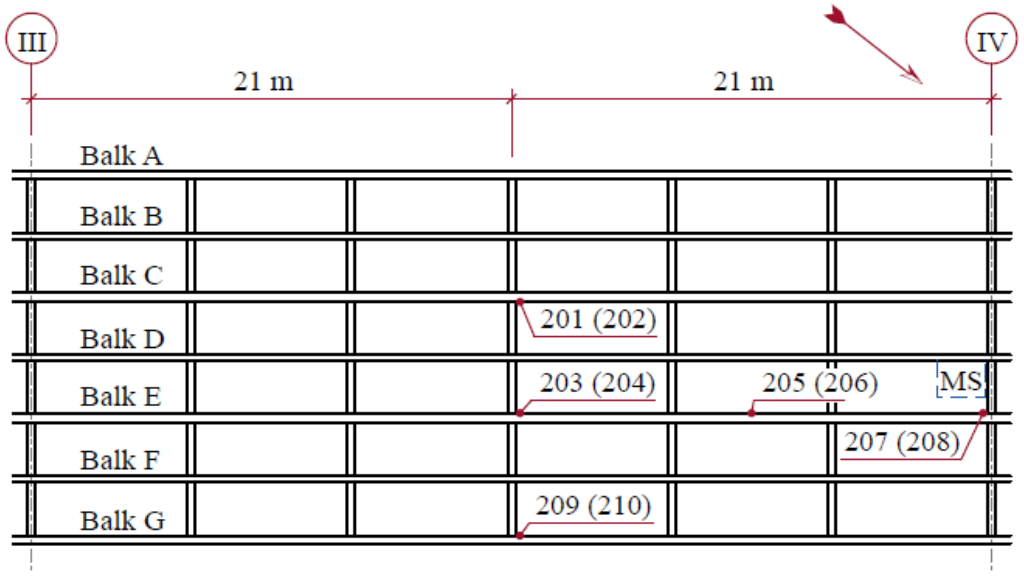


Figure 5-3 Location of the strain gauges between supports III and IV (river part). Even numbers correspond to strain-gauges installed to bottom flanges, (Leander et al., 2015).

Figure 5-4 A photo of the measuring area between supports III and IV, (John Leander, 2015).

After the installation of the strain gauges on the bridge, the calibration of the system was performed based on the passages of two trucks. The trucks with the known weight of 25.0 metric tons and 25.1 metric tons respectively, see Figure 5-6, traversed the bridge with a known velocity. The central axis between the two trucks coincided with the main girder E, see Figure 5-5.

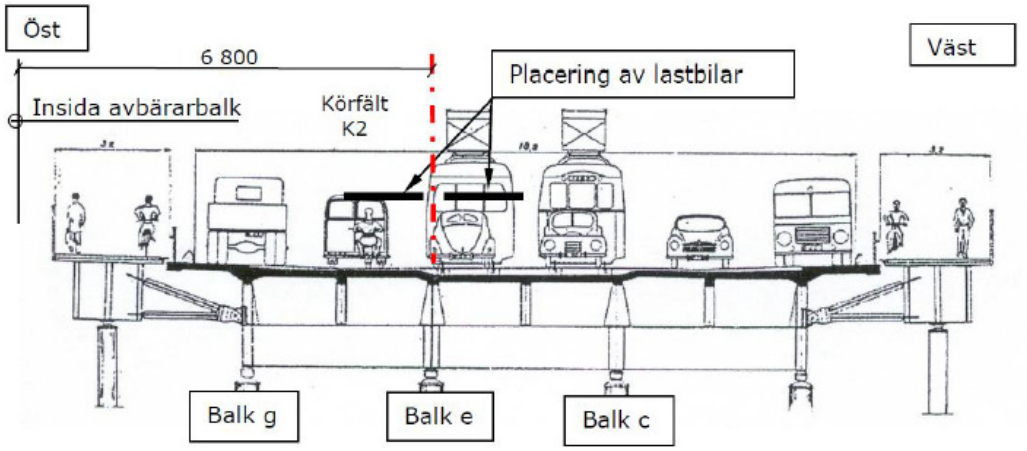


Figure 5-5 Placement of trucks, the measurement between support III and IV. The Swedish words in the figure are translated as follows, Öst: east, Väst: west, Placering av lastbilar: placement of the trucks, Balk: girder.

Figure 5-6 Photos of the trucks used for calibration of the system.

During each passage, strains were registered continuously. Records of registered measurements were utilized for calibration of the system installed on the bridge and later on for validation of the FE model.

5.2 Introduction of the provided FE model

Due to the significant size of the bridge and for greater ease when processing the data, the whole FE model was divided into 5 sub-models. For purposes of this thesis, only the river part of the bridge was utilized for the process of calibration and optimization, see Figure 5-7 for an illustration of the bridge model.

The total length and width of this sub-model, for the river part, is equal to 135 m and 23.76 m, respectively. Materials and sections used in the creation of the model are compatible with data presented in Chapter 1.

The model consists of two types of the finite elements. The deck of the bridge was modeled by means of shell elements and all the steel elements of the bridge were modeled as beam elements. Due to the regular shape of the deck, Coons mesh was utilized. Reinforcement was not modeled in the concrete since its influence in stress range of the regular traffic is negligible. According to the documentation, the Göta Älv Bridge was designed as a steel bridge, not a composite bridge. Shear studs in the form of U-profiles existed but their number was theoretically insufficient for the complete composite action. Nevertheless, from the measurements it can be inferred that some interaction between concrete and steel exists. Therefore, the concrete deck and steel girders were connected with each other by means of elastic springs. In these springs all rotations and translations, besides UX direction, were blocked. The stiffness of the springs was established for:

- 500 000 000 kN/m – above supports
- 5 000 000 000 kN/m – in the span

The main girders and secondary beams were modeled with the function of offset i.e. eccentricity regarding the connection between the beams and the deck was introduced. The values of the offset were selected in this way to allow placing upper flange of the

girders under the bottom surface of the concrete deck. What is important to mention here is that crossbeams were modeled without an offset in the original model.

Fork connections between columns and steel girders were modeled using the supports with all rotations released aside from the UZ direction which was blocked. Supports in the axis 1I were additionally modeled with blocked translations of UX and UY. The nodes, where the model was cut into sub-models, were blocked only in UY and UZ directions.

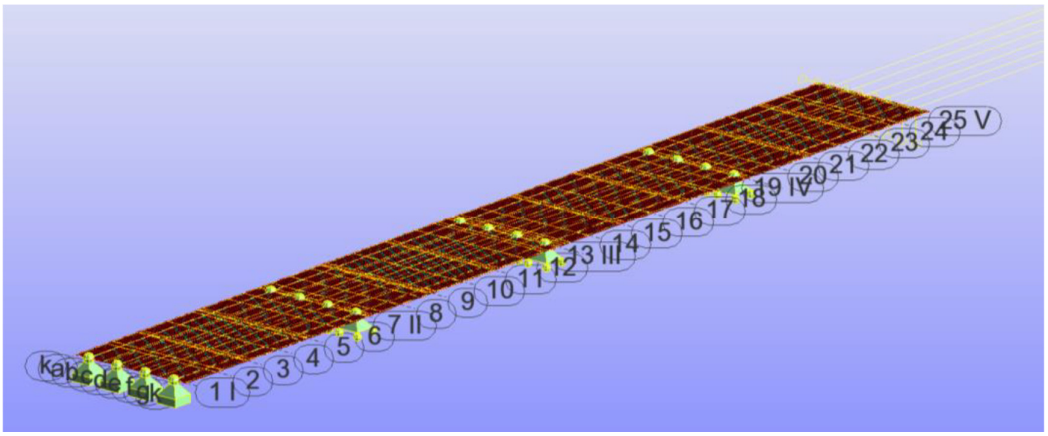


Figure 5-7 *South river part of the bridge model. The numbering from 1 to 25 refers to the cross beams, the numbering from I to V refers to the supports, and the notation ranging from a to g refers to the longitudinal beams. The k notations are for the edge of the concrete, notated in Swedish by "kant betong".*

5.3 Manual calibration

The main aim of the manual calibration was to find parameters which could be used in the automatic calibration. During this validation among selected parameters, the two extreme cases for each parameter were investigated to check if the intermediate values are worth examining in the optimization process. What is more, a secondary objective of manual calibration was to exclude unimportant parameters from automatic validation. In other words, the aim was to narrow the range of parameters which should be optimized, thus shortening the computational time. The parameters for manual calibration were chosen according to engineering intuition and uncertainties regarding technical documentation. The manual calibration was performed for cases listed below:

- Case 1- original FE model
- Case 2a – a shift of the crossbeams 2000 mm below the deck
- Case 2b – a shift of the crossbeams with released rotations in connections to the main girders
- Case 3a – offset of the main girders 100 mm
  Case 3b – offset of the main girders 200 mm
  Case 3c – offset of the main girders 300 mm

- Case 4 – change of the E-modulus of the concrete, the range of the concrete grade from C16/20 to C90/105
- Case 5 – change of the E-modulus of the steel, the range of the steel with E-modulus from 180 to 210 GPa

Case 2 in comparison to original, the FE-model was modeled with crossbeams 2000 mm below the deck and connecting them only with main girders. The difference between two first cases is that crossbeams in case 1 are not only connected to steel girders but also to the concrete deck, see Figure 5-8.

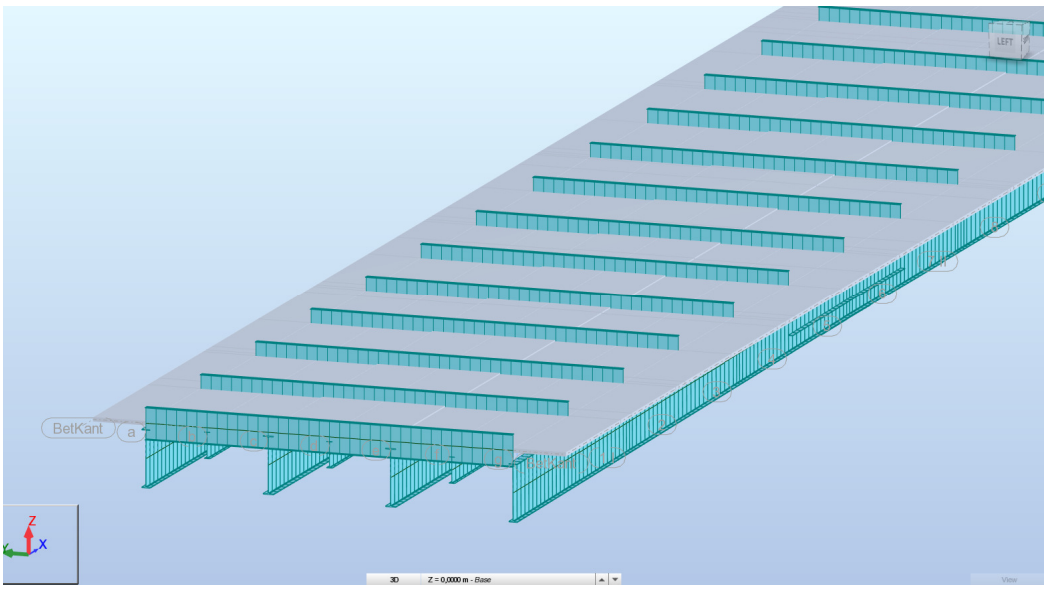


Figure 5-8 The FE-model for case 1.

This approach is not reflecting the reality, however, this approach yielded better results for ULS. What is important to note is that the offset function does not work correctly in this case to model the crossbeams. When the offset is introduced to the crossbeams, there still exists intermediate nodes in-between the two parallel main girders. In this case, these intermediate nodes are contributing to increasing the moment of the inertia for the deck in the direction of the crossbeams. In the reality, the crossbeams are located close to the bottom flanges of the main girders and are only connected with the steel web of the main girders, see Figure 5-9.

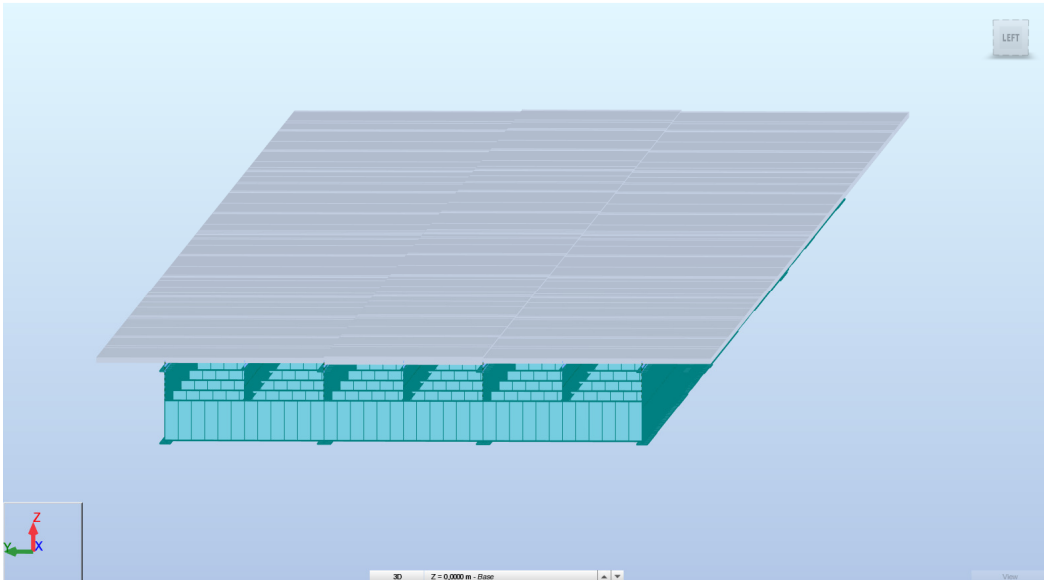


Figure 5-9 The FE-model for case 2.

The problem with intermediate nodes was solved creating new crossbeams, which were shifted 2 m below the deck. The shifted crossbeams were then only connected with girders by means of artificially created rigid bars (Figure 5-10). Rigid bars means that their stiffness is ten times bigger than adjacent elements. This solution was much easier to implement in the already existing FE model, rather than creating compatible nodes which is not the most efficient tool in RSA. The obtained results are supposed to be the same in comparison to connecting nodes using compatible nodes.

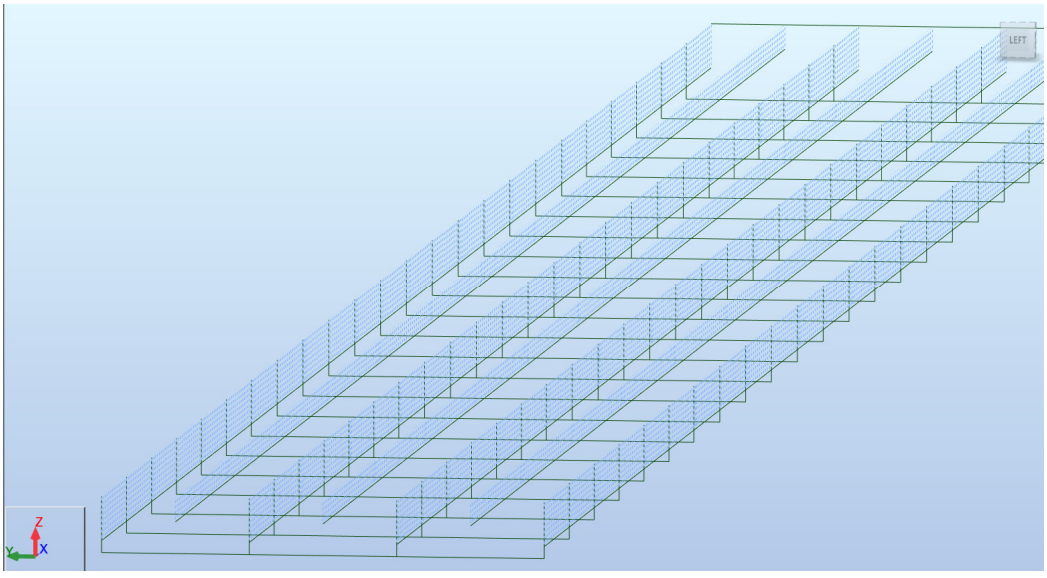


Figure 5-10 Computational model of the beams' grillage with an offset of the longitudinal beams and artificial rigid bars.

In **Case 3** the values of the offset, set to 100, 200 and 300 mm, are referring to ribs placed under the concrete deck. For a better understanding and justification of the case

3, Figure 5-11 explains this concept. For a simplification, the deck was modeled using only two thicknesses of the deck, either 180 or 270 mm. Due to this simplification, all small variations in the thickness of the deck are neglected and the contributions of the small ribs marked in the red color in Figure 5-11 are ignored as well. To compensate for the latter, the additional offset was implemented. The height of the ribs below the deck changes from 100 mm to approximately 300 mm, depending on the place where it is located and the inclinations in the deck's plate.

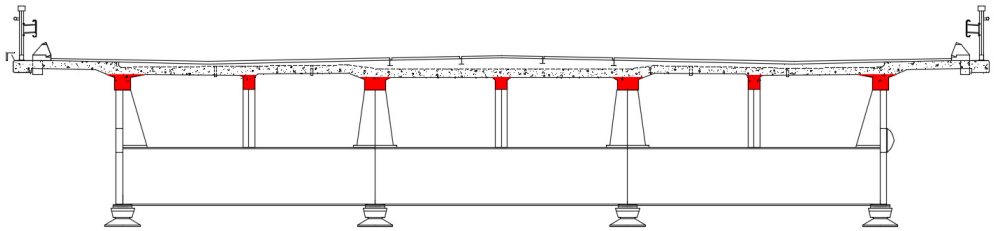


Figure 5-11 Cross-section of the bridge with marked ribs below the deck.

Each of the cases was developed incrementally, that means changes from Case 2 were implemented in Cases 3, 4 and 5. For further investigation, some cases included sub-cases (denoted by a, b, or c). this was to look into different alternatives in order to find the best results in comparison to the measurements from the bridge.

5.4 Result and conclusions after manual calibration

In this section, the results obtained from all described cases are presented, apart from cases 4 and 5, seen in Figure 5-12 to Figure 5-16. The results are presented as influence lines of the normal stresses at the locations of strain measurements. For each diagram, the influence line evaluated from calibration measurements is also drawn to allow for easy comparison of the results.
Results from cases 4 and 5 are not presented in this chapter since they yield very similar results to case 3. Plotting these two additional curves would make the already crowded plots even more cluttered.
Regarding the line styles, they are consistent for all of the figures in question, Figure 5-12 to Figure 5-16.

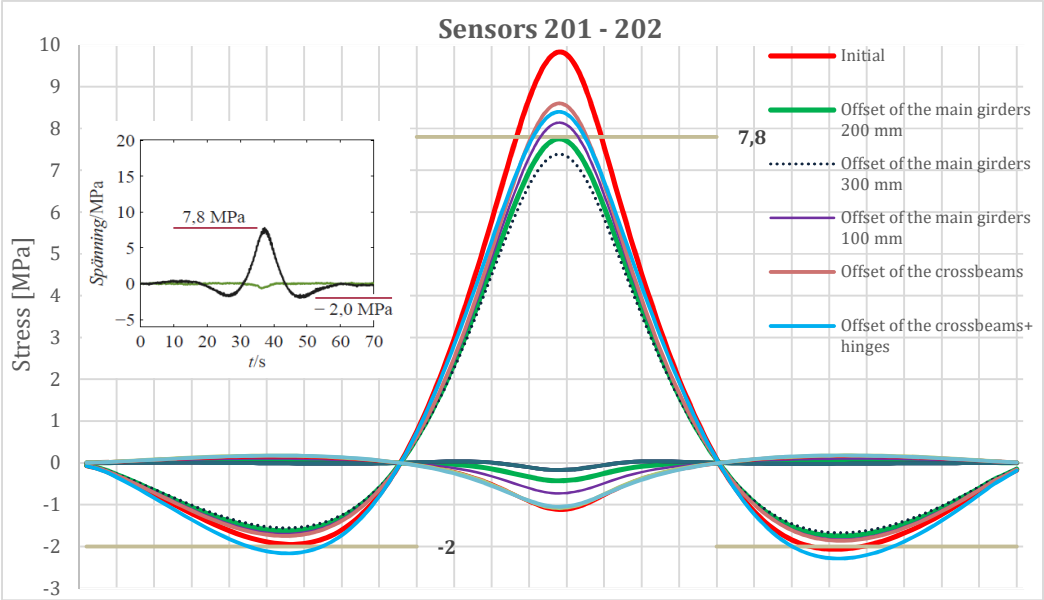


Figure 5-12 Influence lines for sensors 201-202. The most important influence lines are the red one, representing the original model and the green one representing the case yielding the best values.

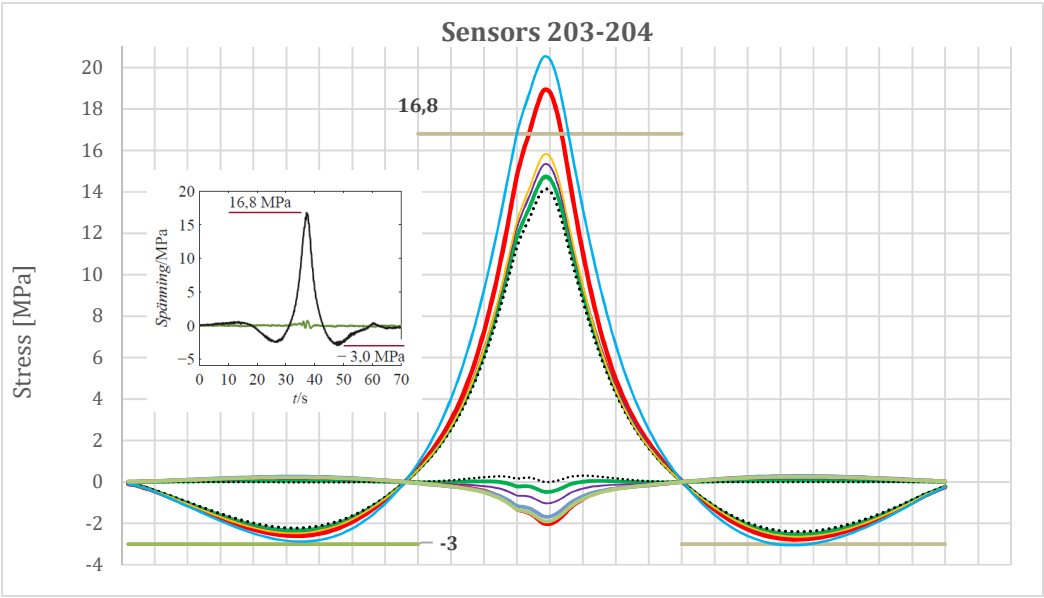


Figure 5-13 Influence lines for sensors 203-204.

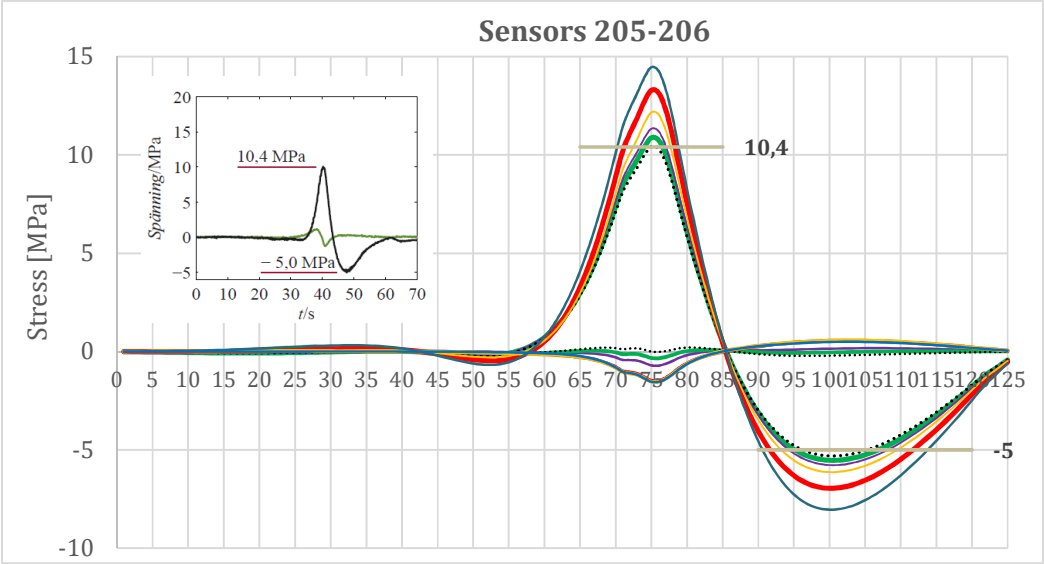


Figure 5-14 Influence lines for sensors 205-206.

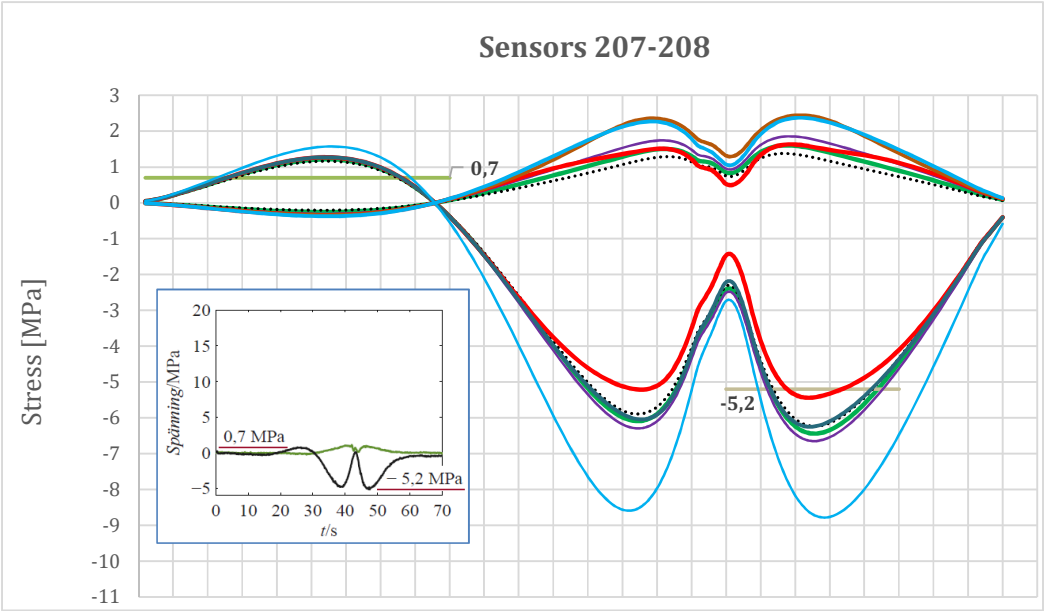


Figure 5-15 Influence lines for sensors 207-208.

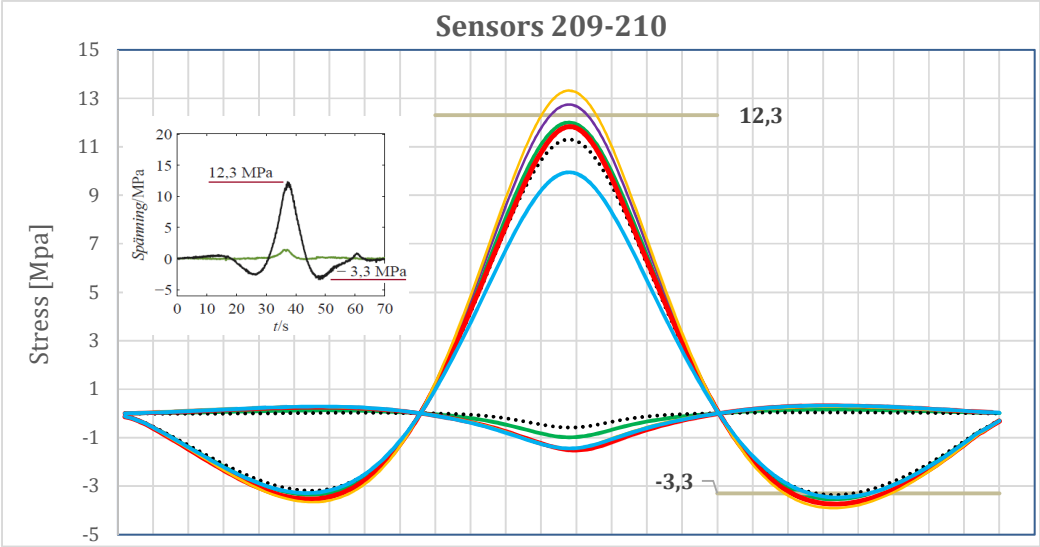


Figure 5-16 Influence lines for sensors 209-210.

All of the obtained curves were compared by calculating the mean square errors for the extreme values from the graphs. The results of these calculations were compiled in Table 5-1

Table 5-1 Mean square errors (MSE) for various studied finite element models.$\bar{e}$ at the end of each column indicates the mean of MSE for that column. For the sensor locations, see Figure 5-3.

Sensor location	Absolut measured value	Case 1		Case 2a		Case 2b		Case 3a		Case 3b		Case 3c	
		Absolut calculated value	$(x_{mod}-x_{real})^2$	Absolut calculated value	$(x_{mod}-x_{real})^2$	Absolut calculated value	$(x_{mod}-x_{real})^2$	Absolut calculated value	$(x_{mod}-x_{real})^2$	Absolut calculated value	$(x_{mod}-x_{real})^2$	Absolut calculated value	$(x_{mod}-x_{real})^2$
201	0,400	1,050	0,423	1,070	0,449	1,070	0,449	1,020	0,384	1,040	0,410	1,060	0,436
202	1,800	1,950	0,023	1,850	0,003	2,150	0,123	1,750	0,003	1,780	0,000	1,800	0,000
202	1,900	2,050	0,023	1,850	0,002	2,200	0,090	1,860	0,002	1,790	0,012	1,920	0,000
202	7,800	9,800	4,000	8,600	0,640	8,400	0,360	8,840	1,082	8,600	0,640	8,140	0,116
203	0,500	1,890	1,932	1,800	1,690	1,760	1,588	1,600	1,210	1,650	1,323	1,720	1,488
204	3,000	2,880	0,014	2,900	0,010	3,050	0,002	3,020	0,000	2,950	0,002	3,100	0,010
204	3,000	3,040	0,002	2,850	0,023	2,900	0,010	2,880	0,014	2,840	0,026	2,960	0,002
204	16,800	18,890	4,368	15,714	1,179	20,900	16,810	16,280	0,270	15,700	1,210	15,020	3,168
205	1,700	1,800	0,010	1,750	0,003	1,600	0,010	1,780	0,006	1,890	0,036	1,840	0,020
206	10,400	13,300	8,410	12,160	3,098	14,450	16,403	11,730	1,769	11,200	0,640	10,670	0,073
206	5,000	6,950	3,803	6,130	1,277	8,040	9,242	6,400	1,960	6,020	1,040	5,200	0,040
207	0,700	1,510	0,656	1,890	1,416	1,300	0,360	2,000	1,690	1,900	1,440	1,750	1,103
208	5,200	5,420	0,048	6,730	2,341	3,600	2,560	7,260	4,244	6,900	2,890	6,600	1,960
209	1,500	1,450	0,003	1,460	0,002	1,470	0,001	1,490	0,000	1,480	0,000	1,520	0,000
210	12,300	13,320	1,040	11,830	0,221	9,950	5,523	12,900	0,360	11,600	0,490	10,670	2,657
210	3,300	3,900	0,360	3,920	0,384	3,470	0,029	3,400	0,010	3,600	0,090	3,700	0,160
Mean square error [MPa]		$\bar{e}$=	0,313	$\bar{e}$=	0,223	$\bar{e}$=	0,457	$\bar{e}$=	0,225	$\bar{e}$=	0,200	$\bar{e}$=	0,209

From the presented results, it can be noticed that case 3b yields the best results in comparison to the measured values. The Mean square error equal to 0.20 MPa is close to the real behavior of the bridge, however, manual calibration of the model should be treated more like a guide for automatic calibration, rather than real optimization. These endeavors gave a better understanding of the structure and helped to nominate the parameters which should be further investigated.

From all the presented cases, case 1 and case 3 deserve special attention. From the examination of these two cases, it could be concluded that some assumptions provided better results close to the support, whereas other assumptions were better for the mid-span. This comparison shows how important it is to find a balance in manipulating parameters, otherwise, the model will be misrepresented.
Case 1 seems to provide curves closer to influence lines from sensors 207-208 than other cases. Nevertheless, case 1 fits only well in the support area, in other sensor locations the calculated results deviate from measured values.
The valid conclusion drawn from this investigation is that rotational springs introduced in the supports and between crossbeams and main girders might provide satisfying results and better reflect the real behavior of the bridge. The case with rotational springs was not examined due to the number of possible cases to check. It was decided that this parameter should be investigated in automatic optimization.
Regarding case 4, it was examined that the grade of the concrete has an insignificant impact on the shape of the influence lines. The obtained results showed that the difference is negligible, therefore the decision was made not to include the curves from this case. However, another interesting conclusion can be drawn from this case: The concrete grade does not affect the shape of the influence line. Although, some contribution of the concrete in carrying the load exists since it causes the shift of neutral axis of the composite section towards the upper flange of the girder. Moreover, this statement implies that there is a substantial interaction between concrete deck and steel girders. That is why, it can be claimed that the structure acts more like a composite bridge than steel bridge, what is contrary to design assumption made in the 1930s'.

Case 5 was similar in obtained results to case 4. Change of the E-modulus of the steel to 180 GPa did not provide a significant change in results.

6 Automated optimization of FE model

For the automated calibration of the bridge model, the Nelder-Mead simplex method is used to minimize the error function. This is because of the inability to differentiate the function for the problem at hand. The graphical method could be done but would require a substantial number of function evaluations to get a contour plot to evaluate, provided that the function should be dependent on at the most two parameters. However, from this set of values, the minimum value could have been selected, but it would not be certain that it corresponds to the true minimum of the function, thus additional evaluations would have to be made, and in the end it is uncertain if a minimum has been missed due to insufficient resolution in the created plot.
The automated calibration of the model was done through scripting in the programming language Python, in IronPython 2.7.7, a .NET modified version of the Python programming language. The script was used to access the Application Programming Interface (API) of Autodesk Robot Structural Analysis 2017 (RSA). Along with IronPython 2.7.7, an additional software was required to perform the optimization, this software was Extreme Optimization Numerical Libraries for .NET, for the purpose of the thesis project a trial version of the software was used. This additional software was used for the implementation of the Nelder-Mead optimization function. Originally the plan was to use Python's Scipy module which has the Nelder-Mead optimization function, but due to incompatibility issues, this was not possible.

6.1 The optimization process

In order to have a gradual increase of complexity for the creation of the script as well as to validate the results created by the script. The process was divided into the following three steps, optimization of a simple span beam, a multi-span beam, and finally the actual model. The simpler cases allowed for faster runtimes during the creation and testing of the scripts. The simplicity also allowed for easier error searching. Regardless of the case, the principal target was the same, namely the creation of a target function that could be used as input for the Nelder-Mead optimization procedure. The target function had one strict requirement that the output should be a scalar value. The target function for all of the cases took the form of the root mean squared error, computed between the measured and computed values.
The scripts can be found in the appendices,

- F, for the bridge model
- G, for the single-span beam
- H, for the multi-span beam.

6.2 Simple span optimization process

The simple span optimization was essentially a two-step process, firstly the creation of the beam model along with the production of the true results, and secondly the optimization using the Nelder-Mead function.
The creation of the simple span beam was done through scripting in IronPython and thus used the RSA's programming interface to create the different elements, nodes, and boundary conditions. The creation required predefined parameters that, for instance, defined the beam, the node locations, the cross-section, the support configuration, the load and so forth. The parameters that were to be optimized in this case were the rotational and vertical spring constants that were assigned to the supports. These springs

were given some arbitrary values. For the arbitrarily created load, the calculation was run and the internal forces, for a subset of bar elements, were saved to an external CSV file. The target function evaluated the error for the forces, axial and bending, in the beam, between the true case and evaluated cases.
Before the initialization of the optimization process, a set of almost arbitrarily selected initial guesses was defined for the support spring constants. The only restriction for the selection was that the guess was to be far from the true values. This was done to test the functionality of the script and the optimization function. These initial guesses were then used as input for the optimization.
The target function's purpose was to apply the guesses to the model, initiate the calculation, extract the values that were to be compared, compare these values to the true values and calculate the error. The scalar-valued error was then the output of the target function, which was returned to the Nelder-Mead function. The Nelder-Mead optimization then used the output for the different guesses to iterate until the optimization met its target condition.

6.2.1 Single span optimization results

Figure 6-1 shows the single span beam which consists of six bar elements supported on the ends by the springs. The loading of the system consisted of a single point load in the center of the span.

Figure 6-1 *The single span beam, with the element numbering in black and the node numbering in red, the support springs are also illustrated at the edges of the beam.*

The true parameters set for the support springs are defined below. From these values, the initial guesses were set to one-tenth of the original values ascribed to the support springs.

$$k_z = 10 \left[\frac{\text{MN}}{\text{m}}\right],\ h_y = 100 \left[\frac{\text{MN}}{rad}\right] \tag{6.1}$$

Where k_z is the spring constant in the vertical direction and h_y is the rotational spring constant. The results from the optimization are shown in Figure 6-2 and Figure 6-3

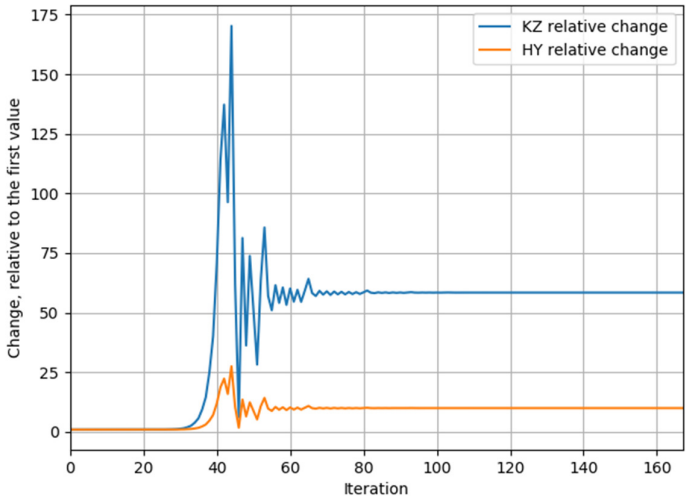


Figure 6-2 The relative change of the optimized parameters, relative to the initial guess.

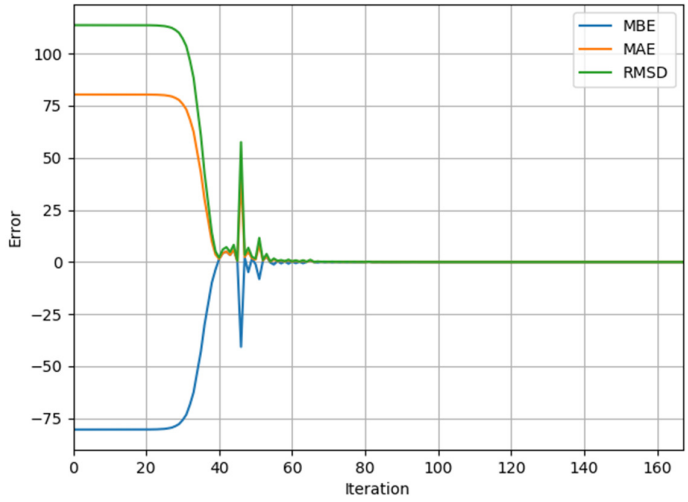


Figure 6-3 The different methods of quantifying errors change, over the optimization process.

The interesting result is that the parameter k_z converges to a value substantially larger than the initial value, 5.8 times larger, whereas the parameter h_y ends up at the true value. Despite this discrepancy, the calculated error does approach zero, for all of the three methods of measurement. A further validation was made by comparing the internal forces within the software, between the original model and the optimized model, which showed that the internal forces were indeed correct.

6.3 Multi-span optimization process

The multi-span optimization follows the same two-step process as outlined in chapter 6.2. The creation of a multi-span beam was required for the optimization process to be initiated. Again, the creation was done through scripting. The number of supports and

beam elements per span were defined, after which the beam was created and assigned a cross-section. The supports were created in the same process as for the simple span beam, i.e. with vertical and rotational springs. Load cases were created based on the number of spans created, one for each span, with a point load created in the mid-span. After this, the results were generated for the different load cases and were saved to external CSV files.
Before the optimization was initialized, a poor initial guess was made, with the only restriction that the guess had to be far separated from the true values. The optimization process then proceeded from this guess and continued evaluating until its finish condition was met.

6.3.1 Multi-span optimization results

The multi-span beam creation script could easily be modified to create a beam with a different number of spans, number of elements per span and length per span. The two selected spring stiffness could also easily be changed in the main script. Figure 6-4 shows the half of the evaluated multi-span beam, it was modeled with 50 beam elements and 51 nodes in total. The selected cross section for this test was IPE400.

Figure 6-4 *The multi-span beam, with constant cross-section, node numbers in red and the support-springs. Only half of the beam is shown to give proper resolution and detail. The supports are located at node 1, 11, 21, 31, 41 and 51.*

The true values of the support springs are defined below. From these values, the initial guesses were set to 50% and 200% respectively.

$$k_z = 10\ \left[\frac{\mathrm{MN}}{\mathrm{m}}\right], h_y = 90\ \left[\frac{\mathrm{MN}}{rad}\right] \tag{6.2}$$

The results of the automated optimization are shown in Figure 6-5, Figure 6-6 and Figure 6-7. In this case, the optimization worked as intended and resulted in the same parameters as initially given. Thus, the script works as intended and to further prove this, additional tests were done with other combinations for the initial guess, which all converged to the true values. These values are not shown here since they do not add any additional insight.

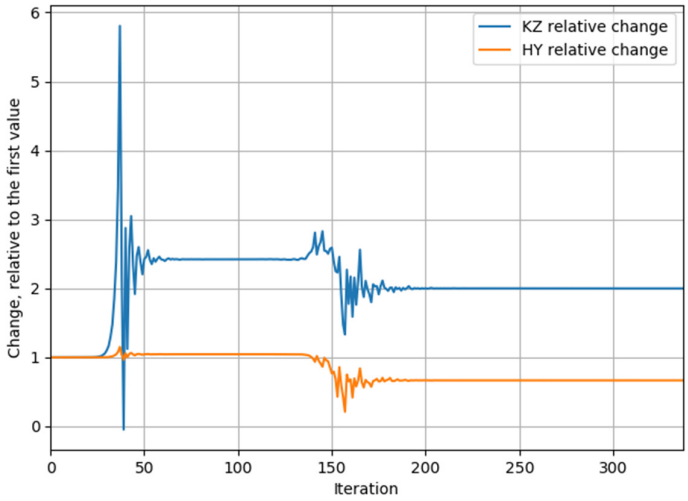


Figure 6-5 The relative change of the optimized parameters, relative to the initial guess.

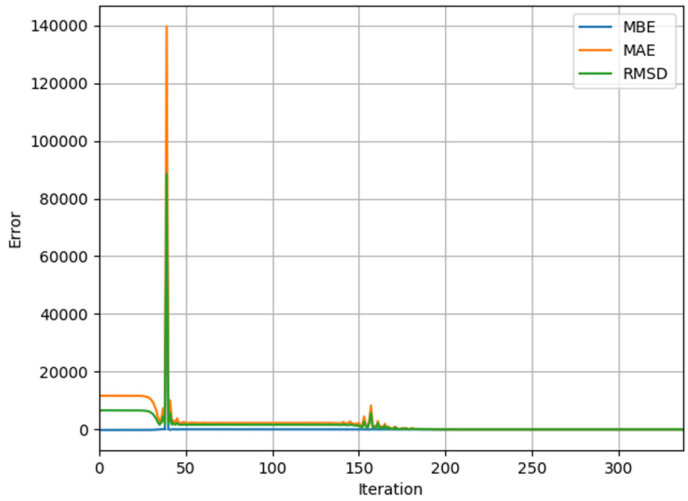


Figure 6-6 The change of the different errors, over the optimization process.

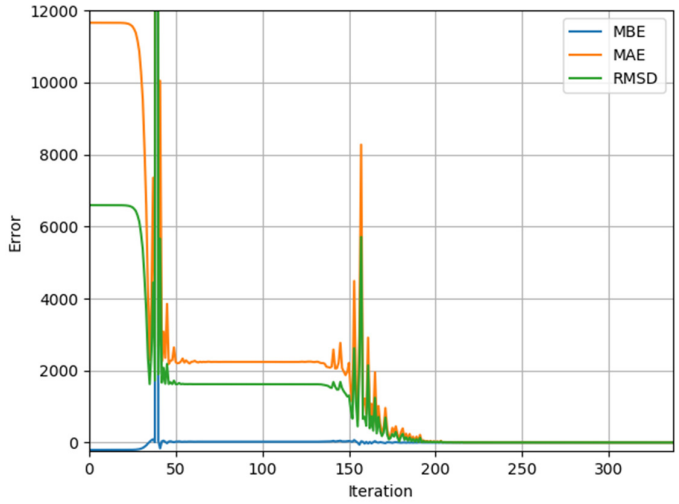


Figure 6-7 The change of the different errors, over the optimization process, but cropped for better resolution.

6.4 Bridge model optimization

The process for the bridge model was slightly different because the model already existed. However, it required some manual modification prior to being subjected to the optimization script. Firstly, the parameters needed to be investigated in order to see which were most influential for the behavior of the bridge. The most influential ones were then to be evaluated in the optimization process. This manual calibration also gave a better initial guess for the optimization script. The parameters of interest needed to be grouped, so that they could easily be altered by the script, this grouping process was also done in the manual calibration. For more details about the manual calibration, see Chapter 5. The parameters chosen for the automated process are given in Table 6-1:

Table 6-1 The parameters in the initial guess used in the automated calibration.

Parameter:	Initial guess
Main beam offset	1.57 $[m]$
Transversal beam offset	0 $[m]$
Rotational springs connecting the transversal beams at intersections with the longitudinal beams	10 $\left[\frac{\mathrm{MN}}{rad}\right]$
Rotational springs connecting the longitudinal beams at the support	10 $\left[\frac{\mathrm{MN}}{rad}\right]$

The loading used for the automated calibration process was based on the calibration run, which was conducted by KTH (Leander et al., 2015). The calibration run was two trucks with known axle weight traversing the bridge. There were essentially two ways that the automated calibration could be conducted, either the entire run of the trucks along the bridge could be used, or a subset of easily defined load cases from this calibration run. Due to consideration of computational time, a subselection of three load cases was made. This sub-selection was when the trucks were in the mid-span of the span with the strain gauges, and the two adjacent spans. This selection resulted in only

three modeled load cases, instead of 160 which otherwise would have been needed to represent the full run of the trucks. This sub-selection of load cases also serves to optimize the model with regard to minimizing the extreme values created by the calibration run.
The target function for the bridge model optimization had essentially the same function as for the two previous trials. However, some modifications were needed for the final process. For the comparison of the values between the model and the measurements, the measured stresses were extracted for the representative load cases modeled in RSA. In addition to these values, the element numbers in the RSA model that represented the location of the strain gauges also had to be identified. This data was compiled in a series of CSV files, one for each load case. Thus, each CSV file contained the following data in each row, bar id, the stress in the top flange and the stress in the bottom flange.

6.4.1 Bridge model optimization results

The result of the optimization of the bridge model is illustrated in Figure 6-8 and Figure 6-9.

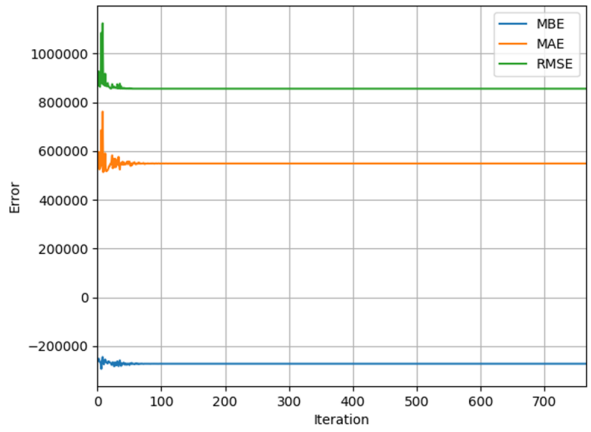


Figure 6-8 The change in the error measuring techniques. The horizontal axis represents the number of iterations, the vertical axis represents the error in Pascals.

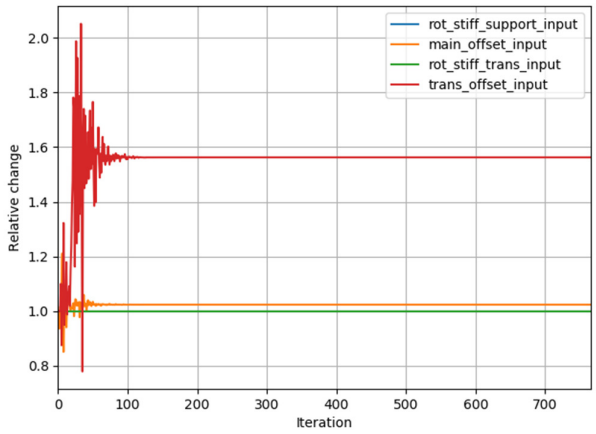


Figure 6-9 How the different parameters change over the iterations

The results of the optimization are summarized in Table 6-2. There is some improvement regarding the error RMSE, which the optimization script was set to optimize for. But for the other two error evaluation methods there was a reduction in accuracy.

Table 6-2 The results of the error measurements with regards to the first iteration, i.e. the starting condition, and the last iteration, i.e. the accepted solution.

Error estimator	First iteration, values in MPa	Last iteration, values in MPa
Mean bias error (MBE)	-0.260	-0.272
Mean absolute error (MAE)	0.533	0.549
Root mean squared error (RMSE)	0.869	0.855

The final parameters based on the automated calibration process is given in Table 6-3. Comparing it to Table 6-1 it becomes evident that only the two beam offsets vary in any significant way and that the results from the FE-model were insensitive to any change in the rotational springs.

Table 6-3 The final parameters from the automated calibration process.

Parameter:	Final guess
Main beam offset	$1.607\ [m]$
Transversal beam offset	$0.563\ [m]$
Rotational springs connecting the transversal beams at intersections with the longitudinal beams	$10\ \left[\frac{\mathrm{MN}}{rad}\right]$
Rotational springs connecting the longitudinal beams at the support	$10\ \left[\frac{\mathrm{MN}}{rad}\right]$

Table 6-4 compares the stresses from the measurements to the values of the FE-model in different stages of the optimization process. As the table shows, the accuracy for the mean value has decreased over the optimization process, but the accuracy of the mean absolute value (MAV) and the root mean square value (RMSV) have improved. Thus, the calibration has improved the model in some aspects.

Table 6-4 Comparison of different mean stress values (stresses from locations 201 to 208) from various sources: measured values, last iteration in the numerical optimization, end result of the manual calibration, and the values from the original model; (MV: mean value, MAV: mean absolute value, RMSV: root mean square value).

Value	Measured, MPa	Last iteration, MPa	Manual calibration, MPa	Original model, MPa
MV	0,51	0,24	0,27	0,42
MAV	2,43	2,42	2,39	2,88
RMSV	4,44	4,21	4,20	4,87

Overall the change is minor, and little was gained from subjecting the model to the automated optimization script. Most of the gain regarding the optimization was achieved in the manual calibration of the model. A more detailed breakdown of the deviations between the stresses calculated in the model and stresses calculated from the strains are presented in Table 6-5,
Table 6-6, and Table 6-7 for load case 1, 2, and 3, respectively.

Table 6-5 Comparison of the stresses derived from the measurements to the calculated stresses from the last iteration, for load case 1. All values are given in MPa.

Strain gauge location	Gauge number, top /bot.	Top strain gauge		Bottom strain gauge	
		Measured	Calculated	Measured	Calculated
Beam C, midspan	201 / 202	0,0	0,0	-1,8	-1,6
Beam E, midspan	203 / 204	0,0	0,0	-2,5	-2,1
Beam G, midspan	209 / 210	0,0	0,1	-0,3	-2,8
Beam E, fourth point	205 / 206	0,0	0,0	-0,4	-0,1
Beam E, support	207 / 208	0,0	-0,2	0,7	1,2

Table 6-6 Comparison of the stresses derived from the measurements to the calculated stresses from the last iteration for the load case 2. All values are given in MPa.

Strain gauge location	Gauge number, top/bot.	Top strain gauge		Bottom strain gauge	
		Measured	Calculated	Measured	Calculated
Beam C, midspan	201 / 202	-0,9	-0,3	7,6	7,9
Beam E, midspan	203 / 204	0,0	-0,3	16,6	14,6
Beam G, midspan	209 / 210	1,5	-0,9	11,8	10,8
Beam E, fourth point	205 / 206	1,1	0,1	2,5	2,9
Beam E, support	207 / 208	0,8	0,9	-4,4	-5,1

Table 6-7 Comparison of the stresses derived from the measurements to the calculated stresses from the last iteration for the load case 3. All values are given in MPa.

Strain gauge location	Gauge number, top/bot.	Top strain gauge		Bottom strain gauge	
		Measured	Calculated	Measured	Calculated
Beam C, midspan	201 / 202	0,0	0,0	-2,0	-1,9
Beam E, midspan	203 / 204	0,0	0,0	-3,0	-2,5
Beam G, midspan	209 / 210	0,6	0,1	-3,3	-3,3
Beam E, fourth point	205 / 206	0,3	-0,1	-5,0	-5,3
Beam E, support	207 / 208	0,7	1,2	-5,2	-6,2

7 Spatial adjustment Factor (SAF)

7.1 The concept of Spatial Adjustment Factor (SAF)

Spatial adjustment factor (SAF) is a concept proposed by (Liu, Frangopol, & Kwon, 2010) in their study of fatigue reliability assessment of retrofitting distortion-induced cracks in steel bridges. In their work, structural health monitoring (SHM) and validated FE model are incorporated to identify the cracking location in the structure. In this paper, Liu and colleagues show a concept of SAF as a suggestion when identified fatigue cracking location in FE model is different than the location of monitoring sensors. The introduced idea gives the possibility to modify measured data using a simple factor to transform results to the point of interest. The approach presented in Liu's article was an inspiration for SAF presented in this thesis. However, the definition of SAF in this study is substantially modified.

The methodology presented by Liu is based on comparing stresses between sensor location and the place where the crack arises. Stress ratio was investigated under static load using FE model. What is important, this ratio is highly dependent on the place where the load is located. In other words, this ratio has its own unique value for the particular position.

The concept presented by Liu was changed and developed to better take into consideration variation of the load and its location. For this purpose, instead of static load, influence lines and moving loads were incorporated. Using the calibrated FE model created in RSA allows for generating a number of reliable influence lines and further comparison of them. Influence lines are treated as stress-time history and were used to create stress histogram using cycle counting. After cycle counting, equivalent stresses were calculated for two different locations. Later, the equivalent stresses and the total number of cycles were compared to obtain two distinct Spatial Adjustment factors: one for the equivalent stresses and one for the total number of cycles. By having SAFs and monitoring data, it is possible to modify measured data to attain stress ranges and number of cycles in a location, where for some reason performing measurements is impossible. For better understanding, Figure 7-1 illustrates the simplified procedure to obtain SAFs.

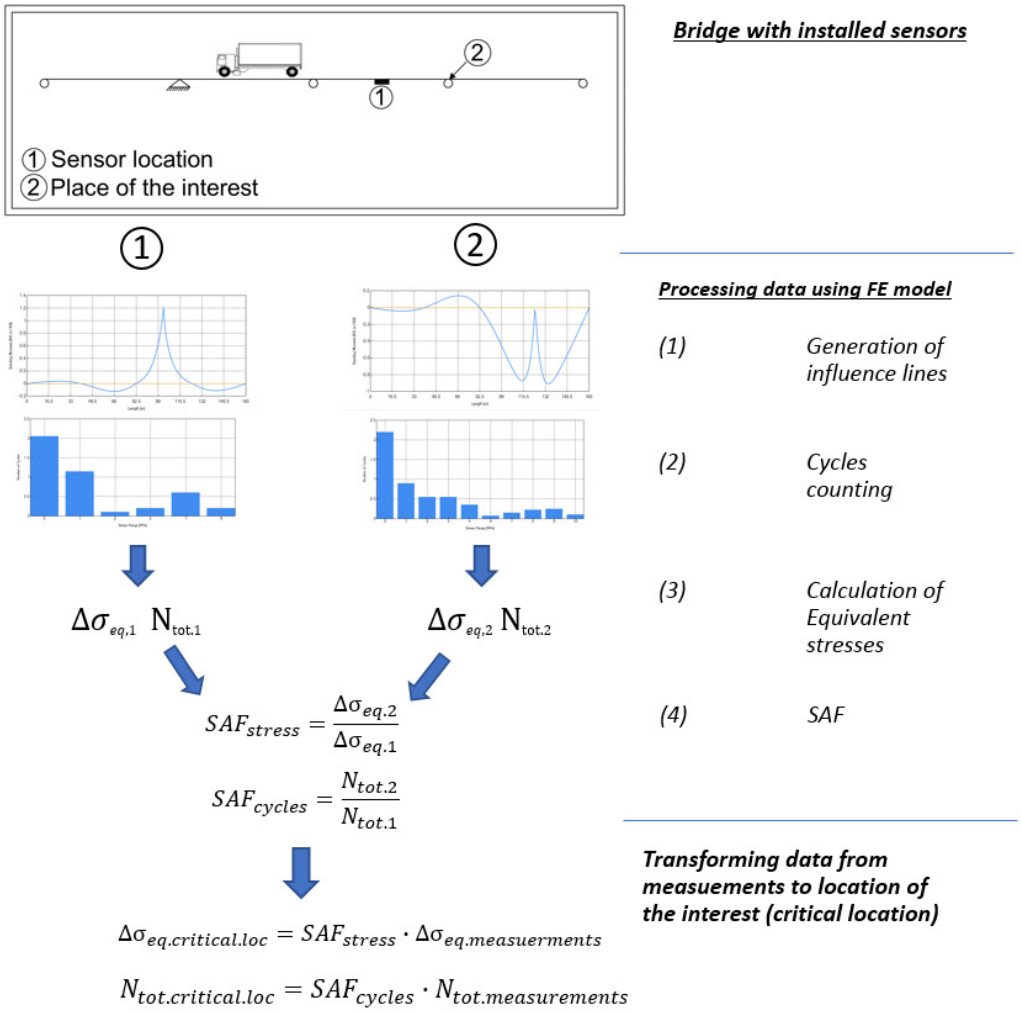


Figure 7-1 Procedure for obtaining SAFs.

7.2 Development of the SAF

To prove the validity of the concept, simplified cases were investigated. Firstly, the correlation between the support and the mid-span were examined. This case was called General Case and it was studied for multi-span beams such as two, three, four-span beams and Göta Älv Bridge model (river part). All of the simplified systems are similar to each other, that means the same beam section was used and length of the spans were equal to 20 m. About the Göta Älv Bridge model, the description can be found in Section 5.2.

As the next step of the investigation, a "Specific Case" was studied. Specific Case is similar to General Case; however, the difference is that the spacing between two points of interest is approximately 2 m. It is called Specific Case since it is referring to a case from viaduct part of Göta Älv Bridge, where sensors are located ~1750 mm from the support (cracking location). In reality, installing sensors adjacent to the support region was obstructed by practical limitations, therefore the need for transforming measured

data from one location to another arose. Similar to the previous case, Specific Case was checked for the same models mentioned in the previous paragraph.

Additionally, the case of a one-span beam with different span lengths was examined in more detail in order to better understand how the length of the span affects the SAFs. This case was investigated for 5, 10, 20 and 40 m long beams and correlation was done between mid-span and a point situated at one-fifth of the span length from the support.

7.3 Traffic models

Each of the models was subjected to 7 different traffic load models (load models). Following traffic load models were chosen:

0. Reference traffic - traffic from measurements in Sweden (Carlsson, 2011). The mentioned report contains essential information and guidelines how to proceed with λ method (alternative for fatigue evaluation to damage accumulation method proposed by Eurocode). Contained data provides information about heavy vehicles occurring on the roads in Sweden. Additionally, delivered data specified number of axles, axle's load and percentage of each vehicle in the traffic. This information helped to create traffic model which may be treated as a representative traffic existing on the roads in Sweden.
1. Single axle (100 kN)
2. 4-axle truck (250 kN)
3. FLM 4 – local traffic
4. FLM 4 – medium distance traffic
5. FLM 4 - long distance traffic
6. Random traffic with vehicles up to 12 m long created based on the axles spacing from vehicles used in FLM 4

Fatigue load model 4 is presented in Table 2-1. Table 7-1 and
Table 7-2 presents the types of lorries used for the reference case (0) and the case number 6, respectively. The truck which was used for load model 2 has the same geometry as the truck used in the calibration measurements, see Section 5.1.

Table 7-1 Set of the lorries used for reference traffic (0) (Carlsson, 2011).

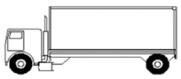
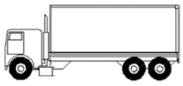
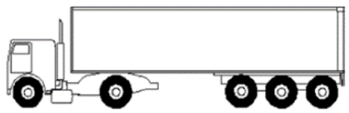
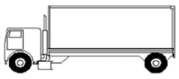
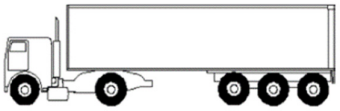
Traffic from measurements			
Lorry type	**Axle spacing [m]**	**Axle loads [kN]**	**Lorry precent age [%]**
	4,5	50	26,2
		84	
	4,2	50	13,5
	1,3	65	
		65	
	3,4	50	6,3
	6	90	
	1,8	40	
		40	
	3,2	50	19,8
	5,2	105	
	1,3	40	
	1,3	40	
	11	40	
	2,7	71	11,6
	1,3	50	
	5,2	50	
	1,3	50	
	1,3	50	
		50	
	2,6	75	19,3
	1,3	80	
	1,3	80	
	5,2	60	
	1,3	60	
	1,3	50	
		50	
	2,6	75	3,2
	1,3	60	
	1,3	60	
	6,2	60	
	1,3	47	
	1,3	47	
	1,3	47	
		47	
	2,6	75	0,1
	1,3	60	
	2,4	60	
	1,3	55	
	6	55	
	1,3	50	
	3	50	
	1,3	30	
		30	

Table 7-2 Set of the lorries used in case 7.

Random traffic with vehicles up to 12 m long			
Lorry type	Axle spacing [m]	Axle loads [kN]	Lorry precentage [%]
	4,5	50	45,5
		84	
	4,2	50	25,1
	1,3	65	
		65	
	3,4	50	9,5
	6	90	
	1,8	40	
		40	
	3,2	50	19,9
	5,2	105	
	1,3	40	
	1,3	40	
		40	

7.4 Comparison of the results

In this section, the results are summarized for the investigation that was carried out. It was decided that the most objective comparison will be shown by means of the relative difference between reference case and each of the other cases. Thorough results pertaining to a total number of cycles, equivalent stresses, and SAFs can be found in Appendices A, B, and C. Conclusions and discussion are provided in the following section.

Table 7-3 to Table 7-5, present the relative differences between resulted SAFs from each of the traffic load models and the reference traffic for the General Case, the Specific Case, and One-Span Beam Case, respectively.

Table 7-3 Relative differences for SAFs between reference traffic and traffic load models 1-6 in the General Case.

		Relative difference					
	Load case	1	2	3	4	5	6
2 Span	n_{tot}	49,0%	23,5%	22,3%	28,4%	28,0%	2,1%
	$\Delta\sigma eq$	7,3%	7,1%	4,0%	19,6%	1,7%	7,4%
3 Span	n_{tot}	34,2%	2,6%	6,9%	10,5%	11,9%	5,6%
	$\Delta\sigma eq$	79,7%	73,1%	77,5%	80,8%	71,9%	67,5%
4 Span	n_{tot}	33,7%	36,8%	13,5%	19,2%	18,3%	9,3%
	$\Delta\sigma eq$	11,0%	3,2%	9,8%	4,6%	3,6%	4,2%
GÄB	n_{tot}	40,0%	22,2%	35,8%	30,1%	26,4%	51,5%
	$\Delta\sigma eq$	8,9%	35,5%	0,9%	1,8%	2,3%	9,6%

Table 7-4 Relative differences for SAFs between reference case and load cases 1-6 in Specific Case.

		Relative difference					
	Load case	1	2	3	4	5	6
2 Span	n_{tot}	29,0%	29,0%	1,8%	9,1%	8,8%	3,2%
	$\Delta\sigma eq$	17,9%	11,7%	7,7%	4,3%	3,6%	7,9%
3 Span	n_{tot}	10,6%	10,6%	21,3%	16,9%	6,8%	12,2%
	$\Delta\sigma eq$	18,1%	3,8%	5,8%	1,8%	16,7%	5,3%
4 Span	n_{tot}	16,9%	16,9%	11,7%	1,2%	0,3%	10,5%
	$\Delta\sigma eq$	16,5%	4,9%	8,6%	6,7%	6,1%	3,2%
GÄB	n_{tot}	7,1%	7,1%	3,7%	6,0%	9,1%	6,9%
	$\Delta\sigma eq$	29,9%	12,9%	4,1%	2,7%	1,8%	0,3%

Table 7-5 Relative differences for SAFs between reference case and load cases 1-6 in One-span Beam Case.

		Relative difference					
	Load case	1	2	3	4	5	6
5 m	n_{tot}	36,9%	26,3%	29,7%	14,6%	8,1%	16,2%
	$\Delta\sigma eq$	13,5%	3,9%	28,2%	28,9%	29,3%	35,0%
10 m	n_{tot}	46,4%	46,4%	1,7%	11,8%	17,4%	5,5%
	$\Delta\sigma eq$	12,7%	29,3%	5,0%	4,9%	6,1%	8,3%
20 m	n_{tot}	46,8%	46,8%	9,2%	9,2%	11,8%	6,5%
	$\Delta\sigma eq$	16,3%	16,0%	8,6%	10,4%	10,3%	6,0%
40 m	n_{tot}	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
	$\Delta\sigma eq$	0,8%	1,5%	4,0%	4,5%	4,6%	1,3%

7.5 Discussion and conclusions about SAF

Trying to evaluate obtained results it may be easily concluded that SAF application is limited with regard to the distance between examined positions. The general Case shows significant spread between results, as opposed to the Specific Case. The second case, where the spacing between the two cross-sections is substantially smaller, yields better results regarding SAFs. This phenomenon can be explained by the different shapes of the influence lines (IL). In other words, when the two locations are close to each other, shapes of the ILs are similar and satisfying correlation may be obtained.

Having a closer look at results from the Specific Case it may be observed that the single axle load case yields the worst results. It can be explained by the fact that single axle is not able to reflect the real traffic existing on the bridge. On the other hand, models from FLM 4 give a satisfying image of existing traffic and the set of the equivalent lorries proposed in FLM 4 can be certainly used for generating SAFs. Even though a 3 span beam yields worse correlation for FLM 4 in comparison to the 4-axle truck, it is still safe to use SAFs since it yields more conservative results. It should be pointed out that SAF for the total number of cycles is higher, compared to the real traffic, and SAF for the equivalent stresses is close to the measurements, therefore the total damage caused by a combination of these two factors should be on the safe side. As a recommendation for use of SAFs in a situation where it is not possible to compare them to real measurements, SAF for a total number of cycles and equivalent stresses should be selected separately (as the most conservative values) from 3 variants of FLM 4.

As an additional matter of investigation, it was checked how excluding stress ranges below the cut-off limit affects SAFs. From this consideration, it was concluded that in each case, equivalent stresses were higher, however, divergence in the number of cycles was significant and SAF for the number of cycles was not reliable. It is also important to mention that due to fairly low-stress ranges for the selected location, for the simplification all of the equivalent stresses were calculated with the S-N curve slope of m=5.

Furthermore, the effect of different span lengths on the SAF was studied. This part of the investigation was carried out only for the one-span beam to better understand this issue. The detailed results are presented in Appendix C. From gathered data, it was concluded that the longer the IL is, the better correlation may be obtained. In case of very short IL (5m), relative differences are significant. This behavior can be caused by the presence of active and inactive axles on the bridge. The situation with active and inactive axles arises when during the passage of a vehicle, it is not possible to place all of the vehicle's axles on the bridge at the same time. In other words, when the vehicle is longer than the bridge, it creates stress cycles that can interfere with each other.

At the same time, surprising is the fact that worse correlation can be observed for case 2 (4-axle single truck) between 5 m and 10 m long beams. Since this phenomenon seems to be counterintuitive, this occurrence was investigated closer. Appendix C includes the detailed stress response, stress histograms and equivalent stresses from passages of the 4-axle single truck from case 2. Closer examination of these cases helps to understand that the short truck (6,45m between the first and the last axle) is closer in its behavior to the single axle than to the set of the lorries representing road traffic. The characteristic feature in the case of one-span beams for the single axle is SAF for the number of cycles

is always equal to one. A similar feature can be observed for moving short truck along influence lines longer than 5 meters. This remark explains why the correlation for longer IL (10 and 20 m) is worse in comparison to the shortest one (5 m) for the 4-axle single truck.

As a recommendation for further study, the impact of span length on the SAFs in multi-span beams and its relation to the maximal spacing between two points of the interest can be considered.

What is more, in the last stage of investigation of SAF, it was concluded that in the future SAF should be considered as a vector corresponding to each stress range, rather than one factor corresponding to equivalent stress. This approach can provide wider applicability of this method and it seems like that this approach can exclude spacing limitation between two points.

The last, but not least remark is that SAF may be a promising concept in the future for transforming measurements from one location to another when spatial limitations are existing. Nevertheless, more investigation needs to be carried out to fully prove the concept.

8 Processing of strain data for fatigue evaluation

This chapter will specify and explain the methodology for processing of a given dataset of strain measurements, to the final result of a defined sum of accumulated damage is explained. A script in Python programming language was developed to carry out the required processing tasks, including temperature-compensation of the recorded strains, rainflow cycle counting, and cumulative fatigue damage calculations.

8.1 Script structure

The structure of the python script used to compute and predict the damage caused to a detail is illustrated as a flowchart in Figure 8-1.

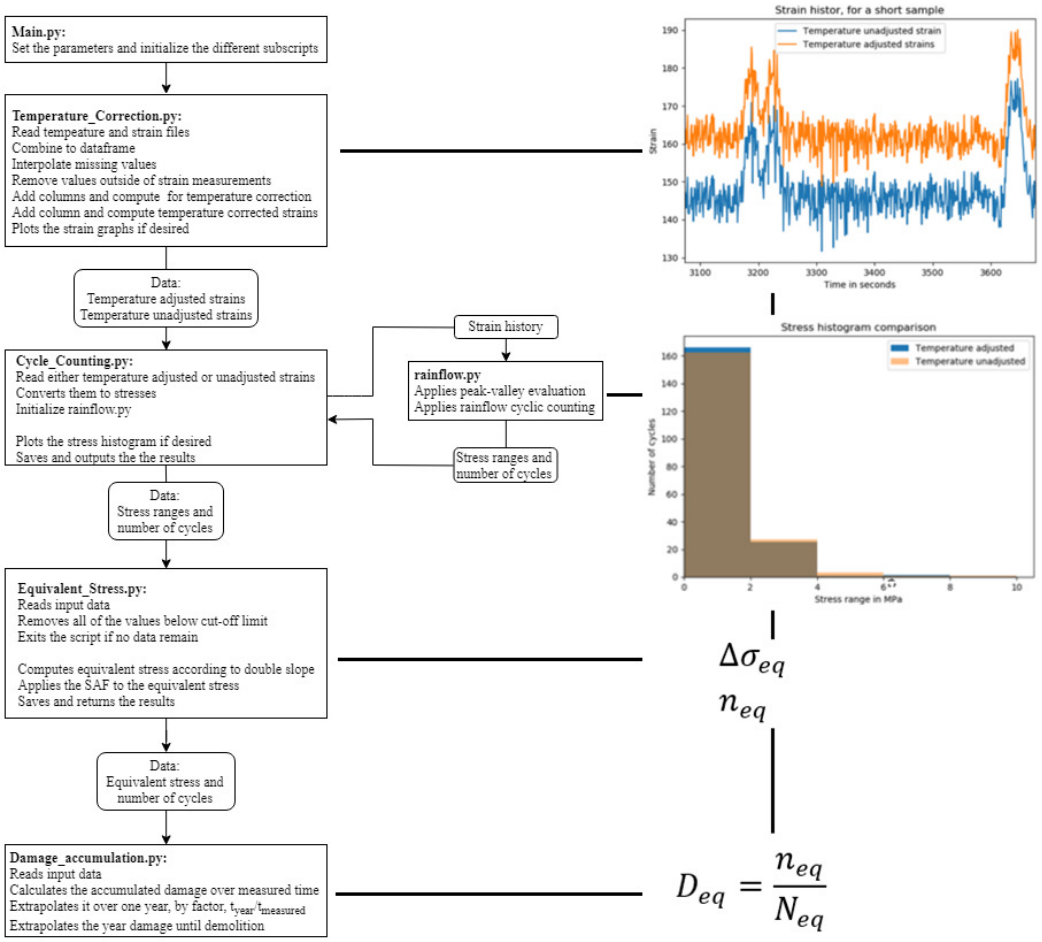


Figure 8-1 The evaluation script in a flowchart, as well as the results of each step.

The script is a multistep process. The first step is the main script, in which all the parameters are set, such as detail category, number of cycles to failure, SAF and so forth. The second part is the temperature correction script, which adjusts the collected strains for temperature variation. The temperature correction is further detailed in Section 8.2. After the strains are corrected for temperature, the data is sent to the cycle counting script, in which the stress ranges and number of cycles are computed from the

stress history, further expanded upon in 8.3. With the stress ranges and number of cycles computed, the data is sent to the equivalent stress script, in which the script discards the stress ranges below the cut off limit, divides the stress ranges into the two slope regions and proceeds to compute the equivalent stress for the chosen slope angle, further expanded in 8.4. With the computed equivalent stress, the main script sends the data to the damage accumulation calculation script, which proceeds to compute the damage caused during the measured period, which is then extrapolated over one year and then until the demolition of the bridge, further expanded in 8.5.

8.2 Temperature correction

To better explain how the temperature correction is performed, some background information is needed. The strain gauges collect data on the change in electrical resistance between two points, this resistance is then converted to strains. However, the resistance measured is affected by temperature variations, thus also the resulting strains. The correction of the collected strains is done in a two-step process. The first part of the correction has to do with the correction of the thermal output, the change in resistance of the strain gauge as the temperature varies. The second part has to do with the gauge factor, which is the correction of how the relation between the resistance and the strain varies (VPG Sensors, 2014). The temperature correction used in this thesis is based upon the manufacturer's guidelines.
The adjustment for the thermal output is done through the subtraction of a thermal output correcting strain. This thermal output correcting strain is given derived by the polynomial:

$$\varepsilon_{TO} = A_0 + A_1 \cdot T + A_2 \cdot T^2 + A_3 \cdot T^3 + A_4 \cdot T^4 \tag{8.1}$$

Where A_{index} are strain gauge specific constants and T is the temperature at the time of the measurement. With this correcting strain, the strain corrected for thermal output can be computed as follows:

$$\varepsilon_{Corrected\ for\ TO} = \varepsilon_{measured} - \varepsilon_{TO} \tag{8.2}$$

The adjustment for the gauge factor is done through the application of a factor to the strains corrected for thermal output. The factor is computed by:

$$k_{Gauge\ correction} = \frac{2}{k_{Gauge\ factor} \cdot \left(1 - TC \cdot \left(\frac{T - 24^oC}{100^oC}\right)\right)} \tag{8.3}$$

Where $k_{Gauge\ factor}$ is the gauge factor specific to each produced batch of strain gages, TC is the change of the strain gauge's conversion factor between resistance and strain, in percent per 100 degrees Celsius, and T is the temperature.
When the corrections are combined, the corrected strains are computed by the following expression:

$$\varepsilon_{Temperature\ corrected} = \varepsilon_{TO} \cdot k_{Gauge\ correction} \tag{8.4}$$

8.2.1 Temperature correction validation

The temperature correction validation process given below utilizes the final version of the script, which appears illogical since the full process of the script has not yet been

defined and specified. But the workings of the script are not needed to understand the validation process.

8.2.1.1 Temperature approximation

One issue was that temperatures were not recorded at the time of strain measurements in the viaduct part. In addition, the temperature collection at the river part was not operational during the extended data collection as well. Thus, an approximation of the temperature was made. The approximation was that the temperature in the girders was the same as in the air temperature measured in Gothenburg. To evaluate this approximation, a comparison was made of the last month worth of measurements in Gothenburg and in the bridge girder. The measured air temperature was collected by (SMHI, 2018). The comparison is shown in Figure 8-2. The temperature in the girder and in the air follow roughly the same pattern. The two measured temperature curves have a better agreement for the peak temperature but differ by a value of approximately 5℃ for the local minimum values in the figure.

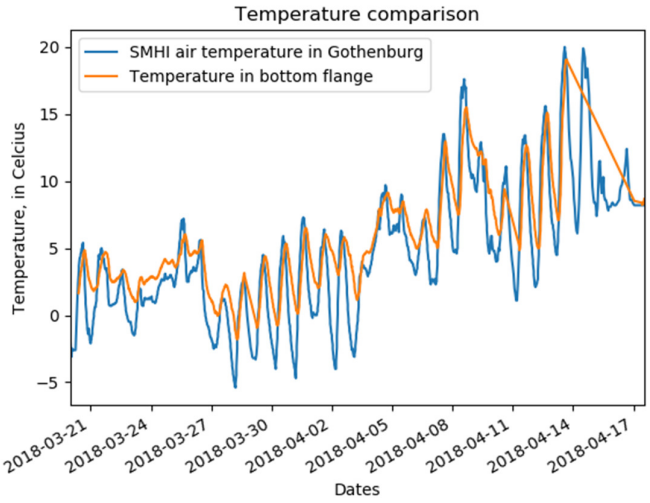


Figure 8-2 Comparison of the temperature in the bottom flange to the air temperature measured by (SMHI, 2018) in Gothenburg.

8.2.1.2 Temperature correction validation, based on measurements

To validate the temperature correction script, a short sample of measurements was selected. The strain history for this sample is shown in Figure 8-3. As is shown in the figure, the visually determinable difference between the two strain histories is that they are offset from one another, smaller variations are not determinable from visual evaluation. Thus, due to their overall resemblance, they should produce the same stress histogram.

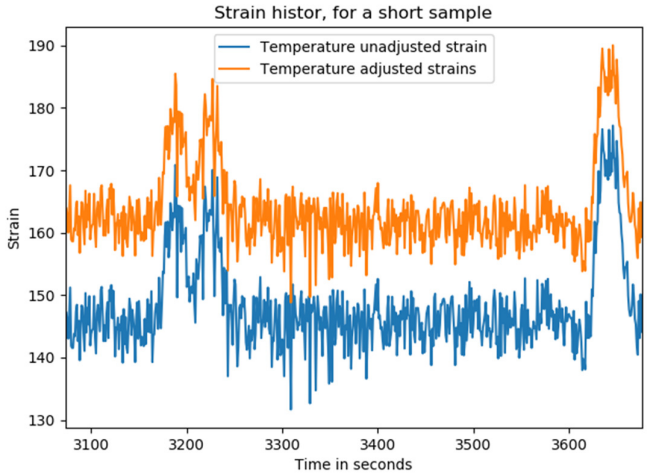


Figure 8-3 A short sample of strain history, illustrating the effect of the temperature adjusted strains, relative to the unadjusted strains.

Applying the stress cycle counting to said strain sample produces the results shown in Figure 8-4. As is shown, the results are almost the same. But, the temperature adjusted stress histogram has slightly more cycles at the lowest stress range, as well as fewer at the higher stress ranges.

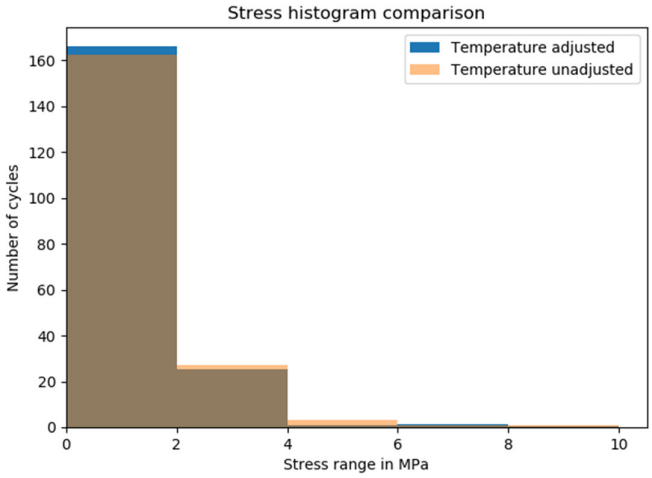


Figure 8-4 Stress histogram, based on the short sample of strain history, comparing the results from the temperature adjusted and unadjusted strains.

8.2.1.3 Temperature correction validation, based on fictive data

As the evaluation based on measured data resulted in some minor differences, further investigation was required. Thus, a fictive strain history was created, such that the resulting stress range would be constant over each cycle, with a value of about 30 MPa. A temperature history was also created, which had a constant temperature of 150^{O}C. The high temperature was set to get sufficient separation between the two plotted strain

histories. The plotted strain histories can be seen in Figure 8-5. Again, there is no visible difference between the plots, aside from them being offset relative to each other.

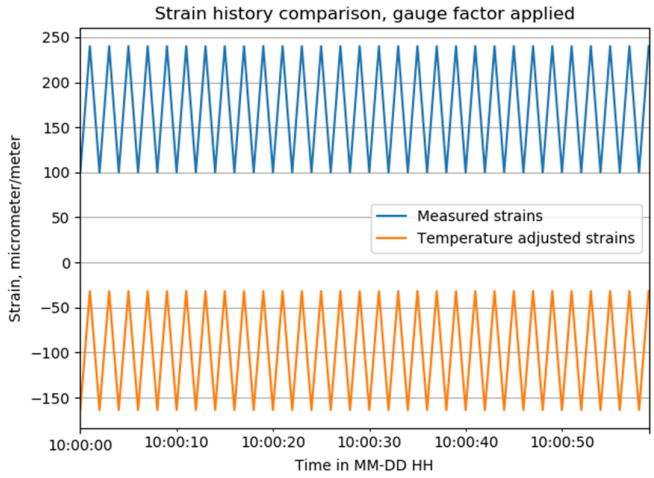


Figure 8-5 Strain history of strain history with constant-amplitude

The result of subjecting the two strain histories to cycle counting is visualized in Figure 8-6, where the stress histograms for both are shown. The difference in results is more easily discernable in this figure, where the temperature-adjusted strains have a lower stress range, but the same number of cycles. The values illustrated in the figure are detailed in Table 8-1.

Table 8-1 Stating the stress range and number of cycles illustrated in Figure 8-6:

	Δσ, MPa	n
Unadjusted	29.4	28.5
Adjusted	27.7	28.5

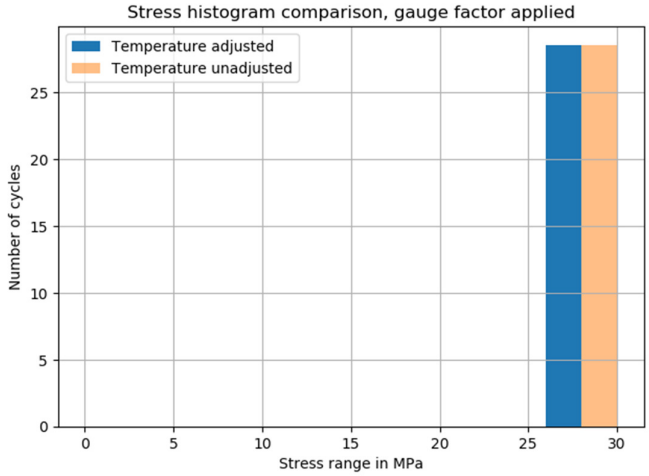


Figure 8-6 The resulting stress histograms for the two strain histories.

The discrepancy, between the two stress ranges, is due to the application of the gauge factor. The thermal output correction factor only serves to offset the two strain histories and thus has no effect on the resulting stress ranges. The gauge factor, however, influences the resulting stresses since it will almost always differ from 1 since it only becomes 1 at 24°C. The proof that this is the factor causing the discrepancy is given below. It is done through the calculation of the gauge factor, using the relationships given earlier along with the strain gauge parameters defined in Table 8-2, and comparing it to the quotient between the stress ranges.

Table 8-2 Parameters used to compute the gauge factor

Parameter	TC	$k_{Gauge\ factor}$	T
Value	1.2	2.155	150

$$k_{Gauge\ correction} = \frac{2}{k_{Gauge\ factor} \cdot \left(1 - TC \cdot \left(\frac{T - 24^oC}{100^oC}\right)\right)} = 0.942 \tag{8.5}$$

$$\frac{\Delta\sigma_{Adjusted}}{\Delta\sigma_{Unadjusted}} = \frac{27.7}{29.4} = 0.942 \tag{8.6}$$

Thus, this proves the authenticity of the temperature correction procedure, through the determination of where the discrepancy appears.

8.3 Cycle counting

The script for the cycle counting first converts the strains, either only from the adjusted strains or both the adjusted and unadjusted strains to stresses.

$$\sigma_i = \varepsilon \cdot E_{steel} \cdot 10^{-6} \cdot 10^{-6} = \frac{\varepsilon \cdot E_{steel}[MPa]}{10^{12}} \tag{8.7}$$

The first division by ten to the power of six is to convert the strains from the unit of microstrains (μm/m) to strain (m/m), and the second division is to convert the stresses from Pa to MPa. In the above relation $E_{steel} = 210 \times 10^3 MPa$.
Using the newly converted stress history, the script sends the data to the rainflow counting script which computes the stress ranges and corresponding number of cycles for either one or both stress histories. The script then plots the stress histograms, if desired.

8.3.1 Rainflow counting script

The rainflow counting was implemented through the use of the module "rainflow.py" (Janiszewski, 2016). The script is done according to the rainflow counting specified in (ASTM, 2011). To validate its authenticity, it was tested against a prescribed load history specified in the said standard, defined in Table 8-3.

Table 8-3 Prescribed stress history used to test the rainflow counting script.

Step	1	2	3	4	5	6	7	8	9	10	11
Stress	-1	-2	1	-3	5	-1	3	-4	4	-2	-1

The test against the prescribed load history yielded the same result as in the specified standard, shown in Table 8-4.

Table 8-4 Cycle counting result of the fictional stress history

Stress range	3	4	6	8	9
Cycles	0.5	1.5	0.5	1	0.5

In addition to the comparison to the values from the standard, it was tested against a self-made script for cycle counting, based on the principles outlined in (Amzallag et al., 1994) and thus according to the flowchart illustrated in Figure 2-9. The scripts yielded the same results, but the script created by Janiszewski was considerably more efficient regarding computational time.

8.4 Evaluation of equivalent stress

The equivalent stress calculations are based on the final equations given in Section 2.3.2.2. How the script works is illustrated in the flowchart in Figure 8-7.

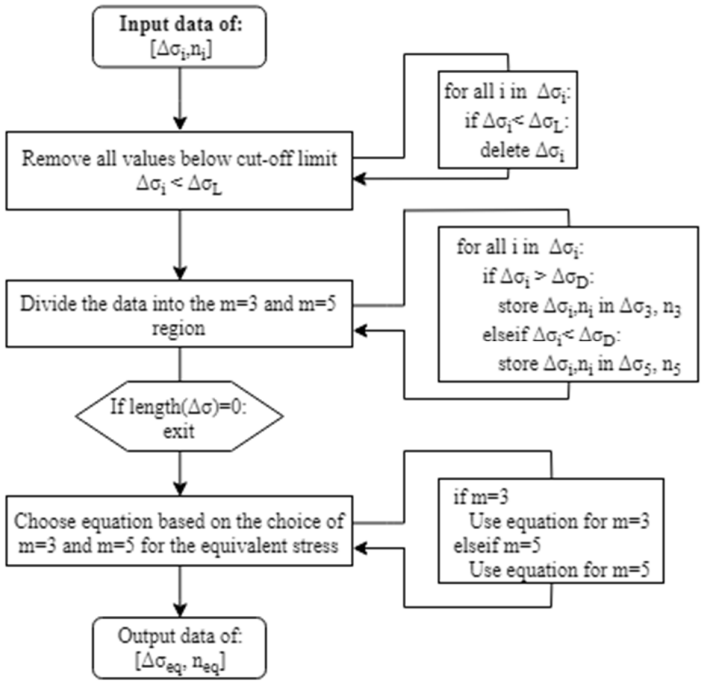


Figure 8-7 A flowchart of how the equivalent stress is computed.

8.5 Damage accumulation

The damage accumulation part of the script is short. The assumption is made that the damage accumulated during the measurement period is representative and can be used for an extrapolation over one year. The yearly extrapolated data is then extrapolated

over the number of years that remain until the demolition of the bridge. The process is visualized in Figure 8-8.

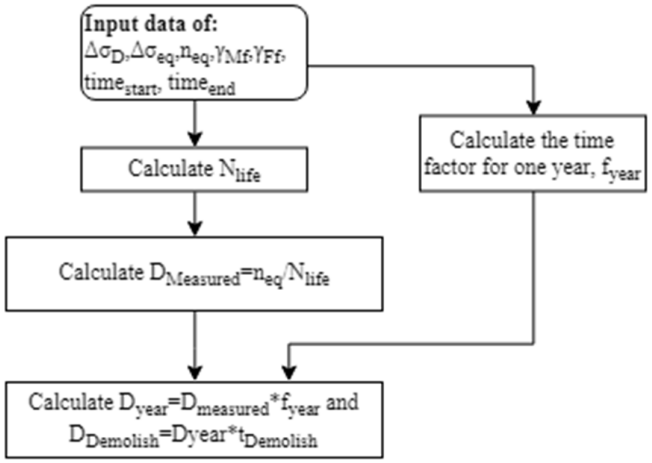


Figure 8-8 The process by which the script computes the equivalent damage.

9 Fatigue evaluation of Göta Älv Bridge

The fatigue assessment was carried out for two details existing in the bridge. Both details are similar to each other because they are connections between the longitudinal beams and crossbeams. Fatigue life of both details is governed by normal stresses.
The first detail is located in the mid-span between axes III and IV (river part), look at Figure 5-3.
The second detail is located above the support (southern viaduct - the intersection of the axes E-S13, as seen in Figure 5-2. Visualization of both details can be seen in Figure 9-1 and Figure 9-2. What is more, contrary to the connection in the mid-span, in the detail above the support, additional cover plates exist to maintain the continuity of the longitudinal beams.

In case of the first detail, measurements were carried out directly in the point of interest, unlike to the second detail where sensors were placed 1750 mm apart from the support. To solve this problem SAF concept was applied to the results obtained from strain-gauge close to the support. It is also important to mention here that the choice of the critical details was limited by the locations of the installed sensors. In fact, the detail from the northern viaduct, where a crack was detected and repaired in the late 1990's, could not be chosen. For the matter of this thesis, Palmgren-Miner's method was chosen for the fatigue evaluation and this it is not applicable if cracks are already detected. In such cases, fracture mechanics analyses should be incorporated, nonetheless, it was out of the scope for this thesis.

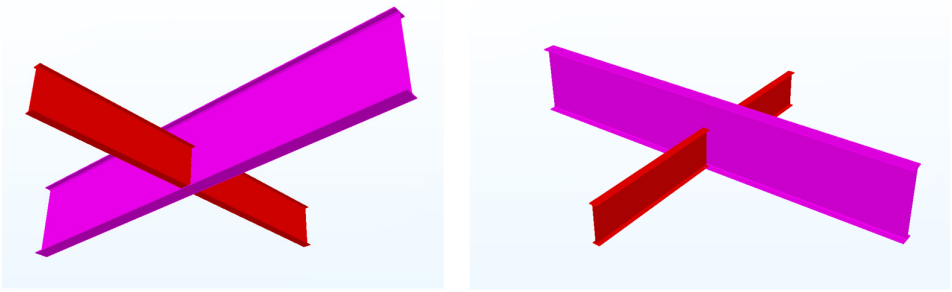


Figure 9-1 Detail number one - the connection between a crossbeam and a main girder in the river part (mid-span).

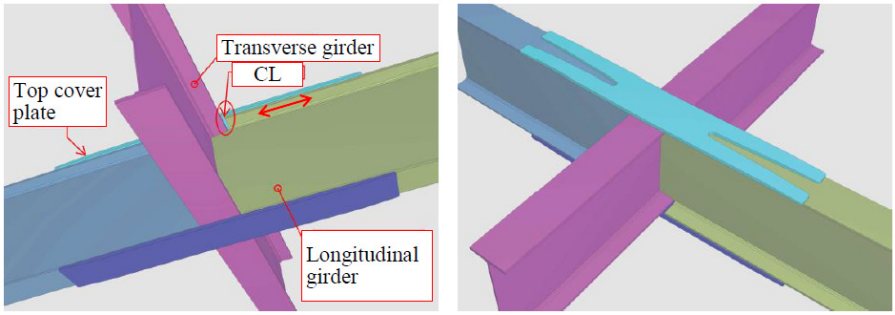


Figure 9-2 Detail number two – the connection between a crossbeam and a main girder southern viaduct (support location). Figure from (Zamiri, 2018).

9.1 Choice of the detail category according to EC

For the detail number 1, representative number 5 from Table 8.4 in EN 1993-1-9 (Figure 9-3) was chosen. This detail category reflects in the best manner the situation existing in the bridge. It corresponds to the fatigue class C40. Nevertheless, this detail category considers current governing welding standards which were not the case when the Göta Älv Bridge was constructed. To consider inferior welding techniques and lower steel quality than used nowadays, it was decided to lower the detail category to C36.

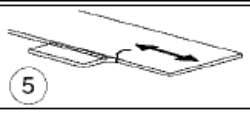
Detail category	Constructional detail	Description
40		5) As welded, no radius transition.

Figure 9-3 Detail 5 from Table 8.4 of EN 1993-1-9.

About detail number 2 the situation is not so straightforward since, in Eurocode, there is not exact detail representing this case. However, the most similar one is the detail 1 from Table 8.5 in EN 1993-1-9 (Figure 9-4). This connection may be compared to the tee joint with the support length greater than 120mm and the plate thickness equal to 28 mm. These characteristics would qualify this detail to C-class 50, nonetheless, in this case, detail category was lowered to C36 to compensate for the welding imperfections, likewise in detail number 1.

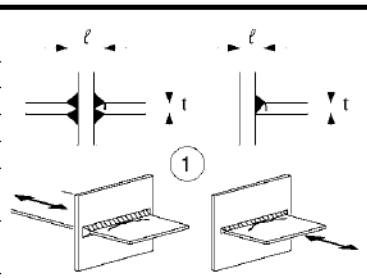
Detail category	Constructional detail			Description
80	ℓ<50 mm	all t [mm]		Cruciform and Tee joints: 1) Toe failure in full penetration butt welds and all partial penetration joints.
71	50<ℓ≤80	all t		
63	80<ℓ≤100	all t		
56	100<ℓ≤120	all t		
56	ℓ>120	t≤20		
50	120<ℓ≤200 ℓ>200	t>20 20<t≤30		
45	200<ℓ≤300 ℓ>300	t>30 30<t≤50		
40	ℓ>300	t>50		

Figure 9-4 Detail 1 from Table 8.5 in EN 1993-1-9.

9.2 Fatigue assessment 1939–2015

The fatigue assessment from 1939–2015 is called past fatigue evaluation. During these years, monitoring sensors were not installed. Therefore, fatigue damage occurred in this period was assessed based on the estimation of historic traffic done by (Larsson, 2004). In his report, Larsson not only estimated the past traffic by review of historical data but also suggested a prediction of how traffic would grow until 2020. Having data like this, it was possible to categorize vehicles types, weight and a total number of passages. This

data helped to create simple yet reliable traffic models, which were used to carry out the fatigue evaluation using Palmgren-Miner method.

Past fatigue evaluation was possible to do using the influence lines generated from the FE model, Figure 9-5 and Figure 9-6, and traffic models from Larsson's estimation. The created traffic models are described in the following section.

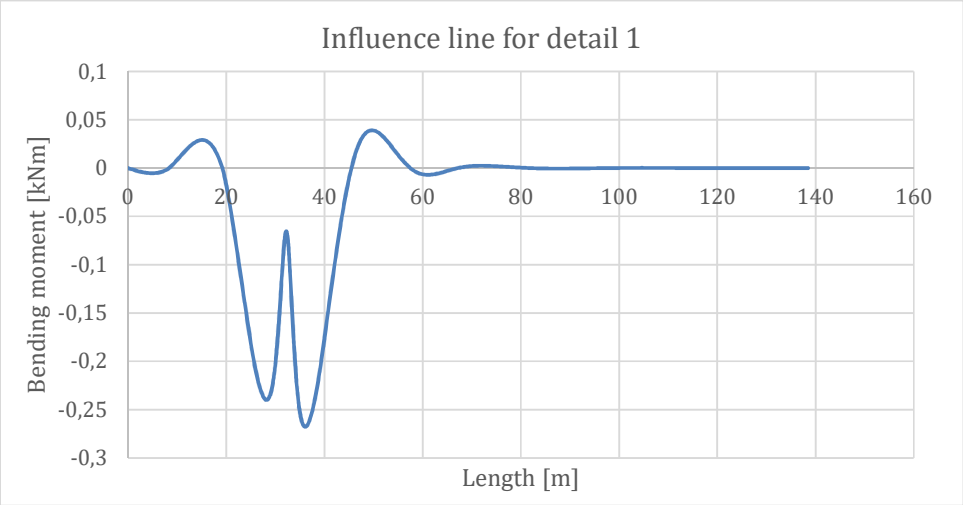


Figure 9-5 Influence line for the location number 1 (southern viaduct above the support).

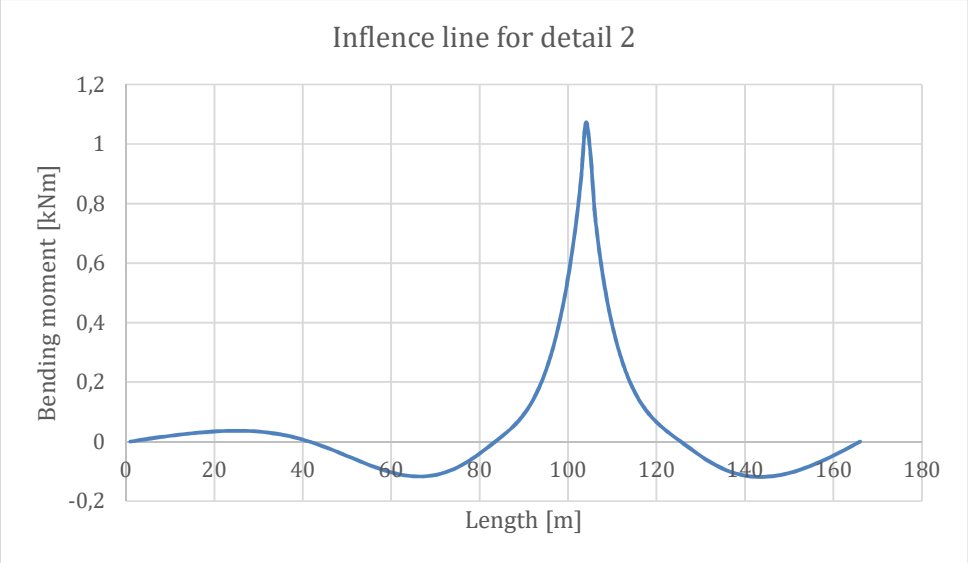


Figure 9-6 Influence line for the location number 2 (river part mid-span).

The procedure for fatigue assessment was presented earlier in Chapter 7, see Figure 7-1 for a summary. The fatigue calculations were done by moving traffic load models through influence lines. In this way, stress-time histories were created, and cycle

counting was possible. Subsequently, the remaining service life was calculated based on the damage accumulation method. Furthermore, the analysis was conducted for characteristics values, γ_{Ff} and γ_{Mf} equal to 1.

9.2.1 Measured traffic loads

The evaluation carried out by Larsson in 2004 on Göta Älv Bridge, summarized the traffic occurring on the bridge during 1939–2020. The results presented by Larsson are somewhat of an estimation for a future, based on the data collected from the past.
The delivered information on specified types of vehicles crossing the bridge, total number of passages and vehicles' gross weights. Raw data was compiled and an average annual number of passages, or average daily passage in case of tramways, for each group were obtained. Summarized results illustrating total load on the bridge in years 1939–2020 are presented in Figure 9-7.

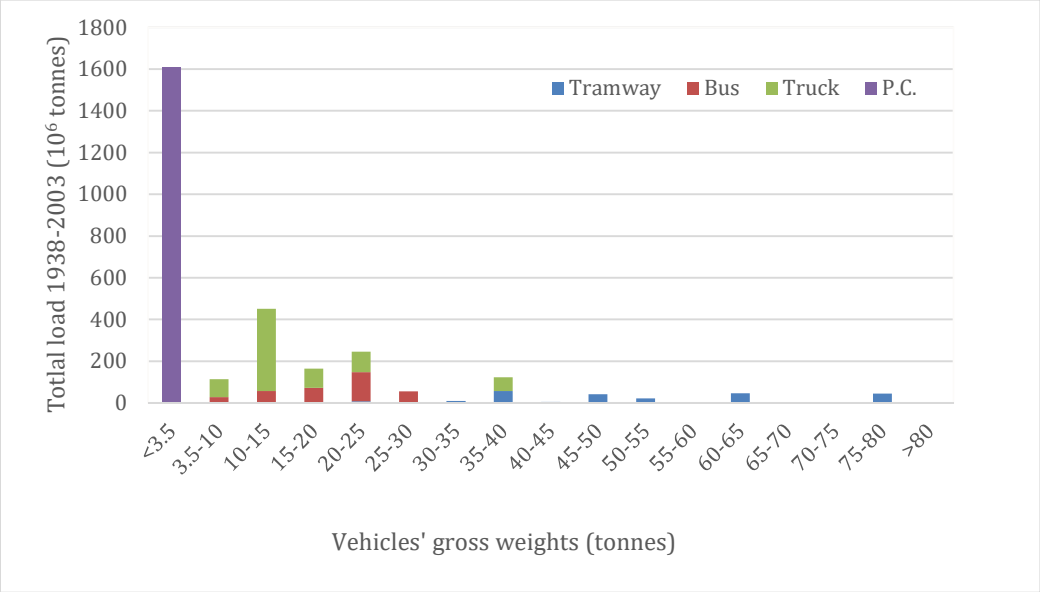


Figure 9-7 Estimated traffic on the bridge, from 1939 to 2020.

Tramway traffic
During the service life of the bridge, various types of trams were in use. For simplifying the fatigue evaluation, three of the most frequent sets of trams were nominated:

- 1x M21
- 1x M31
- 2x M21

With axle weights and spacing between axles are shown in Figure 9-8. To adjust the vehicles' weights indicated in Figure 9-7, each axle weight was multiplied by the ratio of total weight of a tram and maximal nominal weight of a tram. For instance, the weight of each axle for the tram in the first row from Table 9-1 was reduced by the factor 77.5/96=0.81, where values from nominator and denominator correspond to maximal and maximal nominal tram weight, respectively.

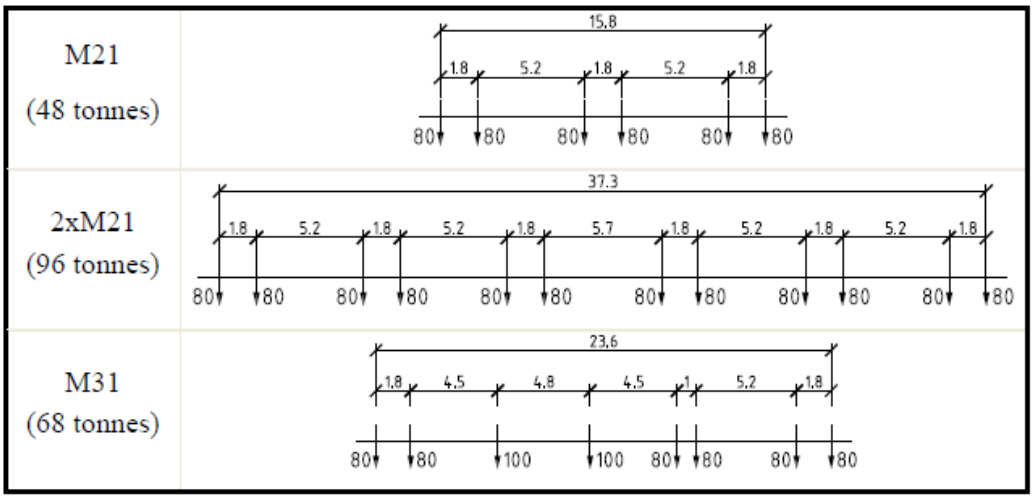


Figure 9-8 Axles' weights and spacing between axles in trams M21, M31 and 2xM21, (Zamiri, 2018).

Due to the significant dynamic impacts from the tramway traffic, the dynamic amplification factor (DAF) was included. DAFs were calculated for viaduct and river part separately according to article 2.3.3.2.5 from Trafikverket's "Krav" document (2017). The document yields similar results to Annex C of Eurocode EN 1991-2 for dynamic amplification factors. DAF values are as follows and the detailed calculations can be found in Appendix D.

$$DAF_{river\ part} = 1{,}073 \quad \wedge \quad DAF_{viaduct} = 1{,}229 \tag{9.1}$$

Table 9-1 shows the tramway traffic used for the fatigue calculations, annual average daily traffic (AADT) and the fatigue damage index caused by each vehicle. Figure 9-9 and Figure 9-10 show the stress ranges causing fatigue damage by trams occurring on the bridge.

In this point, it is good to mention that calculations of damage accumulation, creating stress ranges, histograms and time histories were supported by a small program called BridgeFAT.

Table 9-1 Tramway traffic used for fatigue evaluation and fatigue damage index.

Tram weight [t]	Average weight [t]	Type	AADT	Total number of trains	Partial damage,d_i	
					Viaduct	River part
75-80	77,5	2XM21	19	547 865	0,678	0,613
65-70	67,5	M31	2	57 670	0,086	0,074
60-65	62,5	M31	24	692 040	0,443	0,385
50-55	52,5	M31	14	403 690	0,081	0,065
45-50	47,5	M31	28	807 380	0,036	0,029
40-45	42,5	M21	4	115 340	0,004	0,002
35-40	37,5	M21	48	1 384 080	0,000	0,000
30-35	32,5	M21	9	259 515	0,000	0,000
25-30	27,5	M21	4	115 340	0,000	0,000
20-25	22,5	M21	10	288 350	0,000	0,000
Total Fatigue Damage				Σd_i	1,329	1,168

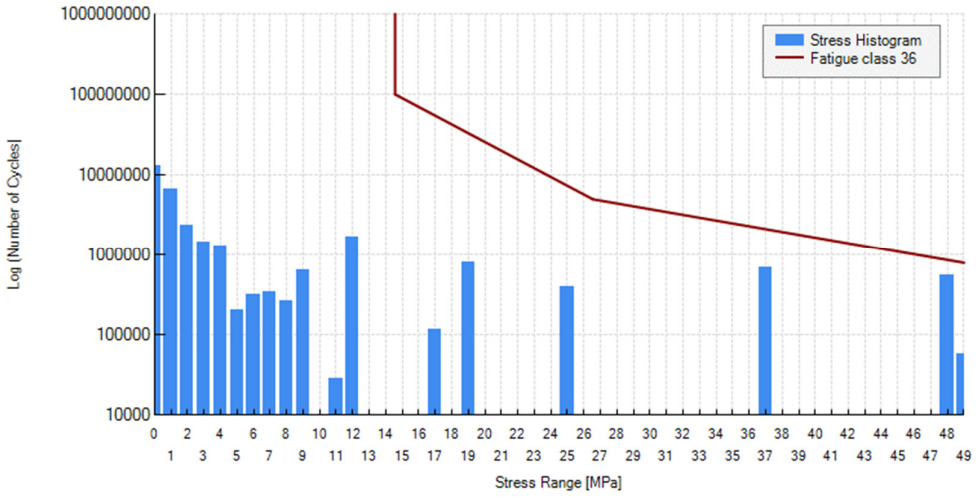


Figure 9-9 Stress histogram for the detail 1.

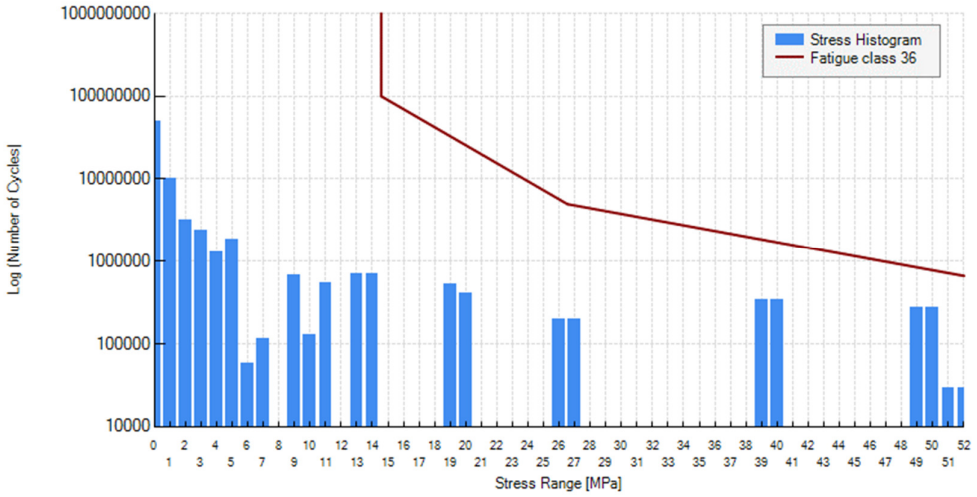


Figure 9-10 Stress histogram for the detail 2.

Looking at Table 9-1, it can be noticed that damage factor already has exceeded 1. According to the assumption of the damage accumulation method, the service life for this bridge is already spent. Moreover, another conclusion drawn from the Table 9-1 is that vehicles with average weight up to 37,5 tons are not contributing to fatigue damage in the bridge, Therefore there is no point in conducting further analysis with road traffic since no bus or truck in the traffic data exceeds this average weight.

9.3 Fatigue assessment based on measurements

Although the results of the past section indicated the theoretical life of the studied details is exhausted, it is still interesting to estimate the amount of the cumulative damage that takes place under real traffic conditions, based on the measured values.

The evaluation of the two details under measured stresses is made for two cases, one with temperature correction and one without temperature correction, as well as for two strain gauges for each detail to better evaluate the authenticity and reliability of the predictions.

9.3.1 Detail 1

The results for detail 2 are produced through the comparison between two strain gauges, which both have the same relative location in the bridge, the gauges being 202 and 204. Both are located on the bottom flange of the main girder in the mid-span but on different girders. The measurements for these two strain gauges span for about one whole day, more precisely 2018-04-16 14:49:52 to 2018-04-17 13:15:29. For this time, there were no temperatures collected because the thermocouple measurement node was offline. So, the approximation that the temperature in the girders was the same as the air temperature in Gothenburg was made to adjust the collected strains.

9.3.1.1 Strain gauge 202

The strain history for strain gauge 202 is shown in Figure 9-11. The most obvious change is the offset of the temperature adjusted strains, relative to the measured strains. Aside from this, it is difficult to discern any other change between the strain histories.

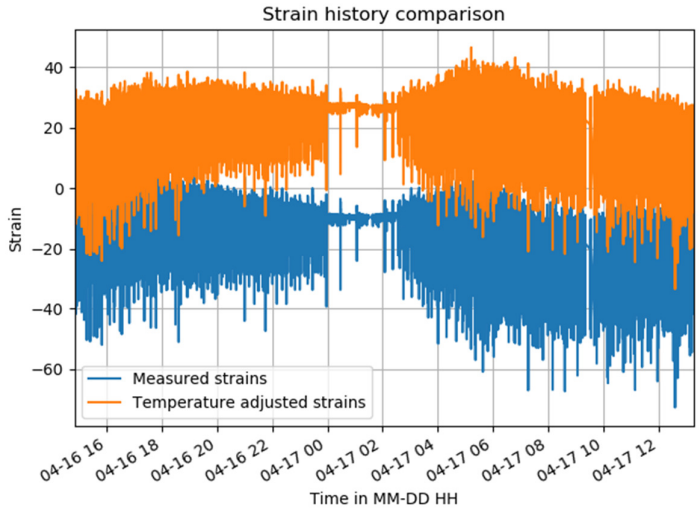


Figure 9-11 Strain history for strain gauge 202, for both the temperature adjusted and unadjusted strains.

Regarding the stress range histogram for strain gauge 202, shown in Figure 9-12, Figure 9-13 and Figure 9-14, there are some differences between the two stress range histograms. For the higher stress ranges, there are more cycles for the unadjusted strains, relative to the temperature adjusted strains. However, most of these values are below the cutoff limit, which for this detail is 14.57 MPa. As seen in Figure 9-14, the remaining stress cycles after the cutoff limit are the same for the adjusted and unadjusted values. The result after calculating the equivalent stress range for the detail is shown in Table 9-2.

Table 9-2 Equivalent stress range and a total number of cycles for stress range histograms based on Temperature-adjusted and unadjusted recorded strains at location 202.

	Strain gauge 202	
	Adjusted	Unadjusted
n_{eq}	1	1
$\Delta\sigma_{eq}$	15.96	16.34

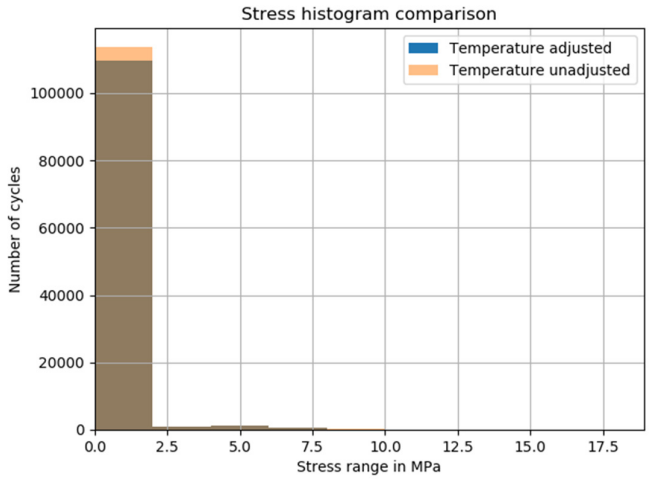


Figure 9-12 The full stress histogram for strain gauge 202.

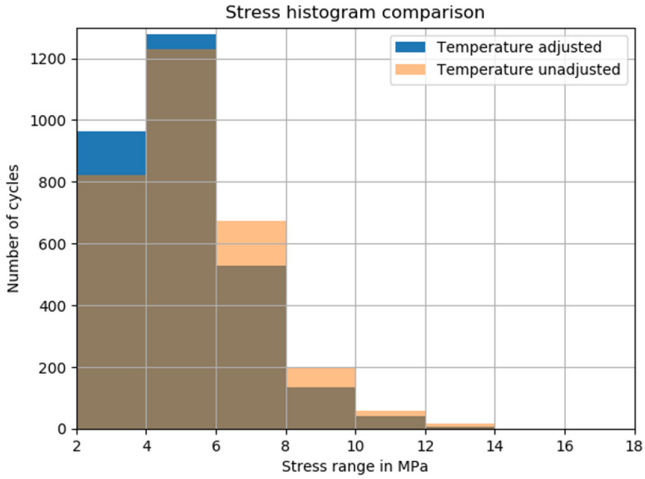


Figure 9-13 Slightly cropped stress histogram for strain gauge 202, from 2MPa to 18MPa

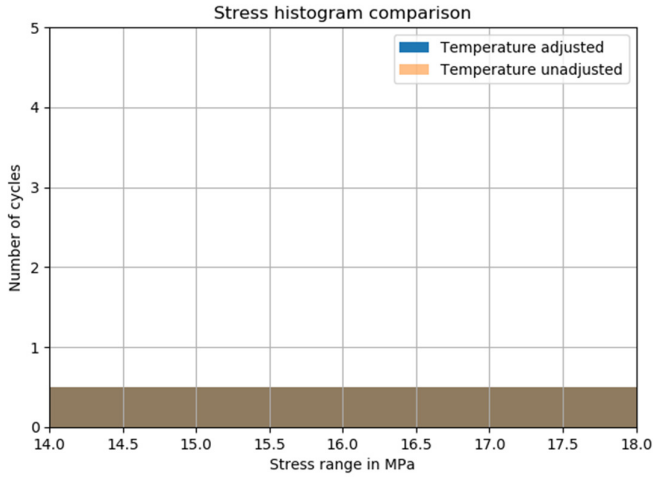


Figure 9-14 Cropped stress histogram for strain gauge 202, from 14MPa to 18MPa.

9.3.1.2 Strain gauge 204

The strain history for strain gauge 204 is shown in Figure 9-15. The most obvious change is the offset of the temperature adjusted strains, relative to the measured strains. Aside from this change, it is difficult to discern any other change between the strain histories.

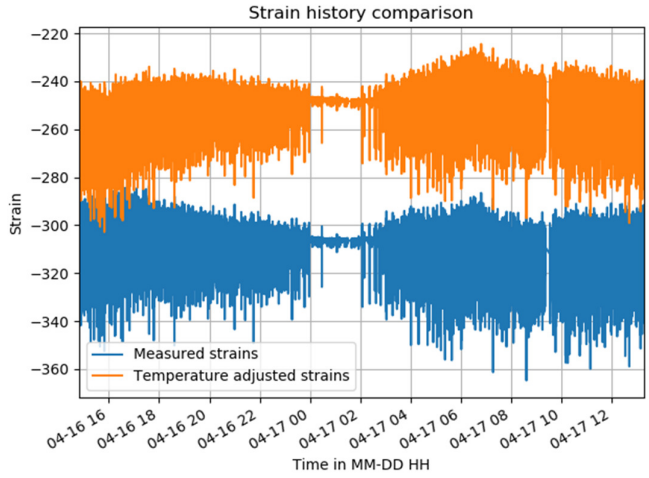


Figure 9-15 Strain history for strain gauge 204, for both the temperature adjusted and unadjusted strains.

The stress range histogram for strain gauge 204 exhibits the same pattern as the strain histogram for strain gauge 202, as can be seen in Figure 9-16, Figure 9-17 and Figure 9-18. The repeating pattern is the reduction of cycles for higher stresses, for the temperature adjusted strains.
However, most of the stresses disappear when the cutoff limit is applied. A large difference is observed between the adjusted and unadjusted strain in this case. For the temperature unadjusted strains, there are substantially more cycles remaining after the

cutoff limit, which is about 14.57MPa, is applied, as can be seen in Figure 9-18. This discrepancy is seen in the results summarized in Table 9-3.

Table 9-3 Equivalent stress range and a total number of cycles for stress range histograms based on Temperature-adjusted and unadjusted recorded strains at location 204.

	Adjusted	Unadjusted
n_{eq}	1	3.5
$\Delta\sigma_{eq}$	16.12	15.31

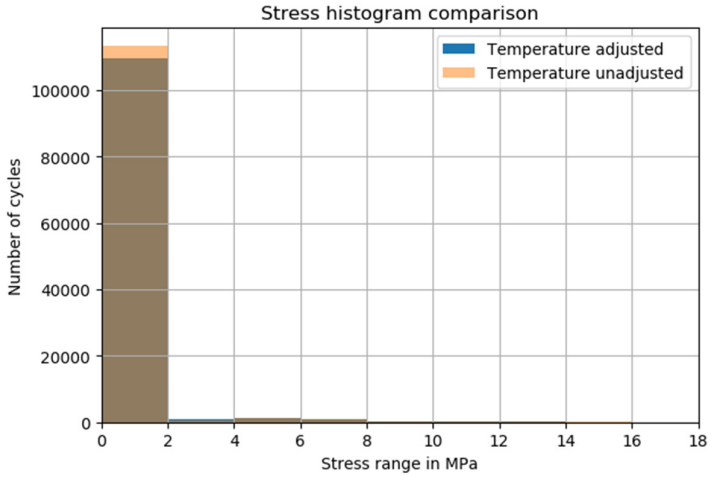


Figure 9-16 The full stress histogram for strain gauge 204.

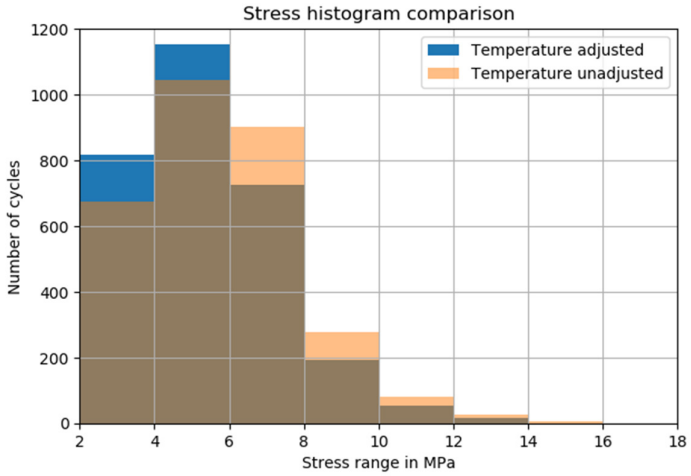


Figure 9-17 Slightly cropped stress histogram for strain gauge 204, from 2MPa to 18MPa.

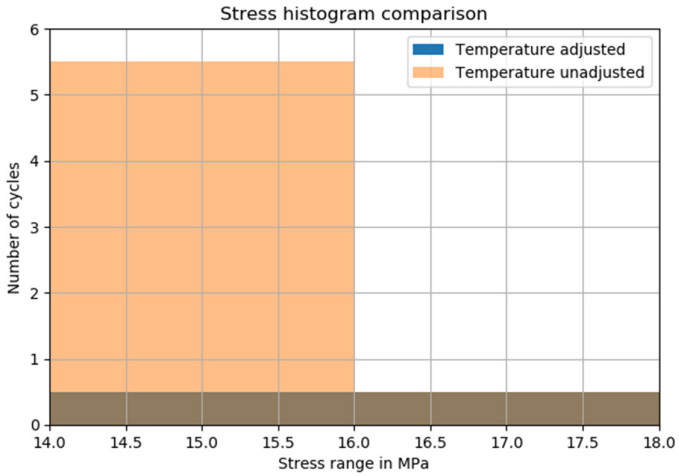


Figure 9-18 Slightly cropped stress histogram for strain gauge 202, from 14MPa to 18MPa

9.3.1.3 Difference between strain gauge 202 and 204

In Table 9-4 the combined results from strain gauge 202 and 204 are compared, for the temperature adjusted strains as well as for the temperature unadjusted strains. It is interesting to compare the damage caused at the two strain gauge locations since they are located at the same part of the bridge, but on opposite sides relative to the centerline of the bridge. Due to this, it can be theorized that the damage should be similar.

Table 9-4 Summarized result of the calculated damage for detail 1, based either on strain gauge 202 or 204, as well as either temperature corrected or uncorrected strains.

	Strain gauge 202		**Strain gauge 204**	
	Adjusted	Unadjusted	Adjusted	Unadjusted
n_{eq}	1	1	1	3.5
$\Delta\sigma_{eq}, MPa$	15.96	16.34	16.12	15.31
$D_{measured}$	$7.838 \cdot 10^{-9}$	$8.802 \cdot 10^{-9}$	$8.232 \cdot 10^{-9}$	$22.24 \cdot 10^{-9}$
$D_{Year\ equivalent}$	$3.061 \cdot 10^{-6}$	$3.438 \cdot 10^{-6}$	$3.215 \cdot 10^{-6}$	$8.689 \cdot 10^{-6}$
$D_{Dismantel}$	$1.862 \cdot 10^{-5}$	$2.092 \cdot 10^{-5}$	$1.956 \cdot 10^{-5}$	$5.286 \cdot 10^{-5}$

Most importantly, it is observed that the recorded stresses in the bridge part are generally small, such that only one loading cycle above the cutoff limit has occurred during roughly 24 hours of measurements.
For strain gauge 202, there is a slight difference between the result of the temperature unadjusted and adjusted strains. It is slight, however, the equivalent stress is marginally higher for the unadjusted strains but with the same number of cycles. This results in a slightly larger predicted damage for the unadjusted strain.
For strain gauge 204 there is a larger difference between the two datasets. The stress range for the unadjusted strains is lower, but the number of cycles is higher. This causes substantially more damage to be predicted by the unadjusted strains. Regardless of the higher damage caused, both calculated damages are small.

9.3.2 Detail 2

The results for detail 1 is produced through the comparison of two different strain gauges. These strain gauges numbers are 110 and 112. They are located on the same girder, E, but are on different points along it. 110 is about 2 meters from the support, and 112 is located at mid-span.
To generate the results at detail 1, two sets of spatial adjustment factors need to be used, each set specific to each of the two strain gauges. The data used for the calculations was collected from 2015-04-29 09:39:30 until 2015-04-30 03:50:24, during the calibration run conducted by KTH, detailed in (Leander et al., 2015). The data consisted of 6.5 million data points per strain gauge. The measurements are not entirely continuous between the two timestamps, there is a slight discontinuity in the measurements. Furthermore, there was no temperature collection at the time of the measurements, therefore the approximation was made that the girders had the same temperature as in the air. During the measurements conducted by KTH described in (Leander et al., 2015) an additional strain gauge was present. This additional strain gauge was meant to be used for compensating temperature induced strains. But, during the thesis, the decision was made to use the analytical method proposed by the manufacturer. This was decided because no such gauge was implemented when the newer measurements were done. Also, there were gauges collecting temperature at the time of the newer measurements. Thus, the analytical method could be applied in both cases, whereas the compensation strain gauge results could only have been applied to the KTH measurements.

9.3.2.1 Strain gauge 110

The resulting strain history can be seen in Figure 9-19. The strain history exhibits the same behavior as described in chapter 8.2, i.e. that the temperature adjusted curve is offset and flatter. Some discontinuities in the measurements can also be seen in the figure, as well as some measurements that might be faulty.

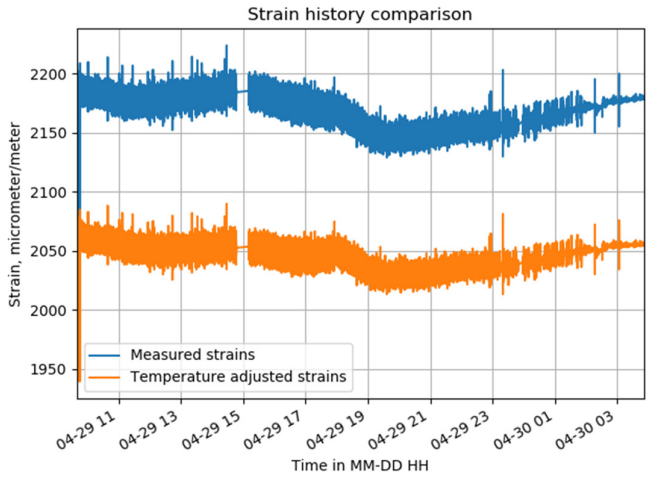


Figure 9-19 Strain history for strain gauge 110, both for temperature adjusted and unadjusted strains

The stress range histogram resulting from the stress history for detail 1 is shown in Figure 9-20, Figure 9-21, Figure 9-22 and Figure 9-23. The different figures represent

different parts of the histogram for better resolution and detail. It exhibits the same pattern as described in Section 8.2; there are fewer cycles at the higher stress ranges for the temperature adjusted strains. The result of converting the stress ranges and cycles to an equivalent stress range is represented in Table 9-5. The reduction of a number of cycles at the higher stress ranges, for the adjusted strains, does indeed have an effect on the number of cycles corresponding to the equivalent stress range, as shown in the table.

Table 9-5 Equivalent stress range and a total number of cycles for stress range histograms based on Temperature-adjusted and unadjusted recorded strains at location 110.

	Adjusted	Unadjusted
n_{eq}	363	523
$\Delta\sigma_{eq}$	20.48	20.61

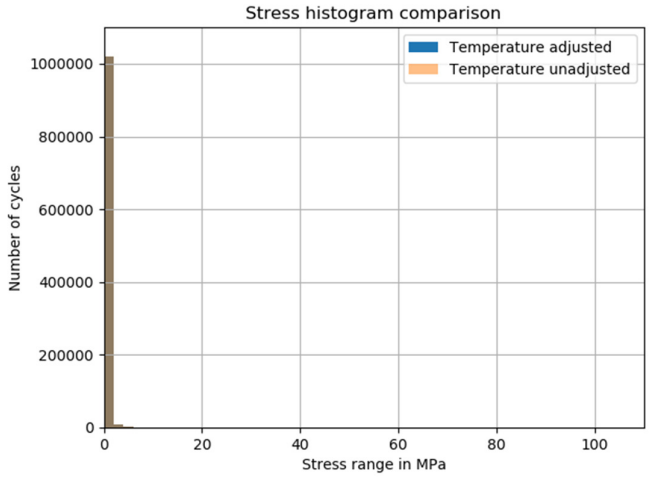


Figure 9-20 The full stress histogram for strain gauge 110.

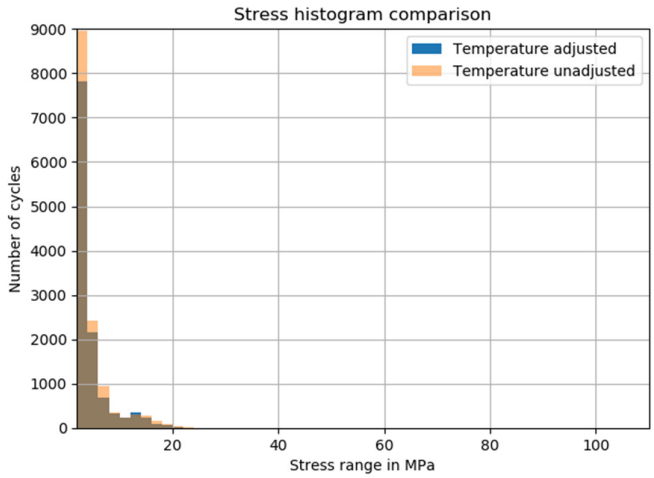


Figure 9-21 Slightly cropped stress histogram for strain gauge 110, excluding stress ranges below 2MPa

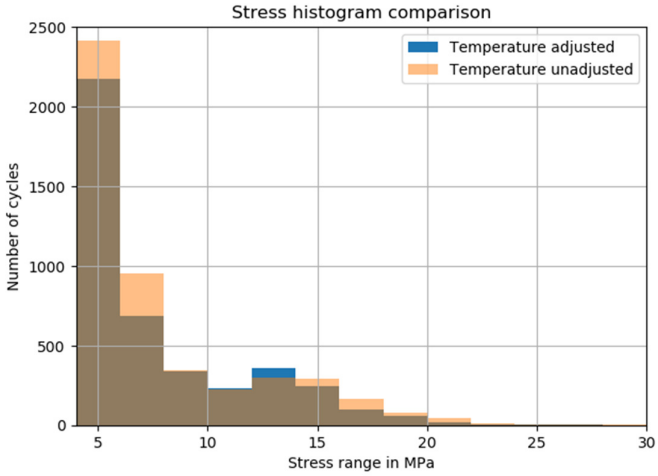


Figure 9-22 Stress histogram of strain gauge 110, showing stress ranges between 4MPa and 30MPa.

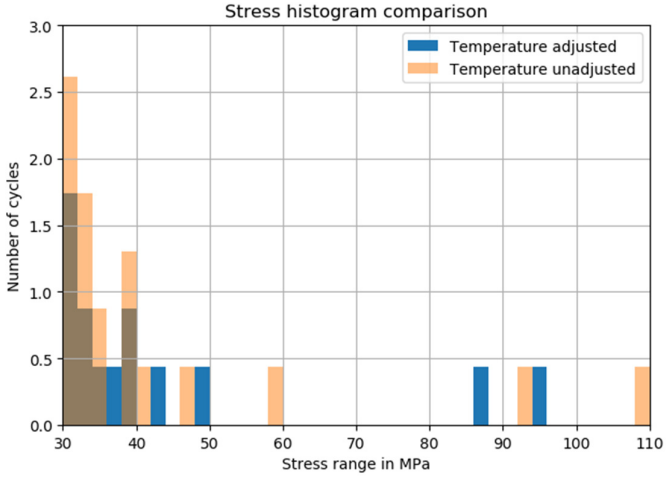


Figure 9-23 Stress histogram for strain gauge 110, from 30MPa to 110MPa

9.3.2.2 Strain gauge 112

The strain history for this detail does not show the same behavior as detailed for strain gauge 110, in that it is not uniformly offset and does not appear to have a flatter curve, as is shown in Figure 9-24.

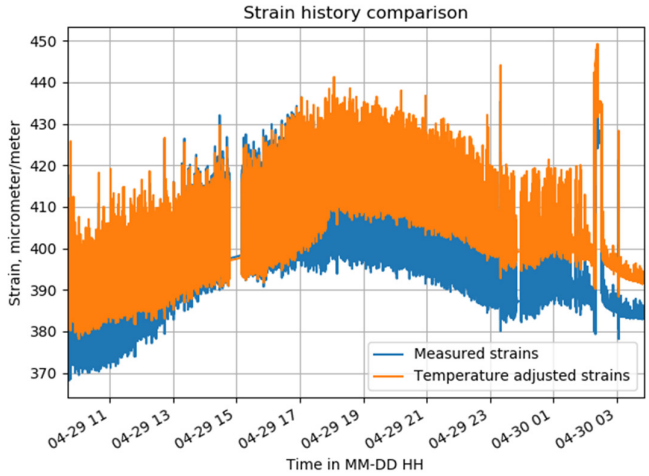


Figure 9-24 Strain history for strain gauge 112, for both the temperature adjusted and unadjusted strains.

The stress range histogram for the detail is shown in Figure 9-25 to Figure 9-28. The different figures represent different parts of the histogram for better detail. The histograms show the same behavior as for gauge 110. The results after converting the stress ranges and cycles to an equivalent stress range are represented in Table 9-6. The fewer cycles at the higher stress ranges becomes apparent again.

Table 9-6 The equivalent stress range and total number of cycles based on the stress history given in Figure 9-24. For strain gauge 112.

	Adjusted	Unadjusted
n_{eq}	52	86
$\Delta\sigma_{eq}$	18.28	18.27

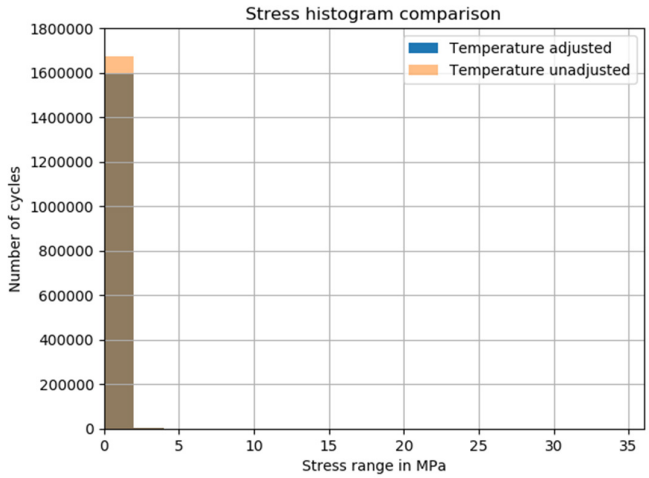


Figure 9-25 The full stress histogram for strain gauge 112.

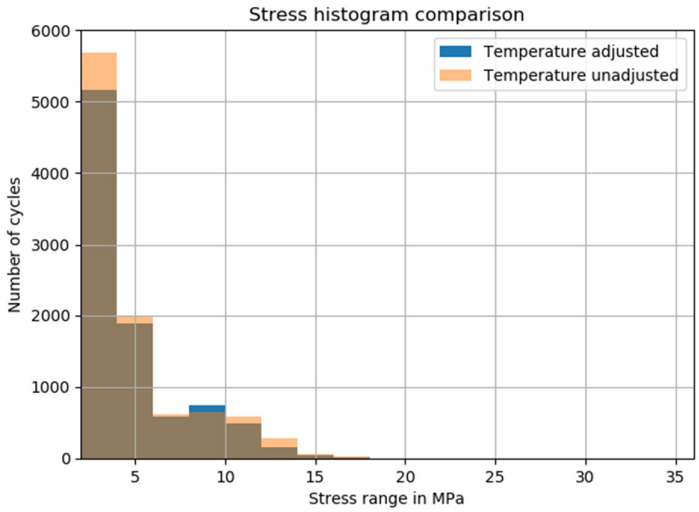


Figure 9-26 Slightly cropped stress histogram for strain gauge 112, from 2MPa to 36MPa

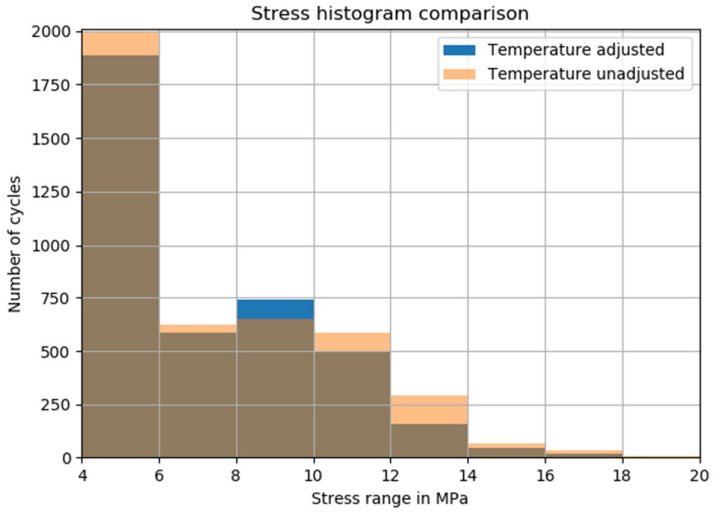


Figure 9-27 Stress histogram for strain gauge 112, from 4MPa to 20MPa

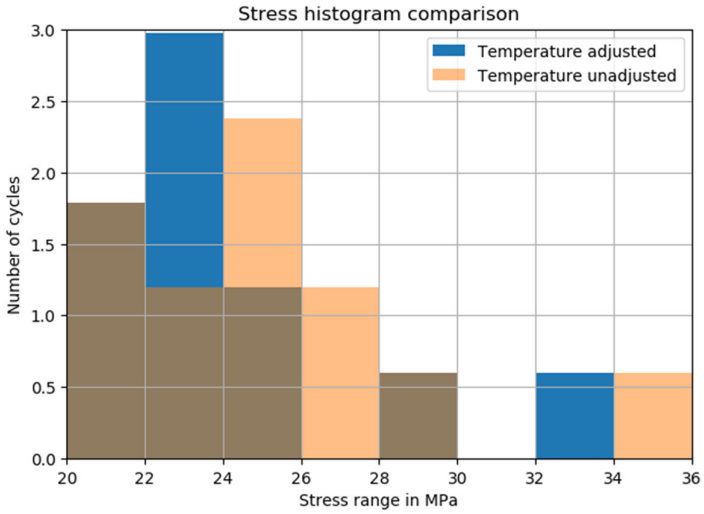


Figure 9-28 Stress histogram for strain gauge 112, from 20MPa to 36MPa

9.3.2.3 The difference in results between measurement locations 110 and 112

The combined results from strain gauges 110 and 112 are compared, for the temperature adjusted strains as well as temperature unadjusted strains, in Table 9-7. As is evident from the numbers, the temperature unadjusted strains cause a higher damage relative to the temperature adjusted strains. The main contributing factor to the higher damage is the higher number of cycles that exist for the unadjusted strains. The other main point is the difference in results of the predicted damage at the location of detail 1. Strain gauge 110 predicts a damage that is more than ten times larger, compared to strain gauge 112. The reason behind the difference in predicted damage is possibly due to the difference in the applied SAF between the two locations, as well as the lower accuracy for using the SAF to convert stress ranges between two locations that are far separated.

Table 9-7 Summarized result of the calculated damage for detail 1, based either on strain gauge 110 or 112, as well as either temperature corrected or uncorrected strains.

	Strain gauge 110		Strain gauge 112	
	Adjusted	Unadjusted	Adjusted	Unadjusted
n_{eq}	363	523	52	86
$\Delta\sigma_{eq}, MPa$	20.48	20.61	18.28	18.27
$D_{measured}$	$98.94 \cdot 10^{-7}$	$147.22 \cdot 10^{-7}$	$7.995 \cdot 10^{-7}$	$13.265 \cdot 10^{-7}$
$D_{Year\ equivalent}$	$47.67 \cdot 10^{-4}$	$70.93 \cdot 10^{-4}$	$3.852 \cdot 10^{-4}$	$6.391 \cdot 10^{-4}$
$D_{Dismantel}$	$29.00 \cdot 10^{-3}$	$43.15 \cdot 10^{-3}$	$2.343 \cdot 10^{-3}$	$3.888 \cdot 10^{-3}$

9.4 Discussion and conclusions

From the presented results, it can be easily concluded that trams are the vehicles which have caused the largest fatigue damage in the bridge. What is unexcepted is that essentially all road vehicles do not contribute to the damage being accumulated. It may

be explained by the fact that the main girders are substantially oversized and road traffic is not able to generate stress ranges above the cut-off limit.

Additionally, the carried out fatigue evaluation without safety factors can still be considered on the safe side. The reason is the fact that detail categories were lowered to take into account misalignments and inferior-quality welds. Furthermore, tramway models nominated to represent tramway traffic were applied to the entire period between 1939–2015, although various types of tramways served on the lines going through Göta Älv bridge. What is more, the intensity and the volume of the traffic varied in a non-linear way, therefore without thorough traffic documentation from each year, it is hard to say when exactly theoretical service life of the bridge was exhausted. Nonetheless, it can be stated that made assumptions are reasonable from an engineering point of view and in sufficiently good manner reflects the traffics impact on the bridge. On the other hand, using a different method, e.g. fracture mechanics, should be incorporated which considers the propagation of the crack to investigate places where cracks arose.

10 Conclusions, discussion, and further studies

The following sections, conclusions, discussion and further studies, are divided according to the main parts of the thesis, the calibration of the FE-model, the spatial adjustment factor, and the fatigue evaluation.

10.1 Conclusions

Calibration of the FE model

From the carried-out calibration of the model, it was concluded that:

- The appropriate implementation of the main girders offset had the biggest impact on the calibration process.
- Proper placement of the crossbeams was the second most significant factor.
- The grade of the concrete or steel did not affect the results in any meaningful way.
- 50% improvement regarding the error was attained in comparison to the original model.

Automated optimization of the FE model

From the completed automatic calibration of the FE-model, the following conclusions were drawn:

- The automated process validated the results of the manual calibration since little gain was made over the manually calibrated model.
- Correctly chosen parameters are important, as the rotational spring values did not change over the course of the calibration process.

Spatial Adjustment Factors

The presented concept in this thesis gives possibilities to transform measured data from one location to another by means of influence lines derived from a calibrated FE model. The suggested approach may help to reduce the number of sensors installed on the structure substantially. The carried-out investigation showed that:

- The shape of the influence lines affects the correlation. The more similar influence lines, the better correlation may be obtained.
- Good correlation was obtained for points located 2 meters from each other.
- The length of the influence line and axles' spacing have a dominant impact on the correlation between points of interest.
- The one-span beam case provided a thorough understanding of the phenomena and explained how the length of the vehicle affects SAF.
- About FLM 4, it turned out that set of lorries proposed in the Eurocode very well reflect traffic existing on the roads and hence satisfying results about SAF are possible to obtain. SAF based on FLM 4 can be treated for a conservative approach.

Fatigue evaluation of the Göta Älv Bridge

Looking at the results of the conducted fatigue evaluation based on historical data, it can be noted that:

- Tramways were the main source of the accumulated fatigue damage in Göta Älv Bridge. The damage caused by the passage of other vehicles was negligible.

- Due to the significant dimensions of the bridge's main girders, it was shown that the bridge was insensitive to vehicles below 40 tones, as they were unable to cause any fatigue damage to the bridge.
- The apparently conservative assumption made about the choice of the C-category can be justified by the fact that two cracks had been detected at the northern viaduct part of the bridge in the past. Hence, obtained damage index above 1 can be considered a realistic value.

The fatigue evaluation based on the measured data amounted to the following:

- For the viaduct, the data gave some interesting results, in that it had stress ranges remaining above the cut-off limit. The damage caused to the detail, extrapolated over the time until its dismantling was estimated at 2.7% of the chosen detail's total endured damage.
- For the river part, the data remaining after the application of the cut-off limit was small, only a few cycles remained. Thus, the damage caused from the time of the measurement to the planned dismantling of the bridge was estimated at about 0.0020% of the detail's total endured damage, which is essentially negligible.

10.2 Discussion

Calibration

The manual calibration could be viewed as a preparation for the automatic calibration. However, the process of evaluating which parameters were sensitive regarding the FE-models response took more time than expected at the beginning of the thesis. It is worth mentioning that the automated calibration cannot be successful without good engineering judgment regarding the selection of parameters.

The script responsible for the automated optimization of the error function, and thus the calibration of the FE-model, is a powerful tool once completed. The time required for the creation of the script can be somewhat long though, and it needs to be fine-tuned for each specific case. However, once such a script is completed, it provides the engineer with a tool that can evaluate a large series of parameter combinations. Thus, it will be a time-saving tool for future users with the knowledge of RSA API and the IronPython scripting language. The automated process might also have benefitted from using more load cases in the comparison between the measured and calculated stresses. But the fewer load cases save on computational time. Thus, it is a question of balancing the two.

SAF

The concept was inspired by a work from Liu et al. (2010). But considerable effort was spent in this study to build upon and modify the method to include the effect of moving loads in the evaluation process. Due to its empirical nature, the method needs more investigation and refinement. Because of our brief investigation, it cannot be certainly approved for a general use case. Nonetheless, its validity was proved for the case study. What is more, it seems to be a promising concept for the future.

Fatigue evaluation

After the performed assessment based on historical data, it was concluded that the fatigue life was exhausted before 2015. It is not readily possible to establish the exact year when the fatigue life was spent. The reason is that source data has a character of

statistical average of the annual traffic existing on the bridge. It is important to mention that traffic volume changed in a non-linear way during the service life of the bridge, therefore we are not able to evaluate the exact year.

Regarding the fatigue assessment based on the measured data in both studied details, the recorded strains without temperature adjustment always lead to a larger damage index, which is due to the temperature correction process. This process is based on the procedure outlined by the manufacturer of the strain gauges, along with the usage of the air temperature for the approximation of the temperature in the girders. Using the latter data was due to the lack of temperature measurements within the girders at the time of the collected strains. Both the procedure and the approximation for evaluating the thermal strains can be improved. Most notable improvement is by direct measurement of thermal strains using a separate strain gage installed on a free-to-move part of the component which only registers the thermal expansion of the member.

10.3 Further studies

Calibration

It is expected that a better convergence with measurements may be obtained if a more advanced FE model would be created. Modeling main girders by means of shell elements instead beam elements should provide more accurate results.

SAF

The following subjects can be suggested for further studies:

- Determining the maximum spacing between the points of interest which still provide satisfying results.
- Length of the span in multi-span beams and how does it affect SAF.
- Utilize a vector form of the spatial adjustment factor concept, instead of a scalar. Thus, the created vector would contain several spatial adjustment factors, each specifically applicable to one stress range and its corresponding number of cycles. It is theorized that this would bring more accurate results as it would be a less crude way of converting the stresses.

11 References

Al-Emrani, M., & Åkesson, B. (2013). *Steel Structures* (No. 2013:10) (p. 268). Göteborg, Sweden: Chalmers University of Technology.

Al-Emrani, M., & Aygül, M. (2013). *Fatigue design of steel and composite bridges* (No. 2014:10) (p. 166). Göteborg, Sweden: Chalmers University of Technology.

Amzallag, C., Gerey, J. P., Robert, J. L., & Bahuaud, J. (1994). Standardization of the rainflow counting method for fatigue analysis. *International Journal of Fatigue*, *16*(4), 287–293. https://doi.org/10.1016/0142-1123(94)90343-3

Antoniou, A., Murray, W., Wright, M. H., & SpringerLink (e-book collection). (2007). *Practical optimization: algorithms and engineering applications*. New York, N.Y: Springer. https://doi.org/10.1007/978-0-387-71107-2

ASTM. (2011). *Standard Practices for Cycle Counting in Fatigue Analysis* (No. E1049 – 85) (p. 10). West Conshohocken. Retrieved from https://doi-org.proxy.lib.chalmers.se/10.1520/E1049-85R11E01

ASTM International. (2017). *Standard Practices for Cycle Counting in Fatigue Analysis* (No. E1049 – 85) (p. 10). West Conshohocken: ASTM International. https://doi.org/10.1520/E1049-85R17.

Carlsson, F. (2011). *Utredning av Ekvivalent Skadefaktor för Utmattningsanalyser av Stålbroar - Baserad på mätningar av verkliga fordonslaster i Sverige* (No. VBK-3062). Lund. Sweden: Lunds Tekniska Högskola Avdelningen för Konstruktionsteknik.

EN 1991-2. (2003). *Eurocode 1: Actions on structures — Part 2: Traffic loads on bridges* (No. EN 1991-2:2003/AC 2010). Brussels, Belgium: European Committee for Standardization (CEN).

Hicken, J., Alonso, J., Farhat, C., & Rajnarayan, D. (2012). *Course literature "AA222 - Introduction to Multidisciplinary Design Optimization."* Stanford University. Retrieved from http://adl.stanford.edu/aa222/Home.html

J. Willmott, C., & Matsuura, K. (2005). *Advantages of the Mean Absolute Error (MAE) over the Root Mean Square Error (RMSE) in Assessing Average Model Performance* (Vol. 30). https://doi.org/10.3354/cr030079

Janiszewski. (2016). rainflow 1.0 (Version 1.0). Retrieved from https://testpypi.python.org/pypi/rainflow

Kühn, B., Lukić, M., Nussbaumer, A., Günther, H. P., Helmerich, R., Herion, S., … Bucak, Ö. (2008). *Assessment of Existing Steel Structures-Recommendations for Estimation of Remaining Fatigue Life*.

Leander, J., Trillkott, S., & Kullberg, C. (2015). *Götaälvbron–Töjningsmätningar för kalibrering av beräkningsmodell (Götaälv bridge–Strain measuremnts for*

calibration of the analysis model). Stockholm, Sweden: Royal Institute of Technology (KTH).

Liu, M., Frangopol, D. M., & Kwon, K. (2010). Fatigue reliability assessment of retrofitted steel bridges integrating monitored data. *Structural Safety*, *32*(1), 77–89. https://doi.org/10.1016/j.strusafe.2009.08.003

Marsh, G., Wignall, C., Thies, P. R., Barltrop, N., Incecik, A., Venugopal, V., & Johanning, L. (2015). Review and application of Rainflow residue processing techniques for accurate fatigue damage estimation. *International Journal of Fatigue*, *2016*(82), 757–765.

Matsuishi, M., & Endo, T. (1968). Fatigue of metals subjected to varying stress. *Japan Society of Mechanical Engineers*, *1968*.

Može, P. (2000, June 12). Lecture 12.2: Advanced Introduction to Fatigue [Faculty of Civil and Geodetic Engineering]. Retrieved April 6, 2018, from http://fgg-web.fgg.uni-lj.si/~/pmoze/esdep/master/wg12/l0200.htm

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*(4), 308–313.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed). New York: Springer.

Okamura, H., Sakai, S., & Susuki, I. (1979). Cumulative fatigue damage under random loads. *Fatigue & Fracture of Engineering Materials and Structures*, *1*(4), 409–419. https://doi.org/10.1111/j.1460-2695.1979.tb01328.x

Olsson, J. (2015). *Bro 1480-1001-1 över Göta älv Götaälvbron Rapport beräkningsmodeller* (p. 40). Göteborg: ÅF-Infrastructure AB.

SMHI. (2018). Meteorologiska observationer. Retrieved April 24, 2018, from https://opendata-download-metobs.smhi.se/explore/?parameter=0

Trafikverket. (2017). *Krav - Bärighetsberäkning av broar (Version 4.0)* (No. TDOK 2013:0267). Sweden: Trafikverket (Swedish road administration).

VPG Sensors, M. measurements. (2014, August 14). Strain Gage Thermal Output and Gage Factor Variation with Temperature. Micro measurements - VPG Sensors.

Appendices

Appendix A – detailed results for General Case (SAF)

Detailed results regarding SAF for General Case. Tables contain total number of cycles, equivalent stress ranges and SAF of investigated models.

2 SPAN BEAM			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	4,313	1,649	2,616
$\Delta\sigma eq$	8,58	19,2746	0,445

2 SPAN BEAM			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	2,000	1,500	1,333
$\Delta\sigma eq$	4,050	8,484	0,477

2 SPAN BEAM			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	3,000	1,500	2,000
$\Delta\sigma eq$	7,130	17,240	0,414

2 SPAN BEAM			
FLM4 - Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,250	1,600	2,031
$\Delta\sigma eq$	7,030	16,460	0,427

2 SPAN BEAM			
FLM4 - Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,650	1,950	1,872
$\Delta\sigma eq$	8,810	16,550	0,532

2 SPAN BEAM			
FLM4 - Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,050	2,150	1,884
$\Delta\sigma eq$	9,450	20,870	0,453

2 SPAN BEAM			
Random traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,839	1,500	2,559
$\Delta\sigma eq$	5,530	13,420	0,412

3 SPAN BEAM			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	4,458	2,348	1,899
$\Delta\sigma eq$	10,06	4,691328	2,145

3 SPAN BEAM			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	2,500	2,000	1,250
$\Delta\sigma eq$	3,125	7,160	0,436

3 SPAN BEAM			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	4,900	2,650	1,849
$\Delta\sigma eq$	9,510	16,480	0,577

3 SPAN BEAM			
FLM4 - Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,800	2,150	1,767
$\Delta\sigma eq$	6,608	13,670	0,483

3 SPAN BEAM			
FLM4 - Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,250	2,500	1,700
$\Delta\sigma eq$	6,820	16,520	0,413

3 SPAN BEAM			
FLM4 - Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,600	2,750	1,673
$\Delta\sigma eq$	10,620	17,606	0,603

3 SPAN BEAM			
Random traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,941	2,199	1,792
$\Delta\sigma eq$	2,739	3,928	0,697

4 SPAN BEAM			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	5,397	2,984	1,809
$\Delta\sigma eq$	9,59	14,67829	0,653

4 SPAN BEAM			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	3,000	2,500	1,200
$\Delta\sigma eq$	3,756	6,460	0,581

4 SPAN BEAM			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	4,000	3,500	1,143
$\Delta\sigma eq$	7,280	11,510	0,632

4 SPAN BEAM			
FLM4 - Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,300	2,750	1,564
$\Delta\sigma eq$	7,160	12,155	0,589

4 SPAN BEAM			
FLM4 - Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,750	3,250	1,462
$\Delta\sigma eq$	9,030	14,491	0,623

4 SPAN BEAM			
FLM4 - Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	5,100	3,450	1,478
$\Delta\sigma eq$	9,800	15,555	0,630

4 SPAN BEAM			
Random traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,839	2,950	1,640
$\Delta\sigma eq$	6,311	10,089	0,626

Göta Älv Bridge			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	4,2	4,9	0,857
$\Delta\sigma eq$	12,46	13,84	0,900

Göta Älv Bridge			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	3,000	2,500	1,200
$\Delta\sigma eq$	1,690	2,060	0,820

Göta Älv Bridge			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	3,000	4,500	0,667
$\Delta\sigma eq$	3,704	3,036	1,220

Göta Älv Bridge			
FLM4 - Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,250	3,650	1,164
$\Delta\sigma eq$	3,380	3,720	0,909

Göta Älv Bridge			
FLM4 - Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,350	3,900	1,115
$\Delta\sigma eq$	4,050	4,580	0,884

Göta Älv Bridge			
FLM4 - Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,550	4,200	1,083
$\Delta\sigma eq$	4,310	4,900	0,880

Göta Älv Bridge			
Random traffic			
	SUPPORT	SPAN	SAF
n_{tot}	5,000	3,850	1,299
$\Delta\sigma eq$	2,467	2,500	0,987

Appendix B – detailed results for Specific Case (SAF)

Detailed results regarding SAF for Specific Case. Tables contain total number of cycles, equivalent stress ranges and SAF of investigated models.

2 SPAN BEAM			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	4,31	3,06	1,41
$\Delta\sigma eq$	8,58	7,93	1,08

2 SPAN BEAM			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	2,00	2,00	1,00
$\Delta\sigma eq$	4,05	4,56	0,89

2 SPAN BEAM			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	2,00	2,00	1,00
$\Delta\sigma eq$	8,47	8,86	0,96

2 SPAN BEAM			
Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,25	2,35	1,38
$\Delta\sigma eq$	8,68	8,68	1,00

2 SPAN BEAM			
Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,65	2,85	1,28
$\Delta\sigma eq$	10,87	10,49	1,04

2 SPAN BEAM			
Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,05	3,15	1,29
$\Delta\sigma eq$	11,58	11,10	1,04

2 SPAN BEAM			
Random traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,84	2,64	1,45
$\Delta\sigma eq$	5,53	5,54	1,00

3 SPAN BEAM			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	4,46	3,99	1,12
$\Delta\sigma eq$	10,06	8,70	1,16

3 SPAN BEAM			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	2,50	2,50	1,00
$\Delta\sigma eq$	4,24	4,48	0,95

3 SPAN BEAM			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	2,50	2,50	1,00
$\Delta\sigma eq$	9,86	8,86	1,11

3 SPAN BEAM			
Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,80	2,80	1,36
$\Delta\sigma eq$	9,98	9,16	1,09

3 SPAN BEAM			
Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,25	3,25	1,31
$\Delta\sigma eq$	12,63	11,11	1,14

3 SPAN BEAM			
Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,60	3,85	1,19
$\Delta\sigma eq$	13,52	14,03	0,96

3 SPAN BEAM			
Random traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,94	3,14	1,26
$\Delta\sigma eq$	6,75	6,16	1,10

4 SPAN BEAM			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	5,40	4,49	1,20
$\Delta\sigma eq$	9,59	7,91	1,21

4 SPAN BEAM			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	3,00	3,00	1,00
$\Delta\sigma eq$	4,02	3,98	1,01

4 SPAN BEAM			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	3,00	3,00	1,00
$\Delta\sigma eq$	9,12	7,92	1,15

4 SPAN BEAM			
Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,30	3,20	1,34
$\Delta\sigma eq$	9,26	8,36	1,11

4 SPAN BEAM			
Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,75	3,90	1,22
$\Delta\sigma eq$	11,58	10,24	1,13

4 SPAN BEAM			
Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	5,10	4,25	1,20
$\Delta\sigma eq$	12,41	10,91	1,14

4 SPAN BEAM			
Random traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,84	3,64	1,33
$\Delta\sigma eq$	6,31	5,38	1,17

Göta Älv Bridge			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	4,90	4,55	1,08
$\Delta\sigma eq$	4,84	3,81	1,27

Göta Älv Bridge			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	3,00	3,00	1,00
$\Delta\sigma eq$	1,80	2,03	0,89

Göta Älv Bridge			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	3,00	3,00	1,00
$\Delta\sigma eq$	4,20	3,80	1,11

Göta Älv Bridge			
Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,25	4,10	1,04
$\Delta\sigma eq$	4,65	3,82	1,22

Göta Älv Bridge			
Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,35	4,30	1,01
$\Delta\sigma eq$	5,93	4,800957	1,234617

Göta Älv Bridge			
Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,55	4,65	0,98
$\Delta\sigma eq$	6,41	5,143401	1,245872

Göta Älv Bridge			
Random traffic			
	SUPPORT	SPAN	SAF
n_{tot}	5,00	4,35	1,15
$\Delta\sigma eq$	3,16	2,500163	1,265254

Appendix C – detailed results for One-span Case (SAF)

Detailed results regarding SAF for One-span case. Tables contain total number of cycles, equivalent stress ranges and SAF of investigated models.

5 meter span			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	4,483	2,83	1,584099
$\Delta\sigma eq$	0,81	2,669166	0,305036

5 meter span			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	0,90	2,6	0,346154

5 meter span			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	4	2	2
$\Delta\sigma eq$	0,91	2,87	0,317073

5 meter span			
Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	2,45	2,2	1,113636
$\Delta\sigma eq$	1,22	3,12	0,391026

5 meter span			
Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	3,45	2,55	1,352941
$\Delta\sigma eq$	1,47	3,74	0,393048

5 meter span			
Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	4,15	2,85	1,45614
$\Delta\sigma eq$	1,53	3,88	0,39433

5 meter span			
measured up to 12m length vehicles			
	SUPPORT	SPAN	SAF
n_{tot}	3,04	2,29	1,327511
$\Delta\sigma eq$	0,84	2,04	0,411765

10 meters span			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	2,991	1,603	1,865876
$\Delta\sigma eq$	4,41	7,605887	0,580182

10 meters span			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	3,40	5,2	0,653846

10 meters span			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	2	2	1
$\Delta\sigma eq$	5,22	6,96	0,75

10 meters span			
Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	2,2	1,2	1,833333
$\Delta\sigma eq$	4,96	8,14	0,609337

10 meters span			
Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	2,55	1,55	1,645161
$\Delta\sigma eq$	5,75	9,45	0,608466

10 meters span			
Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	2,85	1,85	1,540541
$\Delta\sigma eq$	5,92	9,62	0,615385

10 meters span			
measured up to 12m length vehicles			
	SUPPORT	SPAN	SAF
n_{tot}	2,54	1,29	1,968992
$\Delta\sigma eq$	3,55	5,65	0,628319

20 meters span			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	2	1,065	1,877934
$\Delta\sigma eq$	12,33	22,46134	0,548807

20 meters span			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	6,70	10,50	0,638095

20 meters span			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	14,00	22,00	0,636364

20 meters span			
Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	2,05	1	2,05
$\Delta\sigma eq$	13,56	22,75	0,596044

40 meters span			
Reference case			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	37,66	59,30381	0,634976

40 meters span			
single axle 100kN			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	13,50	21,10	0,63981

40 meters span			
single truck 25 tonnes			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	31,00	48,10	0,644491

40 meters span			
Local traffic			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	36,72	55,60	0,660432

40 meters span			
Medium traffic			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	47,18	71,07	0,663853

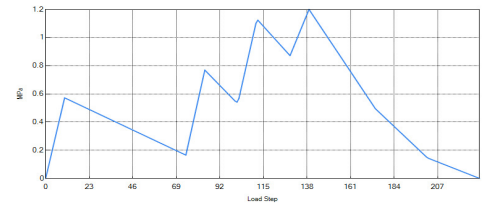
40 meters span			
Long dist. Traffic			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	51,46	77,48	0,664171

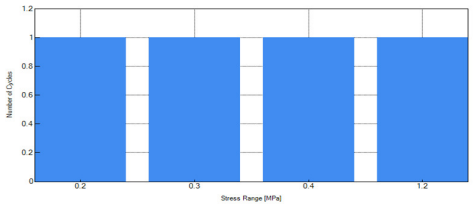
40 meters span			
measured up to 12m length vehicles			
	SUPPORT	SPAN	SAF
n_{tot}	1	1	1
$\Delta\sigma eq$	24,25	37,69	0,643407

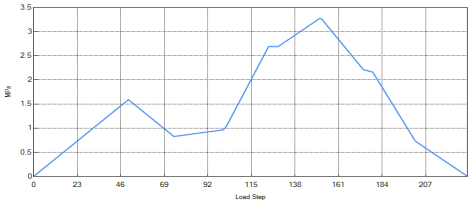
Appendix D – stress-time histories One-span Case

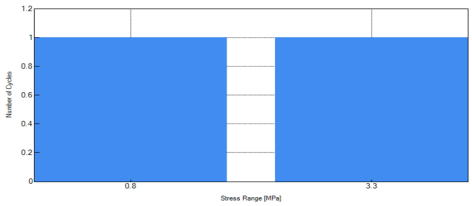
Stress-time history and stress ranges histograms from One-span beams case.

5-meter long influence line





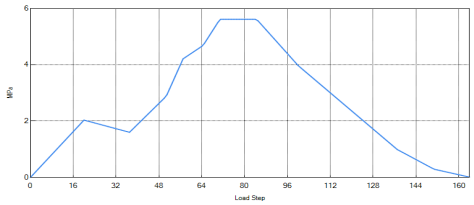


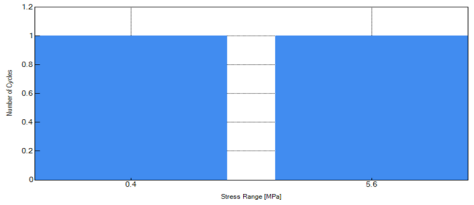


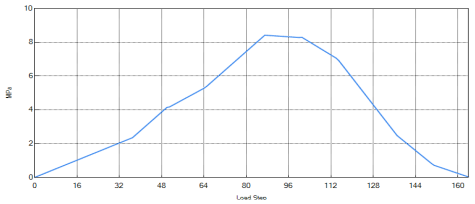
n_{tot}	4,00
$\Delta\sigma eq$	0,91

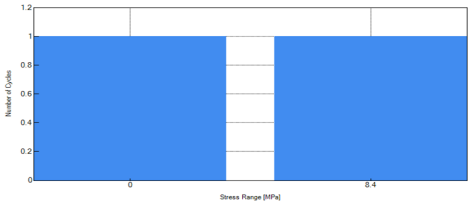
		SAF
n_{tot}	2,00	**2,00**
$\Delta\sigma eq$	2,87	**0,32**

10-meter long influence line





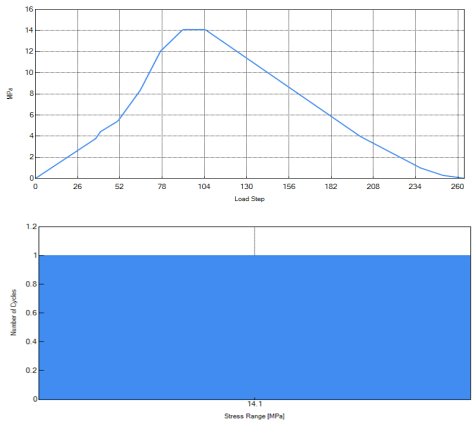


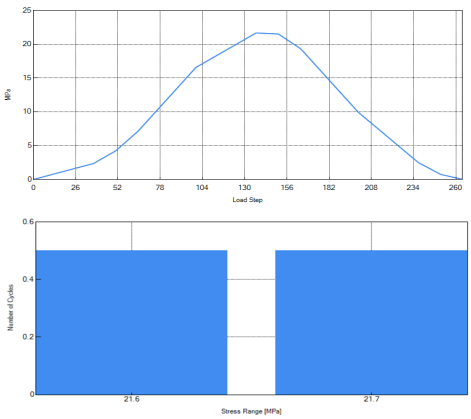


n_{tot}	2,00
$\Delta\sigma eq$	5,22

		SAF
n_{tot}	2,00	**1,00**
$\Delta\sigma eq$	6,96	**0,75**

20-meter long influence line

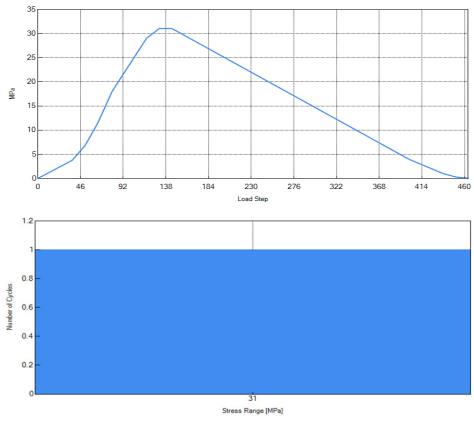


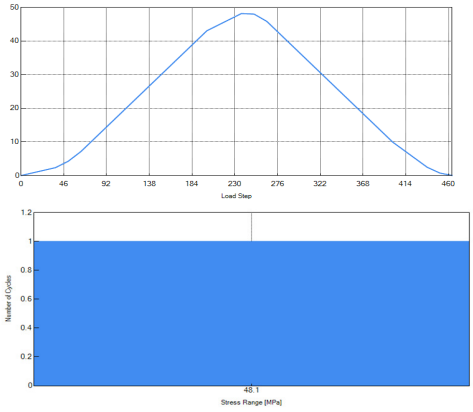


n_{tot}	1,00
$\Delta\sigma eq$	14,00

		SAF
n_{tot}	1,00	**1,00**
$\Delta\sigma eq$	22,00	**0,64**

40-meter long influence line





n_{tot}	1,00
$\Delta\sigma eq$	31,00

		SAF
n_{tot}	1,00	**1,00**
$\Delta\sigma eq$	48,10	**0,64**

Appendix E – Calculation of Dynamic Amplification Factor

Calculation of dynamic amplification factor
According to TRV BRO KRAV §2.3.3.2.5 (similar to EC ANNEX C)

River part - Götaälvbron

$\Sigma L := 165m$ Total length of spans

$n_{spans} := 4$ Number of spans

$L_m := \frac{\Sigma L}{n_{spans}} = 41.25\,m$ Mean span length

$k_{multispan} := 1.5$ Factor from Krav

$L_\varphi := k_{multispan} \cdot L_m$ Determinant length

$L_\varphi = 61.875\,m$

$v := 50\frac{km}{hr} = 13.889\frac{m}{s}$ Maximal permitted speed

$n_0 := 1.64Hz$ First natural vibration frequency of the bridge (from FE model)

$K := \frac{\left(\frac{v}{1\frac{m}{s}}\right)}{2 \cdot \left(\frac{L_\varphi}{1m}\right) \cdot \frac{n_0}{1Hz}}$ $K = 0.068$

$\varphi' := \frac{K}{1 - K + K^4} = 0.073$

$\alpha := \frac{v}{22\frac{m}{s}} = 0.631$

$\varphi'' := \frac{\alpha}{100}\left[56 \cdot \exp\left[-\left(\frac{L_\varphi}{10m}\right)^2\right] + 50\left(\frac{n_0 \cdot L_\varphi}{1Hz \cdot 1m \cdot 80} - 1\right) \cdot \exp\left[-\left(\frac{L_\varphi}{20m}\right)^2\right]\right] = 5.906 \times 10^{-6}$

$D_{TRV} := 1 + \varphi' + 0.5 \cdot \varphi'' = 1.073$ Dynamic amplification factor (DAF)

Calculation of dynamic amplification factor
According to TRV BRO KRAV §2.3.3.2.5 (similar to EC ANNEX C)

Södra vidaukten - Götaälvbron

$\Sigma L := 137.6\,m$ Total length of spans

$n_{spans} := 10$ Number of spans

$L_m := \frac{\Sigma L}{n_{spans}} = 13.76\,m$ Mean span length

$k_{multispan} := 1.5$ Factor from Krav

$L_\varphi := k_{multispan} \cdot L_m$ Determinant length

$L_\varphi = 20.64\,m$

$v := 50\frac{km}{hr} = 13.889\frac{m}{s}$ Maximal permitted speed

$n_0 := 1.64Hz$ First natural vibration frequency of the bridge (from FE model)

$K := \frac{\left(\frac{v}{1\frac{m}{s}}\right)}{2\cdot\left(\frac{L_\varphi}{1m}\right)\cdot\frac{n_0}{1Hz}}$ $K = 0.205$

$\varphi' := \frac{K}{1 - K + K^4} = 0.258$

$\alpha := \frac{v}{22\frac{m}{s}} = 0.631$

$\varphi'' := \frac{\alpha}{100}\left[56\cdot\exp\left[-\left(\frac{L_\varphi}{10m}\right)^2\right] + 50\left(\frac{n_0\cdot L_\varphi}{1Hz\cdot 1m\cdot 80} - 1\right)\cdot\exp\left[-\left(\frac{L_\varphi}{20m}\right)^2\right]\right] = -0.058$

$D_{TRV} := 1 + \varphi' + 0.5\cdot\varphi'' = 1.229$ Dynamic amplification factor (DAF)

Appendix F – FE model calibration scripts

F.1 – Main script of the automated calibration of the Robot Structural Analysis finite element model, optimization using Nelder-Mead

This is the main script that controls the optimization process and initiates the Nelder-Mead method to the target function. The script for the target function is given in appendix F.2.

```
1.  ##########################################################################################################
    #########
2.  # Note:
3.  # The robot model has to be opened before the script is run!
4.  # It has to be the model which was prepared for the calibration, as it has to correct groups for th
    e        # paramter change.
5.  # The script applies the parameters defined below, to the model.
6.  # The numbering of the load cases defined within the model has to start from 1,and continue contino
    usly     # until there is no more load cases, i.e. 1,2,3,4 not 2,8,4,9 or similar randomness.
7.  ##########################################################################################################
    #########
8.  # Importing some stuff
9.  import clr
10. import csv
11.
12. # Target function
13. import tf
14.
15. # Adding references to the Extreme Optimization package
16. clr.AddReferenceToFileAndPath("C:\\Program Files (x86)\\Extreme Optimization\\Numerical Libraries f
    or .NET 6.0\\bin\\Net40\\Extreme.Numerics.Net40.dll")
17. clr.AddReferenceToFileAndPath("C:\\Program Files (x86)\\IronPython 2.7\\DLLs\\Extreme.Numerics.Iron
    Python.dll")
18.
19. # Importing the optimization function
20. from Extreme.Mathematics.Optimization import *
21. from Extreme.Mathematics.LinearAlgebra import *
22. from Extreme.Mathematics import *
23.
24. # Importing RSA
25. clr.AddReferenceToFileAndPath('C:\Program Files\Autodesk\Autodesk Robot Structural Analysis Profess
    ional 2017\System\Exe\Interop.RobotOM.dll')
26.
27. # Importing RSA
28. import RobotOM as rbt
29.
30. # Shortcuts for commonly used commands, in order to minimize the code written
31. robapp=rbt.RobotApplicationClass()
32. struc=robapp.Project.Structure
33. node=struc.Nodes
34. bar=struc.Bars
35. labels=struc.Labels
36. stress=struc.Results.Bars.Stresses
37. force=struc.Results.Bars.Forces
38.
39. robapp.Project.Structure.ResultsFreeze = False
40. robapp.Project.CalcEngine.UseStatusWindow=True
41. robapp.Project.CalcEngine.AnalysisParams.IgnoreWarnings=True
42.
43. # Parameters to be changed, assumed input
44. rot_stiff_support_input=1000000*(180/3.1415926535)          # Rotatioanl stiffenss
45. main_offset_input=-1.57                                    # Note, should be negative
46. rot_stiff_trans_input=1000000*(180/3.1415926535)           # Stiffness of the interaction springs
47. trans_offset_input=0
48.
49. # Inserting the values that are to be changed into a vector for the optimization function
50. initialGuess=Vector.Create[float](4)
```

```
initialGuess[0]=rot_stiff_support_input
initialGuess[1]=main_offset_input
initialGuess[2]=rot_stiff_trans_input
initialGuess[3]=trans_offset_input

# Shortcut to the Nelder-
Mead optimization function. The method is implemented by the NelderMeadOptimizer class:
nm=NelderMeadOptimizer()

# The class has three special properties, that help determine
# the progress of the algorithm. These parameters have
# default values and need not be set explicitly.
nm.ContractionFactor = 0.5
nm.ExpansionFactor = 2
nm.ReflectionFactor = -2

# Limiting the iteration counts, so that the optimzation gets going, aswell as that it does not run
 for an eternity
nm.MinIterations=500
nm.MaxIterations=1000

nm.SolutionTest.AbsoluteTolerance = 1e-18
nm.InitialGuess = initialGuess

# Targetfunction
nm.ObjectiveFunction = tf.target_function

# Initiate
#nm.FindExtremum()

print "Nelder-Mead Method:"
print "  Solution:", nm.Extremum
print "  Estimated error:", nm.EstimatedError
print "  # iterations:", nm.IterationsNeeded
print "  # function evaluations:", nm.EvaluationsNeeded

# Saving the FE-
model so that if the computer is shut down (after the calculation is done) the results are still sa
ved
#robapp.Project.Save()

with open('final_guess.csv', 'w') as csvfile:
            fieldnames=['rot_stiff_support_input','main_offset_input','rot_stiff_trans_input','tran
s_offset_input']
            writer=csv.DictWriter(csvfile,fieldnames)
            writer.writeheader()
            writer.writerow({fieldnames[0]:initialGuess[0],
                             fieldnames[1]:initialGuess[1],
                             fieldnames[2]:initialGuess[2],
                             fieldnames[3]:initialGuess[3]})
```

F.2 – Target function for the automated calibration

The full code for the target function used for the optimization. Specifically created for the Göta Älv Bridge FE-model calibration.

```
# -*- coding: utf-8 -*-
################################################################################################################
# The number of loadcases evaluated is based on couting the number of available load cases within the model
# There should be an equal number of CSV files to this number.
# The CSV files should contain bar_id, sigma_top and sigma_bot.
# The comparison is based on extracting the forces from the bars, the using cross sectional data to get stresses
################################################################################################################
def target_function(initialGuess):

    # Importing some stuff
    import clr
    import math
    import sys
    import csv
    import time

    # Timer
    start = time.time()

    # Accessing RSA
    clr.AddReferenceToFileAndPath('C:\Program Files\Autodesk\Autodesk Robot Structural Analysis Professional 2017\System\Exe\Interop.RobotOM.dll')

    # Importing RSA
    import RobotOM as rbt

    # Shortcuts for commonly used commands, in order to minimize the code written
    robapp=rbt.RobotApplicationClass()
    struc=robapp.Project.Structure
    node=struc.Nodes
    bar=struc.Bars
    labels=struc.Labels
    stress=struc.Results.Bars.Stresses
    force=struc.Results.Bars.Forces

    # Calculation preferences
    robapp.Project.CalcEngine.UseStatusWindow=True
    robapp.Project.CalcEngine.AnalysisParams.IgnoreWarnings=True
    robapp.Project.Structure.ResultsFreeze = False

    ################################################################################################################
    ## Input parameters
    rot_stiff_support_input=initialGuess[0]
    main_offset_input=initialGuess[1]
    rot_stiff_trans_input=initialGuess[2]
    trans_offset_input=initialGuess[3]
    ################################################################################################################
    ## APPLYING THE ROTATIONAL STIFFNESS, "rot_stiff_input", TO ALL OF THE ROATIONAL SPRINGS

    print("Applying the parameters")

    # Applying some changes to the support nodes "UZ",  which has a rotational spring in the X direction and    locked displacement in the Z-direction
    support=labels.Get(rbt.IRobotLabelType.I_LT_SUPPORT,"UZ")
    support_data=support.Data
    support_data.HX=rot_stiff_input
```

```
    labels.Store(support)

    # Applying some changes to the support nodes "UZ 2",  which has a rotational spring in the X di
rection      and locked displacement in the X,Y,Z-direction
    support=labels.Get(rbt.IRobotLabelType.I_LT_SUPPORT,"UZ 2")
    support_data=support.Data
    support_data.HX=rot_stiff_input
    labels.Store(support)

    # Applying some changes to the support nodes "Rotational springs", has a rotational spring in t
he           X-direction
    support=labels.Get(rbt.IRobotLabelType.I_LT_SUPPORT,"Rotational spring HX")
    support_data=support.Data
    support_data.HX=rot_stiff_trans_input
    labels.Store(support)

    # Applying some changes to the support nodes "HY",  which has a rotational spring in the Y dire
ction
    support=labels.Get(rbt.IRobotLabelType.I_LT_SUPPORT,"HY")
    support_data=support.Data
    support_data.HY=rot_stiff_support_input
    labels.Store(support)
    ##############################################################################################
#########
    ## APPLYING THE OFFSET, "main_offset_input"

    # Applying the offset to the main girders
    label_test_bar=labels.Get(rbt.IRobotLabelType.I_LT_BAR_OFFSET,"Main girders")
    label_test_bar_data=rbt.IRobotBarOffsetData
    label_test_bar_data=label_test_bar.Data
    label_test_bar_data.Start.UZ=main_offset_input
    label_test_bar_data.End.UZ=main_offset_input
    labels.Store(label_test_bar)

    # Applying the offset to the transversal beams (cross beams)
    label_test_bar=labels.Get(rbt.IRobotLabelType.I_LT_BAR_OFFSET,"crossbeams")
    label_test_bar_data=rbt.IRobotBarOffsetData
    label_test_bar_data=label_test_bar.Data
    label_test_bar_data.Start.UZ=trans_offset_input
    label_test_bar_data.End.UZ=trans_offset_input
    labels.Store(label_test_bar)

    ##############################################################################################
#########
    ## APPLYING THE INTERACTION STIFFNESS, "", TO THE COMPATIBILITY NODES

    # Applying the interaction stiffness to the field section
    comp_node_label=labels.Get(rbt.IRobotLabelType.I_LT_NODE_COMPATIBILITY,"ÄBD Fält")
    comp_node_label_data=rbt.IRobotNodeCompatibilityData
    comp_node_label_data=comp_node_label.Data
    comp_node_label_data.KX=interact_stiff_input
    labels.Store(comp_node_label)

    # Applying the interaction stiffness to the support section
    comp_node_label=labels.Get(rbt.IRobotLabelType.I_LT_NODE_COMPATIBILITY,"ÄBD Stöd")
    comp_node_label_data=rbt.IRobotNodeCompatibilityData
    comp_node_label_data=comp_node_label.Data
    comp_node_label_data.KX=interact_stiff_input
    labels.Store(comp_node_label)

    ##############################################################################################
#########
    ## THE CALCULATION PART

    # Timer
    end = time.time()
```

```
    print("Parameter application time was "+str(end - start)+" "+"seconds")

    # Timer
    print("Starting calc")
    start = time.time()

    # Run the calculations
    robapp.Project.CalcEngine.Calculate()

    # Timer
    end = time.time()
    print("Calculation run time was "+str(end - start)+" "+"seconds")

    ##########################################################################################
    ## RESULT COMPARISON

    # Timer
    start = time.time()

    # Extracting the number of loadcases that exist within the model
    NLC=struc.Cases.GetAll().Count

    # Creating empty lists of lists to store the real stresses at the top and bottom steel fibre
    bar_nu=[ [] for k in range(int(NLC))]
    sigma_top=[ [] for k in range(int(NLC))]
    sigma_bot=[ [] for k in range(int(NLC))]

    # Extracting the real values. They are saved to lists corresponding to different loadcases, the first       (0th) list is for the first loadcase and so forth...
    # Multiplied by negative one to conform to robots force sign convention. (RSA is the other way around,       minus is tension and plus is compression :S)
    for i in range(1,int(NLC)+1):
        filename='case_'+str(i)+'.csv'
        with open(filename, 'r') as csvfile:
            reader=csv.reader(csvfile)
            next(csvfile)
            for row in reader:
                bar_nu[i-1].append(int(row[0]))
                sigma_top[i-1].append(-1*float(row[1]))
                sigma_bot[i-1].append(-1*float(row[2]))

    # Creating empty lists to store the values of the forces and moments to.
    m_y_calc=[ [] for k in range(int(NLC))]
    f_x_calc=[ [] for k in range(int(NLC))]

    # Looping to save all of the forces for the relevant bars. Gets the average forces in the bar.
             force_calc[i][j] corresponds to loadcase "i" and the bar j, which comes from bar_nu[i][j]
    for i in range(NLC):
        for j in range(len(bar_nu[i])):
            temp_bar=bar_nu[i][j]
            f_x_calc[i].append((force.Value(temp_bar,i+1,0).FX+force.Value(temp_bar,i+1,1).FX)/2)
            m_y_calc[i].append((force.Value(temp_bar,i+1,0).MY+force.Value(temp_bar,i+1,1).MY)/2)

    # Creating empty lists to store the cross sectional values to
    AX=[ [] for k in range(int(NLC))]       # Cross section area
    IY=[ [] for k in range(int(NLC))]       # Second moment of inertia
    VZ=[ [] for k in range(int(NLC))]       # Lever arm to the top fibre
    VPZ=[ [] for k in range(int(NLC))]      # Lever arm to the bottom fibre

    # Extracting the cross sectional data for the bars in question.
    for i in range(int(NLC)):
        for j in range(len(bar_nu[i])):
            # Selecting the bar number form the list of bars
            temp_bar_number=bar_nu[i][j]
```

```
176.
177.             # Selecting the bar, for bar number, temp_bar_number
178.             temp_bar=bar.Get(temp_bar_number)
179.
180.             # Getting the label for the section of the bar in question
181.             temp_bar_section_name=temp_bar.GetLabelName(rbt.IRobotLabelType.I_LT_BAR_SECTION)
182.
183.             # Selecting the label for the section of the bar in question
184.             temp_bar_label=labels.Get(rbt.IRobotLabelType.I_LT_BAR_SECTION,temp_bar_section_name)
185.
186.             # Accessing the label data
187.             temp_bar_label_data=temp_bar_label.Data
188.
189.             # Getting the cross section and the second moment of inertia
190.             temp_bar_AX=temp_bar_label_data.GetValue(rbt.IRobotBarSectionDataValue.I_BSDV_AX)
191.             temp_bar_IY=temp_bar_label_data.GetValue(rbt.IRobotBarSectionDataValue.I_BSDV_IY)
192.             temp_bar_VZ=temp_bar_label_data.GetValue(rbt.IRobotBarSectionDataValue.I_BSDV_VZ)
193.             temp_bar_VPZ=temp_bar_label_data.GetValue(rbt.IRobotBarSectionDataValue.I_BSDV_VPZ)
194.
195.             # Appending the values to the list
196.             AX[i].append(temp_bar_AX)
197.             IY[i].append(temp_bar_IY)
198.             VZ[i].append(temp_bar_VZ)
199.             VPZ[i].append(temp_bar_VPZ)
200.
201.     # Creating empty lists of lists to store the calculated stresses to.
202.     sigma_top_calc=[ [] for k in range(int(NLC))]
203.     sigma_bot_calc=[ [] for k in range(int(NLC))]
204.
205.     # Calculating the stress at the top and bottom of the girders
206.     for i in range(NLC):
207.         for j in range(len(bar_nu[i])):
208.             sigma_top_calc[i].append((m_y_calc[i][j]/IY[i][j])*VZ[i][j]+f_x_calc[i][j]/AX[i][j])
209.             sigma_bot_calc[i].append((m_y_calc[i][j]/IY[i][j])*(-
    VPZ[i][j])+f_x_calc[i][j]/AX[i][j])
210.
211.     # Calculating the root mean squared error (RMSE) and mean absolute error (MAE).
212.     # Summing up the number of measurement points, it is added by two because we add two RMSE per r
    un in the    loop above.
213.     n_eval=0
214.     RMSD=0
215.     MAE=0
216.     MBE=0
217.     for i in range(NLC):
218.         for j in range(len(bar_nu[i])):
219.             RMSD=RMSD+(sigma_top[i][j]-sigma_top_calc[i][j])**2+(sigma_bot[i][j]-
    sigma_bot_calc[i][j])**2
220.             MAE=MAE+abs(sigma_bot[i][j]-sigma_bot_calc[i][j])+abs(sigma_top[i][j]-
    sigma_top_calc[i][j])
221.             MBE=MBE+(sigma_bot[i][j]-sigma_bot_calc[i][j])+(sigma_top[i][j]-sigma_top_calc[i][j])
222.             n_eval=n_eval+2
223.
224.     RMSD=(RMSD/n_eval)**0.5
225.     MAE=MAE/n_eval
226.     MBE=MBE/n_eval
227.
228.     with open('evaluation.csv','ab') as csvfile:
229.         fieldnames=['rot_stiff_support_input','main_offset_input','rot_stiff_trans_input','trans_of
    fset_input','MBE','MAE','RMSD']
230.         writer=csv.DictWriter(csvfile,fieldnames)
231.         writer.writerow({fieldnames[0]:rot_stiff_support_input,
232.                          fieldnames[1]:main_offset_input,
233.                          fieldnames[2]:rot_stiff_trans_input,
234.                          fieldnames[3]:trans_offset_input,
235.                          fieldnames[4]:MBE,
236.                          fieldnames[5]:MAE,
```

```
                        fieldnames[6]:RMSD})

    # Timer
    end = time.time()
    print("Post processing time was "+str(end - start)+" "+"seconds")

    print("Root mean squared error: ",RMSD/1000000, "MPa")
    return(RMSD)
```

Appendix G – Single span calibration script

G.1 – The main script for the single span optimization

```
# Importing some stuff
import clr

# Importing the simple beam creation and the target function
import sbc
import tf

# Adding references to the Extreme Optimization package
clr.AddReferenceToFileAndPath("C:\\Program Files (x86)\\Extreme Optimization\\Numerical Libraries for .NET 6.0\\bin\\Net40\\Extreme.Numerics.Net40.dll")
clr.AddReferenceToFileAndPath("C:\\Program Files (x86)\\IronPython 2.7\\DLLs\\Extreme.Numerics.IronPython.dll")

# Importing the optimization function
from Extreme.Mathematics.Optimization import *
from Extreme.Mathematics.LinearAlgebra import *
from Extreme.Mathematics import *

# Number of bars, each 1 meter long, for the beam creation
nb=6

# Calling sbc (simple beam creation)
sbc.simple_beam_creation(nb)

# Starting guesses
start_value_KZ=float(1000000)
start_value_HY=float(10*start_value_KZ)

# Arranging them into a vector
initialGuess=Vector.Create[float](2)
initialGuess[0]=start_value_KZ
initialGuess[1]=start_value_HY

# Shortcut to the Nelder-
Mead optimization function. The method is implemented by the NelderMeadOptimizer class:
nm=NelderMeadOptimizer()

# The class has three special properties, that help determine
# the progress of the algorithm. These parameters have
# default values and need not be set explicitly.
nm.ContractionFactor=0.5
nm.ExpansionFactor=2
nm.ReflectionFactor=-2
nm.SolutionTest.AbsoluteTolerance = 1e-2
nm.InitialGuess = initialGuess
nm.MinIterations=30

# Targetfunction
nm.ObjectiveFunction = tf.target_function

nm.FindExtremum()

print "Nelder-Mead Method:"
print "  Solution:", nm.Extremum
print "  Estimated error:", nm.EstimatedError
print "  # iterations:", nm.IterationsNeeded
print "  # function evaluations:", nm.EvaluationsNeeded
```

G.2 – The single span beam creation script

```
def simple_beam_creation(nb):
    # Importing some stuff
    import clr
    import math
    import sys
    import csv

    # Accessing RSA
    clr.AddReferenceToFileAndPath('C:\Program Files\Autodesk\Autodesk Robot Structural Analysis Professional 2017\System\Exe\Interop.RobotOM.dll')

    # Importing RSA
    import RobotOM as rbt

    # Shortcuts for commonly used commands, in order to minimize the code written
    robapp=rbt.RobotApplicationClass()
    struc=robapp.Project.Structure
    node=struc.Nodes
    bar=struc.Bars
    labels=struc.Labels
    stress=struc.Results.Bars.Stresses
    force=struc.Results.Bars.Forces

    # Print RSA version and the python version below:
    print('Robot program version:',robapp.ProgramVersion)
    print('Python program version:',sys.version)

    # Close any open project
    robapp.Project.Close

    # Ensuring that RSA is visible and interactive
    robapp.Visible=1
    robapp.Interactive=1

    # Start a new project
    Robproj=robapp.Project.New(rbt.IRobotProjectType.I_PT_FRAME_2D)

    # Number of bars, HAS TO BE EVEN NUMBER
    # Creating nodes
    for i in range(nb+1):
        node.Create(i+1,i,0,0)
    # Creates bars
    for i in range(nb):
        bar.Create(i+1,i+1,i+2)

    # Creating a label, which corresponds to the suports
    support=labels.Create(rbt.IRobotLabelType.I_LT_SUPPORT,"support")
    support_data=rbt.IRobotNodeSupportData      # Needed to see the suggestions for support_data.XX

    support_data=support.Data
    support_data.UX=1
    support_data.UY=1
    support_data.UZ=0
    support_data.KZ=10000000
    support_data.RX=0
    support_data.RY=0
    support_data.HY=support_data.KZ*10
    support_data.RZ=0
    labels.Store(support)

    # Creating a label which corresponds to the bar sections
    bar_label=labels.Create(rbt.IRobotLabelType.I_LT_BAR_SECTION,"HEA100")
    labels.Store(bar_label)


```

```
    ## Creating groups for the nodes
    # Support nodes:
    struc.Groups.Create(rbt.IRobotObjectType.I_OT_NODE,"support_nodes",str(1)+" "+str(nb+1))
    index_support_nodes=struc.Groups.Find(rbt.IRobotObjectType.I_OT_NODE,"support_nodes")
    RGR_support_nodes=struc.Groups.Get(rbt.IRobotObjectType.I_OT_NODE,index_support_nodes)
    # Mid node:
    struc.Groups.Create(rbt.IRobotObjectType.I_OT_NODE,"mid_node",str(nb/2+1))
    index_mid_node=struc.Groups.Find(rbt.IRobotObjectType.I_OT_NODE,"mid_node")
    RGR_mid_node=struc.Groups.Get(rbt.IRobotObjectType.I_OT_NODE,index_mid_node)

    # Selecting all the nodes
    selection_nodes=struc.Selections.Get(rbt.IRobotObjectType.I_OT_NODE)
    # Choosing a subset for the support nodes and applying the label corresponding to the support t
o them
    selection_nodes.FromText(RGR_support_nodes.SelList)
    node.SetLabel(selection_nodes,rbt.IRobotLabelType.I_LT_NODE_SUPPORT,"support")

    # Selecting all the bars
    selection_bars=struc.Selections.Get(rbt.IRobotObjectType.I_OT_BAR)
    selection_bars.FromText("All")
    # Setting the cross section to HEA100
    bar.SetLabel(selection_bars,rbt.IRobotLabelType.I_LT_BAR_SECTION,"HEA100")
    # Setting the material to STEEL, allready defined in the program
    bar.SetLabel(selection_bars,rbt.IRobotLabelType.I_LT_BAR_MATERIAL,"STEEL")

    # Defining loadcases, selfweight and a pointload in the mid span
    # Selfweight
    caseSW=rbt.IRobotSimpleCase
    caseSW=struc.Cases.CreateSimple(1,"SW",rbt.IRobotCaseNature.I_CN_PERMANENT,rbt.IRobotCaseAnaliz
eType.I_CAT_STATIC_LINEAR)
    caseSW.Records.New(rbt.IRobotLoadRecordType.I_LRT_DEAD)
    loadrec=rbt.IRobotLoadRecord
    loadrec=caseSW.Records.Get(1)
    loadrec.SetValue(2,1)
    loadrec.SetValue(15,True)

    # Pointload in the middle of the span
    caselive=rbt.IRobotSimpleCase
    caselive=struc.Cases.CreateSimple(2,"Pointload",rbt.IRobotCaseNature.I_CN_EXPLOATATION,rbt.IRob
otCaseAnalizeType.I_CAT_STATIC_LINEAR)
    caselive.Records.New(rbt.IRobotLoadRecordType.I_LRT_BAR_FORCE_CONCENTRATED)
    loadrec=rbt.IRobotLoadRecord
    loadrec=caselive.Records.Get(1)
    loadrec.SetValue(0,0)
    loadrec.SetValue(1,0)
    loadrec.SetValue(2,-10000)
    loadrec.Objects.FromText(RGR_mid_node.SelList)

    # Setting the program to show the calculation window and ignore warnings
    robapp.Project.CalcEngine.UseStatusWindow=True
    robapp.Project.CalcEngine.AnalysisParams.IgnoreWarnings=True
    robapp.Project.CalcEngine.AutoFreezeResults=False

    robapp.Project.CalcEngine.Calculate()

    # Saving the desired stresses to an external CSV file
    with open('stresses_saved.csv', 'w') as csvfile:
        fieldnames=['bar','s_min_start','s_min_end','s_max_start','s_max_end']
        writer=csv.DictWriter(csvfile,fieldnames)
        writer.writeheader()
        for i in range(1,nb+1):
            writer.writerow({fieldnames[0]:i,
                             fieldnames[1]:stress.Value(i,2,0).Smin,
                             fieldnames[2]:stress.Value(i,2,1).Smin,
                             fieldnames[3]:stress.Value(i,2,0).Smax,
                             fieldnames[4]:stress.Value(i,2,1).Smax})
```

```
127.
128.     # Saving the desired FORCES to an external CSV file
129.     with open('forces_saved.csv', 'w') as csvfile:
130.         fieldnames=['bar','FX_start','FX_end','MY_start','MY_end']
131.         writer=csv.DictWriter(csvfile,fieldnames)
132.         writer.writeheader()
133.         for i in range(1,nb+1):
134.             writer.writerow({fieldnames[0]:i,
135.                              fieldnames[1]:force.Value(i,2,0).FX,
136.                              fieldnames[2]:force.Value(i,2,1).FX,
137.                              fieldnames[3]:force.Value(i,2,0).MY,
138.                              fieldnames[4]:force.Value(i,2,1).MY})
```

G.3 – The target function for the single span beam optimization

```
def target_function(initialGuess):
    # Importing some stuff
    import clr
    import math
    import sys
    import csv

    # Accessing RSA
    clr.AddReferenceToFileAndPath('C:\Program Files\Autodesk\Autodesk Robot Structural Analysis Professional 2017\System\Exe\Interop.RobotOM.dll')

    # Importing RSA
    import RobotOM as rbt

    # Shortcuts for commonly used commands, in order to minimize the code written
    robapp=rbt.RobotApplicationClass()
    struc=robapp.Project.Structure
    node=struc.Nodes
    bar=struc.Bars
    labels=struc.Labels
    stress=struc.Results.Bars.Stresses
    force=struc.Results.Bars.Forces

    # Print RSA version and the python version below:
    print('Robot program version:',robapp.ProgramVersion)
    print('Python program version:',sys.version)

    # BARS THAT SHOULD BE INVESTIGATED
    what_bars=[3,6]

    input_KZ=initialGuess[0]
    input_HY=initialGuess[1]

    # Applying the changes to the support
    support=labels.Get(rbt.IRobotLabelType.I_LT_SUPPORT,"support")
    support_data=support.Data
    support_data.KZ=input_KZ
    support_data.HY=input_HY
    labels.Store(support)


    # Setting the program to show the calculation window and ignore warnings
    robapp.Project.CalcEngine.UseStatusWindow=True
#    robapp.Project.CalcEngine.AnalysisParams.IgnoreWarnings=True
#    robapp.Project.CalcEngine.AutoFreezeResults=False

    # Running the calculations
    robapp.Project.CalcEngine.Calculate()

    # Creating emty lists to save the stresses to
    bar_nu=[]
    fx_0=[]
    fx_1=[]
    my_0=[]
    my_1=[]

    # TESTING READING VALUES FROM CSV FILE
    # Saving the desired FORCES to an external CSV file
    with open('forces_saved.csv', 'r') as csvfile:
        reader=csv.reader(csvfile)
        next(csvfile)
        for row in reader:
            bar_nu.append(float(row[0]))
            fx_0.append(float(row[1]))
            fx_1.append(float(row[2]))
```

```
            my_0.append(float(row[3]))
            my_1.append(float(row[4]))

    # Calculating the root mean squared error (RMSE), mean absolute percentage error (MAPE)
    MAE=0
    MBE=0
    RMSD=0
    for i in range(len(what_bars)):
        temp_bar=what_bars[i]
        temp_bar_i=bar_nu.index(temp_bar)
        RMSD=RMSD+(fx_0[temp_bar_i]-force.Value(temp_bar,2,0).FX)**2 + (fx_1[temp_bar_i]-
force.Value(temp_bar,2,1).FX)**2 + (my_0[temp_bar_i]-
force.Value(temp_bar,2,0).MY)**2 + (my_1[temp_bar_i]-force.Value(temp_bar,2,1).MY)**2

        MAE=MAE+abs(fx_0[temp_bar_i]-force.Value(temp_bar,2,0).FX) + abs(fx_1[temp_bar_i]-
force.Value(temp_bar,2,1).FX) + abs(my_0[temp_bar_i]-
force.Value(temp_bar,2,0).MY) + abs(my_1[temp_bar_i]-force.Value(temp_bar,2,1).MY)

        MBE=MBE+(fx_0[temp_bar_i]-force.Value(temp_bar,2,0).FX) + (fx_1[temp_bar_i]-
force.Value(temp_bar,2,1).FX) + (my_0[temp_bar_i]-
force.Value(temp_bar,2,0).MY) + (my_1[temp_bar_i]-force.Value(temp_bar,2,1).MY)

    MBE=MBE/(4*len(what_bars))
    MAE=MAE/(4*len(what_bars))
    RMSD=(RMSD/(4*len(what_bars)))**0.5


    with open('evaluation.csv','ab') as csvfile:
        fieldnames=['input_KZ','input_HY','MBE','MAE','RMSD']
        writer=csv.DictWriter(csvfile,fieldnames)
        writer.writerow({fieldnames[0]:input_KZ,
                         fieldnames[1]:input_HY,
                         fieldnames[2]:MBE,
                         fieldnames[3]:MAE,
                         fieldnames[4]:RMSD})

    print("Root mean square error:",RMSD)
    return(RMSD)
```

Appendix H – Multi span beam calibration scripts

H.1 – The main script for the multi span optimization

```
# Importing some stuff
import clr

# Importing the multi span creation script
import msc
import tf

# Adding references to the Extreme Optimization package
clr.AddReferenceToFileAndPath("C:\\Program Files (x86)\\Extreme Optimization\\Numerical Libraries for .NET 6.0\\bin\\Net40\\Extreme.Numerics.Net40.dll")
clr.AddReferenceToFileAndPath("C:\\Program Files (x86)\\IronPython 2.7\\DLLs\\Extreme.Numerics.IronPython.dll")

# Importing the optimization function
from Extreme.Mathematics.Optimization import *
from Extreme.Mathematics.LinearAlgebra import *
from Extreme.Mathematics import *

# Set these values
ns=float(5)             # number of spans. MODIFY TARGET FUNCTION AND INSERT IT THERE ASWELL
nb=float(10)            # number of bars, per span, HAS TO BE EVEN!
ls=float(5)             # length of each span
KZ_supp=10000000        # vertical elastic stiffness of support
HY_supp=90000000        # rotational elastic stiffness at support
E_stal=210e9            # E-modulus of steel
CS="IPE 400"            # Cross section of beam
Load_Z=-50000           # Load on the beam

# Calling on the multi span creation scipt, which applies the above stated input
msc.multi_span_creation(ns,nb,ls,KZ_supp,HY_supp,E_stal,CS,Load_Z)

# Arranging them into a vector
initialGuess=Vector.Create[float](2)
initialGuess[0]=KZ_supp*0.5
initialGuess[1]=HY_supp*1.5

# Shortcut to the Nelder-Mead optimization function. The method is implemented by the NelderMeadOptimizer class:
nm=NelderMeadOptimizer()

# The class has three special properties, that help determine
# the progress of the algorithm. These parameters have
# default values and need not be set explicitly.
nm.ContractionFactor = 0.5
nm.ExpansionFactor = 2
nm.ReflectionFactor = -2

# Limiting the iteration counts, so that the optimzation gets going, aswell as that it does not run for an eternity
nm.MinIterations=50
nm.MaxIterations=250

nm.SolutionTest.AbsoluteTolerance = 1e-18
nm.InitialGuess = initialGuess

# Targetfunction
nm.ObjectiveFunction = tf.target_function

# Initiate
nm.FindExtremum()

print "Nelder-Mead Method:"
```

```
59.  print "  Solution:", nm.Extremum
60.  print "  Estimated error:", nm.EstimatedError
61.  print "  # iterations:", nm.IterationsNeeded
62.  print "  # function evaluations:", nm.EvaluationsNeeded
```

H.2 – The multi span beam creation script

```
def multi_span_creation(ns,nb,ls,KZ_supp,HY_supp,E_stal,CS,Load_Z):
    # Importing some stuff
    import clr
    import math
    import sys
    import csv

    # Accessing RSA
    clr.AddReferenceToFileAndPath('C:\Program Files\Autodesk\Autodesk Robot Structural Analysis Professional 2017\System\Exe\Interop.RobotOM.dll')

    # Importing RSA
    import RobotOM as rbt

    # Shortcuts for commonly used commands, in order to minimize the code written
    robapp=rbt.RobotApplicationClass()
    struc=robapp.Project.Structure
    node=struc.Nodes
    bar=struc.Bars
    labels=struc.Labels
    stress=struc.Results.Bars.Stresses
    force=struc.Results.Bars.Forces

    # Print RSA version and the python version below:
    print('Robot program version:',robapp.ProgramVersion)
    print('Python program version:',sys.version)

    # Close any open project
    robapp.Project.Close

    # Start a new project
    Robproj=robapp.Project.New(rbt.IRobotProjectType.I_PT_FRAME_2D)

    # Calculating the delta x increment which separates each node.
    dx=float(ls*ns/(nb*ns))

    # Creating all the nodes
    for i in range(int(ns*nb)+1):
        node.Create(i+1,dx*float(i),0,0)

    # Creating all the bars
    for i in range(int(ns*nb)):
        bar.Create(i+1,i+1,i+2)

    # Creating a label, which corresponds to the suports
    support=labels.Create(rbt.IRobotLabelType.I_LT_SUPPORT,"support")
    support_data=rbt.IRobotNodeSupportData      # Needed to see the suggestions for support_data.XX

    support_data=support.Data
    support_data.UX=1
    support_data.UY=1
    support_data.UZ=0
    support_data.KZ=KZ_supp
    support_data.RX=0
    support_data.RY=0
    support_data.HY=HY_supp
    support_data.RZ=0
    labels.Store(support)

    # Creating a label which corresponds to the bar sections
    bar_label=labels.Create(rbt.IRobotLabelType.I_LT_BAR_SECTION,CS)
    labels.Store(bar_label)

    # Creating a string for the supportnodes
    supp_nodes=range(1,int(ns*nb+2),int(nb))
```

```
    supp_nodes_str=str()
    for i in range(len(supp_nodes)):
        supp_nodes_str=supp_nodes_str+" "+str(supp_nodes[i])
    # Creating a string for the mid nopes
    mid_nodes=range(int(nb/2)+1,int(nb*ns),int(nb))
    mid_nodes_str=str()
    for i in range(len(mid_nodes)):
        mid_nodes_str=mid_nodes_str+" "+str(mid_nodes[i])

    ## Creating groups for the nodes
    # Support nodes:
    struc.Groups.Create(rbt.IRobotObjectType.I_OT_NODE,"support_nodes",supp_nodes_str)
    index_support_nodes=struc.Groups.Find(rbt.IRobotObjectType.I_OT_NODE,"support_nodes")
    RGR_support_nodes=struc.Groups.Get(rbt.IRobotObjectType.I_OT_NODE,index_support_nodes)
    # Mid nodes:
    struc.Groups.Create(rbt.IRobotObjectType.I_OT_NODE,"mid_nodes",mid_nodes_str)
    index_mid_nodes=struc.Groups.Find(rbt.IRobotObjectType.I_OT_NODE,"mid_nodes")
    RGR_mid_nodes=struc.Groups.Get(rbt.IRobotObjectType.I_OT_NODE,index_mid_nodes)

    # Selecting all the nodes
    selection_nodes=struc.Selections.Get(rbt.IRobotObjectType.I_OT_NODE)
    # Choosing a subset for the support nodes and applying the label corresponding to the support t
o them
    selection_nodes.FromText(RGR_support_nodes.SelList)
    node.SetLabel(selection_nodes,rbt.IRobotLabelType.I_LT_NODE_SUPPORT,"support")

    # Selecting all the bars
    selection_bars=struc.Selections.Get(rbt.IRobotObjectType.I_OT_BAR)
    selection_bars.FromText("All")
    # Setting the cross section to HEA 300
    bar.SetLabel(selection_bars,rbt.IRobotLabelType.I_LT_BAR_SECTION,CS)

    # Defining steel material, calling it "stal" to differentiate it from the others
    ste_mat_label=labels.Create(rbt.IRobotLabelType.I_LT_MATERIAL,"stal")
    SteelMaterial=rbt.IRobotMaterialData
    SteelMaterial=ste_mat_label.Data
    SteelMaterial.Type=rbt.IRobotMaterialType.I_MT_STEEL
    SteelMaterial.E=E_stal
    SteelMaterial.NU=0.3
    SteelMaterial.RO=8000
    SteelMaterial.Kirchoff=SteelMaterial.E/(2*(1+SteelMaterial.NU))
    labels.Store(ste_mat_label)

    # Setting the material to "stal", allready defined in the program
    bar.SetLabel(selection_bars,rbt.IRobotLabelType.I_LT_BAR_MATERIAL,"stal")

    cases=[None]*int(ns)
    for i in range(int(ns)):
        cases[i]="caselive_"+str(i)

    for i in range(len(mid_nodes)):
        case_i=cases[i]
        case_i=rbt.IRobotSimpleCase
        case_i=struc.Cases.CreateSimple(int(i+1),"Pointload"+" "+str(i+1),rbt.IRobotCaseNature.I_CN
_EXPLOATATION,rbt.IRobotCaseAnalizeType.I_CAT_STATIC_LINEAR)
        case_i.Records.New(rbt.IRobotLoadRecordType.I_LRT_BAR_FORCE_CONCENTRATED)
        loadrec=rbt.IRobotLoadRecord
        loadrec=case_i.Records.Get(1)
        loadrec.SetValue(0,0)
        loadrec.SetValue(1,0)
        loadrec.SetValue(2,Load_Z)
        loadrec.Objects.FromText(str(mid_nodes[i]))

    # Setting the program to show the calculation window and ignore warnings
    robapp.Project.CalcEngine.UseStatusWindow=True
    robapp.Project.CalcEngine.AnalysisParams.IgnoreWarnings=True
```

```
    robapp.Project.CalcEngine.AutoFreezeResults=False

    # Run the calculations
    robapp.Project.CalcEngine.Calculate()

    # Saves all the forces for all the bars for all the load cases to csv files. Here the number of load cases that are saved for is based on the number of spans, which is equal to the number of loadcases.
    for k in range(1,int(ns)+1):
        with open('forces_saved'+str(k)+'.csv', 'w') as csvfile:
            fieldnames=['bar','FZ_start','FZ_end','MY_start','MY_end']
            writer=csv.DictWriter(csvfile,fieldnames)
            writer.writeheader()
            for i in range(1,int(nb*ns)+1):
                writer.writerow({fieldnames[0]:i,
                                 fieldnames[1]:force.Value(i,k,0).FZ,
                                 fieldnames[2]:force.Value(i,k,1).FZ,
                                 fieldnames[3]:force.Value(i,k,0).MY,
                                 fieldnames[4]:force.Value(i,k,1).MY})

    for k in range(1,int(ns)+1):
        with open('stresses_saved'+str(k)+'.csv', 'w') as csvfile:
            fieldnames=['bar','s_min_start','s_min_end','s_max_start','s_max_end']
            writer=csv.DictWriter(csvfile,fieldnames)
            writer.writeheader()
            for i in range(1,int(nb*ns)+1):
                writer.writerow({fieldnames[0]:i,
                                 fieldnames[1]:stress.Value(i,k,0).Smin,
                                 fieldnames[2]:stress.Value(i,k,1).Smin,
                                 fieldnames[3]:stress.Value(i,k,0).Smax,
                                 fieldnames[4]:stress.Value(i,k,1).Smax})
```

H.3 – The target function for the multi span beam optimization

```
def target_function(initialGuess):
    # Importing some stuff.
    import clr
    import math
    import sys
    import csv

    # Accessing RSA.
    clr.AddReferenceToFileAndPath('C:\Program Files\Autodesk\Autodesk Robot Structural Analysis Professional 2017\System\Exe\Interop.RobotOM.dll')

    # Importing RSA.
    import RobotOM as rbt

    # Milestone.
    print('Started')

    ##########################################################################################################
    # NUMBER OF SPANS AND WHICH BARS THAT ARE OF INTEREST
    ns=float(5)
    what_bars=[22,26,28,29]
    ##########################################################################################################

    # Shortcuts for commonly used commands, in order to minimize the code written.
    robapp=rbt.RobotApplicationClass()
    struc=robapp.Project.Structure
    node=struc.Nodes
    bar=struc.Bars
    labels=struc.Labels
    stress=struc.Results.Bars.Stresses
    force=struc.Results.Bars.Forces

    # Assigning th inpudata to parameters in the script
    input_KZ=float(initialGuess[0])
    input_HY=float(initialGuess[1])
    # E_stal_input=210e9      #float(initialGuess[2])

    # Applying changes to the support stiffnessess.
    support=labels.Get(rbt.IRobotLabelType.I_LT_SUPPORT,"support")
    support_data=support.Data
    support_data.KZ=input_KZ
    support_data.HY=input_HY
    labels.Store(support)

    # Milestone
    print('Applied changes, running calculations')

    # Running calculations and showing calculation window.
    robapp.Project.CalcEngine.UseStatusWindow=True
    robapp.Project.CalcEngine.Calculate()

    # Creating lists of lists to extract the real values to.
    bar_nu=[ [] for k in range(int(ns))]
    fz_0=[ [] for k in range(int(ns))]
    fz_1=[ [] for k in range(int(ns))]
    my_0=[ [] for k in range(int(ns))]
    my_1=[ [] for k in range(int(ns))]

    # Extracting the real values. They are saved to lists corresponding to different loadcases, the first (0th) list is for the first loadcase (when the load is the middle of the first span).
    for i in range(1,int(ns)+1):
        filename='forces_saved'+str(i)+'.csv'
        with open(filename, 'r') as csvfile:
```

```
            reader=csv.reader(csvfile)
            next(csvfile)
            for row in reader:
                bar_nu[i-1].append(float(row[0]))
                fz_0[i-1].append(float(row[1]))
                fz_1[i-1].append(float(row[2]))
                my_0[i-1].append(float(row[3]))
                my_1[i-1].append(float(row[4]))

    # Calculating the root mean squared error (RMSE), mean absolute percentage error (MAPE)
    MAE=0
    MBE=0
    RMSD=0
    for i in range(len(what_bars)):
        temp_bar=what_bars[i]
        for j in range(int(ns)):
            bar_nu_temp=bar_nu[j]
            temp_bar_i=bar_nu_temp.index(temp_bar)
            RMSD=RMSD+(fz_0[j][temp_bar_i]-
force.Value(temp_bar,j+1,0).FZ)**2 + (fz_1[j][temp_bar_i]-
force.Value(temp_bar,j+1,1).FZ)**2 + (my_0[j][temp_bar_i]-
force.Value(temp_bar,j+1,0).MY)**2 + (my_1[j][temp_bar_i]-force.Value(temp_bar,j+1,1).MY)**2

            MAE=MAE+abs(fz_0[j][temp_bar_i]-
force.Value(temp_bar,j+1,0).FZ) + abs(fz_1[j][temp_bar_i]-
force.Value(temp_bar,j+1,1).FZ) + abs(my_0[j][temp_bar_i]-
force.Value(temp_bar,j+1,0).MY) + abs(my_1[j][temp_bar_i]-force.Value(temp_bar,j+1,1).MY)

            MBE=MBE+(fz_0[j][temp_bar_i]-force.Value(temp_bar,j+1,0).FZ) + (fz_1[j][temp_bar_i]-
force.Value(temp_bar,j+1,1).FZ) + (my_0[j][temp_bar_i]-
force.Value(temp_bar,j+1,0).MY) + (my_1[j][temp_bar_i]-force.Value(temp_bar,j+1,1).MY)

    MBE=MBE/(4*len(what_bars))
    MAE=MAE/(4*len(what_bars))
    RMSD=(RMSD/(4*len(what_bars)))**0.5

    with open('evaluation.csv','ab') as csvfile:
        fieldnames=['input_KZ','input_HY','MBE','MAE','RMSD']
        writer=csv.DictWriter(csvfile,fieldnames)
        writer.writerow({fieldnames[0]:input_KZ,
                         fieldnames[1]:input_HY,
                         fieldnames[2]:MBE,
                         fieldnames[3]:MAE,
                         fieldnames[4]:RMSD})


    print("Root mean squared error: ",RMSD)
    return(RMSD)
```

Appendix I – Measurement processing scripts

I.1 – Main script of the fatigue evaluation

The main script that initialized the other scripts for the fatigue evaluation based on measurements. The other scripts for this is given in I.2, I.3, I.4 and I.5.

```
# In[Importing]:

# Import stuffs
import Temperature_Correction
import Cycle_Counting
import Equivalent_Stress
import Damage_Accumulation

# Benchmarking the script
import time
time_start=time.time()

# In[Settings]:

# Set this to where the TWO CSV files are located, one for strains and one for temperature.
inputFolder = "C:\\Users\\A549153\\Desktop\\code\\Compiled evaluation\\2018-04-25 final\\Input\\"
# List of the TWO CSV files in the folder.
CsvList=['Strains.csv','Temps.csv']
# Strains.CSV should be a list of only time and strains, with the columns ['time','strain']
# Temps.csv  should be a list of only time and temperatures, with the columns ['time','temp']

# Set this to where the CSV file contianing the stress ranges and cycles should be saved.
outputfolder = "C:\\Users\\A549153\\Desktop\\code\\Compiled evaluation\\2018-04-
25 final\\Intermittent\\"
# Name it.
outputfilename="stress_and_cycles.csv"

# In[Script settings]:
################################################################################
# Plotting settings:
# Stress range
sr_adjusted_plot=1
sr_unadjusted_plot=1

# Rainflow histogram plotting, it also desides if the unadjusted should be
# calculated for, if not then the returned values are [].
RC_adjusted_plot=1
RC_unadjusted_plot=1

# Bin size for the stress histogram
bin_size=2

# Limit value. The threshold below which the values are removed, not
# neccesarily the same as the cut off limit, used to get better plots
# initially, but for the calculations of the equivalent stress it does not
# influence (unless set higher than DeltaSigma_L which it should NOT be).
# Preferably set to bin_size so that all of the small noisy measurements
# dissapear.
threshold=bin_size
################################################################################
# Calculate damage for adjusted or unadjusted strains
# SET ONLY ONE OF THEM TO ONE
calc_adjusted=1
calc_unadjusted=0

################################################################################
# Rainflow counting settings
# Resolution of the stress range values, defined by ndigits. I.e. rounding of
# values to the number of digits. Desides the resolution in the output file, as
# well as the output lists, containing the stress ranges and number of cycles.
```

```
# It does not influence the histogram plotting.
ndigits=1

# In[Gage parameter data]:

# Gage factor
k=2.155

# TC of gage factor, %/100•C
TC=1.2

# Thermal output parameters, for degrees celcius
A0=-7.33e1
A1=4.82e0
A2=-8.18e-2
A3=3.72e-4
A4=4.13e-7

# In[Fatigue properties]:

# Set sigma_C to the detail category value
sigma_C=float(36)        # In MPa.
sigma_D=0.737*sigma_C   #
sigma_L=0.549*sigma_D   #

# Set these
gamma_Ff=1.00           # Partial factor.
gamma_Mf=1.15           # Partial factor.
m=5                     # Decides which slope the equivalent stress will be
                        # calculated for, choose onlu m=3 or m=5.

# Set these to the aquired constants from the SAF evaluation
SAF_sigma=1             # spatial adjustment factor, for stress range.
SAF_n=1                 # spatial adjustment factor, for number of cycles.

# In[Time remaining]
# Calculates the factor which is used to predict the future caused damage.

end_year=2021
end_month=6
end_time=end_year+end_month/12

evaluated=2015+5/12

time_remaining=end_time-evaluated

# In[Computing the true strains]:

[Combined,start_time,end_time]=Temperature_Correction.TC(inputFolder,CsvList,k,TC,A0,A1,A2,A3,A4,sr
    _adjusted_plot,sr_unadjusted_plot)

# In[Cycle counting]:

[rf_count_adjusted,rf_count_unadjusted]=Cycle_Counting.CC(Combined,ndigits,threshold,outputfolder,o
    utputfilename,RC_adjusted_plot,RC_unadjusted_plot,SAF_sigma,SAF_n,bin_size)

# In[Equivalent stress]:

[sigma_eq,n_eq]=Equivalent_Stress.eq_stress(rf_count_adjusted,rf_count_unadjusted,outputfolder,sigm
    a_C,sigma_D,sigma_L,gamma_Ff,gamma_Mf,m,calc_adjusted,calc_unadjusted)

# In[Damage accumulation calculation]:

D=Damage_Accumulation.dmg_acc(sigma_D,gamma_Ff,gamma_Mf,m,sigma_eq,n_eq,start_time,end_time,time_re
    maining)

```

```
# In[Benchmark]:
time_end=time.time()
print('Computational time was',time_end-time_start)
```

I.2 – Temperature correction script

The script that deals with the temperature correction of the strains, based on the manufacturers guidelines.

```
# In[]:
# Name the csv files "Strains.csv" and "Temps.csv"
# Made to read two files, one for temperatures, and one for strains

def TC(inputFolder,CsvList,k,TC,A0,A1,A2,A3,A4,sr_adjusted_plot,sr_unadjusted_plot):
    # In[Importing]:

    import pandas as pd

    # In[Reading CSV]:

    CsvList_path=[None]*len(CsvList)
    # Combining the folder path and the CSV file name
    for i in range(len(CsvList)):
        CsvList_path[i]=inputFolder+CsvList[i]

    # In[reading CSV files data]:

    StrainRawData=pd.DataFrame.from_csv(CsvList_path[0])

    TempRawData=pd.DataFrame.from_csv(CsvList_path[1])

    # In[Combining and interpolating]:

    # Combining
    Combined=TempRawData.combine_first(StrainRawData)

    # Interpolating
    Combined=Combined.interpolate(method='linear')


    # In[Removing the not needed values]:

    # Reset index
    StrainRawData.reset_index(inplace=True)

    # Get boundaries for time, based on first and last strain measurements
    start_time=StrainRawData['time'][0]
    end_time=StrainRawData['time'][len(StrainRawData)-1]

    ## Reset index
    Combined.reset_index(inplace=True)

    # Remove unwanted values
    Combined.drop(Combined[Combined.time < start_time].index, inplace=True)
    Combined.drop(Combined[Combined.time > end_time].index, inplace=True)

    # In[Calculating the correction factors]:

    # Thermal output correction factor
    Combined['Thermal_output_corr']=( A0 +
                                      A1 * (Combined.temp) +
                                      A2 * (Combined.temp**2) +
                                      A3 * (Combined.temp**3) +
                                      A4 * (Combined.temp**4) )

    # Gage correction factor
    Combined['Gage_factor_corr']=2/(k*(1-(TC*(Combined.temp-24)/10000)))

    # In[Calculating the correct strain measurement]:

```

```
    Combined['Temperature_Adjusted_strains']=((Combined['strain'] -
                                              Combined['Thermal_output_corr']) *
                                              Combined['Gage_factor_corr'] )

    # In[Plotting]:

    if sr_adjusted_plot==1 and sr_unadjusted_plot==1:
        Combined_plot=Combined.plot(y=['strain','Temperature_Adjusted_strains'],x=['time'],grid=Tru
e,title='Strain history comparison')
        Combined_plot.set_xlabel('Time in MM-DD HH')
        Combined_plot.set_ylabel('Strain, micrometer/meter')
        Combined_plot.legend(['Measured strains','Temperature adjusted strains'])
    elif sr_adjusted_plot==1 and sr_unadjusted_plot==0:
        Combined_plot=Combined.plot(y=['Temperature_Adjusted_strains'],grid=True,title='Strain hist
ory')
        Combined_plot.set_xlabel('Time in MM-DD HH')
        Combined_plot.set_ylabel('Strain, micrometer/meter')
        Combined_plot.legend(['Temperature adjusted strains'])
    elif sr_adjusted_plot==0 and sr_unadjusted_plot==1:
        Combined_plot=Combined.plot(y=['strain'],grid=True,title='Strain history')
        Combined_plot.set_xlabel('Time in MM-DD HH')
        Combined_plot.set_ylabel('Strain, micrometer/meter')
        Combined_plot.legend(['Measured strains'])

    # In[Returning complete ]:

    return(Combined,start_time,end_time)
```

I.3 – Cyclic counting script

The script that prepares the strain history, or histories, for the rainflow counting script.

```
def CC(Combined,ndigits,threshold,outputfolder,outputfilename,RC_adjusted,RC_unadjusted,SAF_sigma,SAF_n,bin_size):

    # In[Importing]:
    import numpy as np
    import math
    import matplotlib.pyplot as plt

    import rainflow

    # In[]

    Adjusted_Strains = Combined['Temperature_Adjusted_strains'].as_matrix()

    # Pre-processing to get the stress variations
    # Converting the strains to stresses and fixing units, so that the values
    # are in [MPa]. Times E to get stresses, divided by 1e6 and then 1e6 = 1e12
    # to arrive at the prefix Mega, since the strains originally were in micro.
    Adjusted_Stresses = Adjusted_Strains[:]*210e9/(1e12)

    # Counting using the rainflow script.
    rf_count_adjusted = rainflow.count_cycles(Adjusted_Stresses,ndigits)

    # Making list array.
    rf_count_adjusted=np.asarray(rf_count_adjusted)

    # Applying the SAF
    rf_count_adjusted[:,0]=rf_count_adjusted[:,0]*SAF_sigma
    rf_count_adjusted[:,1]=rf_count_adjusted[:,1]*SAF_n

    # Removing all values below the threshold limit
    rf_count_adjusted = rf_count_adjusted[rf_count_adjusted[:,0] > threshold]

    rf_count_unadjusted=[]
    if RC_unadjusted==1:
        Unadjusted_strains = Combined['strain'].as_matrix()
        Unadjusted_Stresses = Unadjusted_strains[:]*210e9/(1e12)
        rf_count_unadjusted = rainflow.count_cycles(Unadjusted_Stresses,ndigits)
        rf_count_unadjusted = np.asarray(rf_count_unadjusted )
        rf_count_unadjusted[:,0] = rf_count_unadjusted[:,0]*SAF_sigma
        rf_count_unadjusted[:,1] = rf_count_unadjusted[:,1]*SAF_n
        rf_count_unadjusted  = rf_count_unadjusted [rf_count_unadjusted [:,0] > threshold]

    # In[Saving as CSV]:

    save_to_file=outputfolder+outputfilename

    np.savetxt(save_to_file,
               rf_count_adjusted,
               delimiter=",")

    if RC_unadjusted==1:
        np.savetxt(outputfolder+'stress_and_cycles_unadjusted.csv',
                   rf_count_unadjusted,
                   delimiter=",")

    # In[Plotting]:

    if RC_adjusted==1 and RC_unadjusted==1:
        temp_max=np.array([max(rf_count_adjusted[:,0]),max(rf_count_unadjusted[:,0])])
        temp_min=np.array([min(rf_count_adjusted[:,0]),min(rf_count_unadjusted[:,0])])
        max_=math.ceil(temp_max.max())
        min_=int(temp_min.min())
        plt.figure()
```

```
        plt.hist(rf_count_adjusted[:,0],weights=rf_count_adjusted[:,1],bins=np.arange(0,max_+bin_size,bin_size),label='Temperature adjusted')
        plt.hist(rf_count_unadjusted[:,0],weights=rf_count_unadjusted[:,1],bins=np.arange(0,max_+bin_size,bin_size),alpha=0.5,label='Temperature unadjusted')
        plt.xlabel('Stress range in MPa')
        plt.ylabel('Number of cycles')
        plt.title('Stress histogram comparison')
        plt.grid()
        plt.legend()
    elif RC_adjusted==1 and RC_unadjusted==0:
        temp_max=np.array([max(rf_count_adjusted[:,0])])
        temp_min=np.array([min(rf_count_adjusted[:,0])])
        max_=math.ceil(temp_max.max())
        min_=int(temp_min.min())
        plt.figure()
        plt.hist(rf_count_adjusted[:,0],weights=rf_count_adjusted[:,1],bins=np.arange(0,max_+bin_size,bin_size),label='Temperature adjusted')
        plt.xlabel('Stress range in MPa')
        plt.ylabel('Number of cycles')
        plt.title('Stress histogram')
        plt.grid()
        plt.legend()
    elif RC_adjusted==0 and RC_unadjusted==1:
        temp_max=np.array([max(rf_count_unadjusted[:,0])])
        temp_min=np.array([min(rf_count_unadjusted[:,0])])
        max_=math.ceil(temp_max.max())
        min_=int(temp_min.min())
        plt.figure()
        plt.hist(rf_count_adjusted[:,0],weights=rf_count_adjusted[:,1],bins=np.arange(0,max_+bin_size,bin_size),label='Temperature adjusted')
        plt.xlabel('Stress range in MPa')
        plt.ylabel('Number of cycles')
        plt.title('Stress histogram')
        plt.grid()
        plt.legend()

    # In[Returning info]
    return(rf_count_adjusted,rf_count_unadjusted)
```

I.4 – Equivalent stress calculation script

The script that computes the equivalent stress, based on the double slope equation.

```
# In[Info]:
# sigma_i and n_3 represents values in m=3 region
# sigma_j and n_5 represents values in m=5 region

# Calculates the equivalent stress range for values distributed in both the
# m=3 and m=5 region. It requires the selection of which slope the equivalent
# stress range should be calculated for, i.e. in m=3 or m=5 region.

def eq_stress(rf_count_adjusted,rf_count_unadjusted,outputfolder,sigma_C,sigma_D,sigma_L,gamma_Ff,gamma_Mf,m,calc_adjusted,calc_unadjusted):
    # In[Importing libraries]:
    import numpy as np
    import sys

    # In[reading data]:

    if calc_adjusted==1 and calc_unadjusted==1:
        sys.exit("Cant set it to calculate for both adjusted and unadjusted")
    elif calc_adjusted==0 and calc_unadjusted==0:
        sys.exit("No calculation parameter set in main.py")
    elif calc_adjusted==1:
        stress_and_cycles=rf_count_adjusted
        print('Calculating for temperature adjusted stresses')
    elif calc_unadjusted==1:
        stress_and_cycles=rf_count_unadjusted
        print('Calculating for temperature unadjusted stresses')

    # In[Pre processing]:
    # Applying the SAF for all of the stresses and stress ranges.  Then sorting
    # the different stress ranges into their respective slope regions, m=3,
    # m=5 or m=infinity (removing those with m=infintiy).

    # Removing all values below the sigma_L limit:
    stress_and_cycles = stress_and_cycles[stress_and_cycles[:,0] > sigma_L]

    # If no values remain after removing those below cut off limit, then exit.
    if len(stress_and_cycles)==0:
        sys.exit("No values left, all are below Delta_sigma_L")

    # Collecting those in the m=3 region
    s_and_c_m3=stress_and_cycles[stress_and_cycles[:,0] > sigma_D]
    sigma_3=s_and_c_m3[:,0]
    n_3=s_and_c_m3[:,1]

    # Collecting those in the m=5 region
    s_and_c_m5=stress_and_cycles[stress_and_cycles[:,0]<sigma_D]
    sigma_5=s_and_c_m5[:,0]
    n_5=s_and_c_m5[:,1]
    # In[Calculating equivalent stress]:
    # Calculation differs for the different slopes.

    if m==3:
        sigma_eq=( ( (sum(n_3*sigma_3**3)+sum(n_5*sigma_5**5/sigma_D**2)) /
              (sum(n_3)+sum(n_5)) ) ** (1/3) )
    elif m==5:
        sigma_eq=( ( (sum(n_3*(sigma_3**3)*sigma_D**2)+sum(n_5*sigma_5**5)) /
              (sum(n_3)+sum(n_5)) ) ** (1/5) )
    else:
        print("Incorrect slope selection, chose properly -.-")
        exit

    n_eq=sum(n_3)+sum(n_5)

    print("The equivalent stress, for slope of m =",m,", is",sigma_eq, "MPa",
```

```
64.           "and the number of cycles is", n_eq)
65.
66.
67.     # In[Saving the equivalent stress ranges and number of cycles]:
68.
69.     save_to_file=outputfolder+'equivalent_stress.csv'
70.     np.savetxt(save_to_file,
71.                [[sigma_eq,n_eq]],
72.                delimiter=",")
73.
74.     # In[Returning]
75.     return(sigma_eq,n_eq)
```

I.5 – Damage prediction script

The script that calculates the damage predicted, based on the measurements, and extrapolates it over the chosen period.

```
# In[Info]:
# Calculates the damage accumulated on the detail, for the time considered.

def dmg_acc(sigma_D,gamma_Ff,gamma_Mf,m,sigma_eq,n_eq,start_time,end_time,time_remaining):

    # In[Importing]:
    import datetime as dt

    # In[Damage accumulation calculation]:

    N_life=5*10**6 * ((sigma_D*gamma_Mf)/(sigma_eq*gamma_Ff))**m

    D_measured=n_eq/N_life

    print("The damage accumulated for the considered sample of strains is",D_measured)

    # In[Year equivalent]:

    start_time=str(start_time)+'.000000'
    end_time=str(end_time)

    start_dt=dt.datetime.strptime(start_time[0:26], "%Y-%m-%d %H:%M:%S.%f")
    end_dt=dt.datetime.strptime(end_time[0:26], "%Y-%m-%d %H:%M:%S.%f")

    seconds_measured=dt.timedelta.total_seconds(end_dt-start_dt)
    seconds_year=365*24*60*60
    year_factor=seconds_year/seconds_measured

    D_year=D_measured*year_factor

    D_destruction=D_year*time_remaining

    print("The year equivalent damage, based on the measured data, is",D_year)

    print("The damage caused until the destruction, based on the measured data, is",D_destruction)


    # In[]:

    return(D_measured)
```