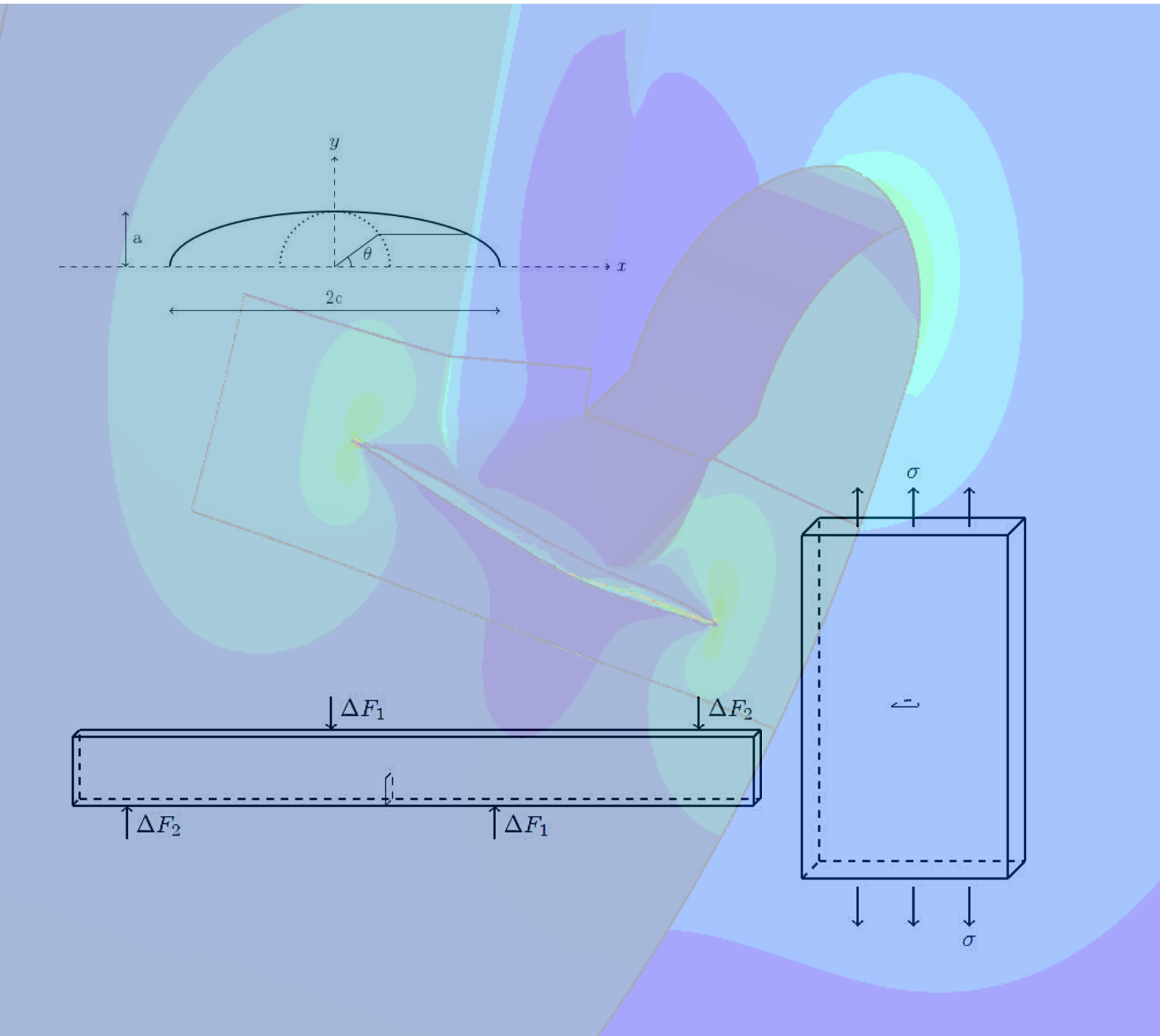
Fracture Mechanics

Application on Orthotropic Steel Decks

D. Malschaert



Fracture Mechanics

Application on Orthotropic Steel Decks

by

D. Malschaert

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Wednesday May 20, 2020 at 2:00 PM.

Student number:	4574796	
Project duration:	May 6, 2019 – May 20, 2020	
Thesis committee:	Prof. dr. M. Veljkovic,	TU Delft, Chair
	W. Wu MSc,	TU Delft, Daily supervisor
	Dr. F. van der Meer,	TU Delft
	Dr. H. Xin,	TU Delft
	B. van Aken MSc,	Iv-Infra, Daily supervisor

TU Delft

Iv-Infra

Preface

This thesis concludes my studies for a master's degree for Civil Engineering at Delft University of Technology. With great pleasure, I look back at all the years I have studied at this university. After a heavy but exciting period, I am both glad and sad to move on to the next stage. With the finalising of this chapter in my life comes the appreciation of all the people that have played a key role in it. There are a number of people that have provided me with both motivation and guidance throughout the process of writing the thesis.

Therefore, to my graduation committee, I would like to express my gratitude. Firstly, to professor Veljkovic, chair of the committee, for helping me find a suitable research topic and for keeping a keen eye on the research progress. To my daily supervisors Weijian Wu and Bart van Aken, on whom I could always rely for both theoretical as practical support. Finally, to Frans van der Meer and Haohui Xin for their valuable input in my thesis work.

Many thanks to Iv-infra for allowing me to write a thesis in the presence of such a great engineering team. There is no doubt that the slogan "Passion for technology" suits the company well.

Last but certainly not least, to my family for their unconditional love and support, without whom I would not be able to start and finish my studies. My girlfriend, your love and care has kept me going even in times where motivation was low. Being a stranger to engineering, your endurance to long speeches about fracture mechanics and bridges has amazed me.

D. Malschaert
Delft, May 2020

Abstract

In the course of this thesis, a research has been conducted to the application of fracture mechanics on an othotropic steel deck (OSD). Obtaining accurate predictions of crack path and fatigue life assessment were the main objectives for this thesis. As 3D crack propagation modelling in orthotropic steel decks is an arduous task, simpler studies were examined first in order to obtain confidence for the task at hand. These studies are comprised of a through thickness crack in an asymmetrical four point bending specimen and a semi-elliptical crack in a centre cracked plate. Furthermore, studies on computational fracture mechanics and the integration of Python scripting language with Abaqus were conducted for numerical analysis of crack propagation in complex structures. The extended finite element method was used for evaluation of contour integrals of a crack with a constant mesh throughout the propagation analysis. Post-treatment of the results was applied by smoothing stress intensity factors and the shape of the crack front. Moreover, inaccurate contour integral results were excluded from analysis through extrapolation. A Python code was developed for updating the crack front for a given number of loading cycles integrated in Paris' law.

The investigated crack was located at the rib-to-crossbeam joint of an OSD. Here, crack propagation occurs along the lower weld toe and in the longitudinal direction of the rib. Validation of results was obtained through experimental studies conducted at the TU Delft laboratory. The weld geometry was incorporated in the model from measurements in order to identify local stresses. Finally, results of strain gauges were used to validate the model. Measurements of the experimental crack growth were obtained at the surface of the crack only. Consequently, assumptions were made of both the depth and shape of the crack in the thickness direction of the rib. A semi-elliptical crack shape was selected with an initial crack depth of 1 mm. Initially, Paris law parameters C and m were selected as $3.98 \cdot 10^{-13}$ and 2.88 respectively.

The numerical results of crack path complied with those of experimental results, with deviations of no more than 1 mm. Both data sets show that the propagation rate remained somewhat constant. On the other hand, fatigue life assessment results showed stronger deviation. After calibration of Paris law parameter C, a value of $2.00 \cdot 10^{-13}$ is recommended for this specific application. A parametric study was performed to investigate the effect of the initial crack depth. Three different cases were examined for which the initial crack depths were selected as 1, 2 and 3 mm. Varying the initial crack depth did not lead to a significant change in crack path at the surface of the rib, while the depth in the thickness direction was affected. Regarding the fatigue life assessment, the results with respect to an initial crack depth of 1 mm, complied best with experimental results.

Given the strong correlation between experimental and numerical results, the methodology followed in this thesis has proven valuable regarding fatigue crack modelling. As the methodology is based on fracture mechanics, an initial crack is required prior to evaluation. Consequently, the application in engineering practices lies in re-evaluation of structures. Moreover, as the parameters in crack propagation relations have a significant influence on the fatigue life assessment, these should be carefully chosen. Therefore, more insights must be obtained regarding the selection of such parameters.

D. Malschaert
Delft, May 2020

Contents

1 Introduction

1.1. Problem statement

Steel structures are mostly exposed to cyclic loading and their fatigue design plays an important role in guaranteeing the structural safety. Since the macroscopic cracks often originate from microscopic cracks or defects and are influenced by residual stresses due to fabrication processes, it is hard to predict when/where a crack will arise or propagate. During fabrication it is unavoidable to have flaws in a structure. Even within fabrication tolerances, these flaws can have a significant impact on the service lifetime. An engineering discipline that deals with these flaws or cracks is fracture mechanics. Fracture mechanics describes the events around and the propagation of cracks.

During the nineteen-sixties and seventies, several orthotropic steel bridge decks were built in the Netherlands. These however, have shown to be susceptible to fatigue loading. After several decades, severe cracks were found leading to premature repairing of bridges [27]. To be able to estimate crack propagation behaviour, a fracture mechanics based model can be used. With this information, inspection and repair periods can be determined more accurately, ultimately reducing the maintenance costs.

The goal of this thesis is to provide information in crack propagation behaviour based on a fracture mechanics approach using finite element modelling. Finite element modelling is necessary to provide solutions for complex structures as is usually the case in engineering practice. Furthermore, this method is applied on a crack in an orthotropic steel bridge deck. The crack which is examined is located at the lower weld toe of the rib-to-crossbeam joint.

1.2. Research questions

This research is conducted to answer the following main research question:

How can crack growth, in orthotropic steel decks, be modelled using fracture mechanics concepts?

In order to answer this question, answers are needed to the following questions:

- How are stress intensity factors obtained through finite element modelling?
- How can crack propagation directions be determined?
- How can finite element modelling be used to make a crack propagate?
- How can fatigue life be estimated using crack propagation relations?
- How do design standards prescribe how to deal with pre-existing flaws?

1.3. Methodology

This is a literature based research where experimental results, commercial software and research papers have been used to validate results obtained. The literature review provides the necessary background information to conduct further research within this thesis. From this, a methodology of modelling crack propagation in structures is composed. Using this approach, computational fracture mechanics and Python scripting proved to be fundamental.

A method to determine stress intensity factors through finite element modelling is discussed. To obtain more experience with computational fracture mechanics, a study on a through thickness crack is performed. Here, a propagating crack is modelled in an asymmetrical four point bending specimen. A pre-existing study performed by Huang et al. [25] serves as a reference and validation study. A second study is applied to a more realistic 3D crack. Here, a semi-elliptical surface crack is examined in a centre cracked plate. Finally, a more complex study is performed as the same method is applied for a crack propagating in an orthotropic steel deck. Crack propagation trajectory and fatigue life assessment are main results obtained from the analyses.

1.4. Structure of report

Firstly a literature review was performed which serves as a foundation of this thesis. Alongside this literature review, modelling in Abaqus and scripting with Abaqus are studied and are presented in appendices B and C respectively. These have proven to be crucial skills for this thesis regarding fracture mechanics application on a computational model as is discussed in chapters 3, 4 and 5. Finally, conclusions and recommendations are given in the final chapter of this thesis, chapter 6. An overview of the structure of this report is given in figure 1.1.

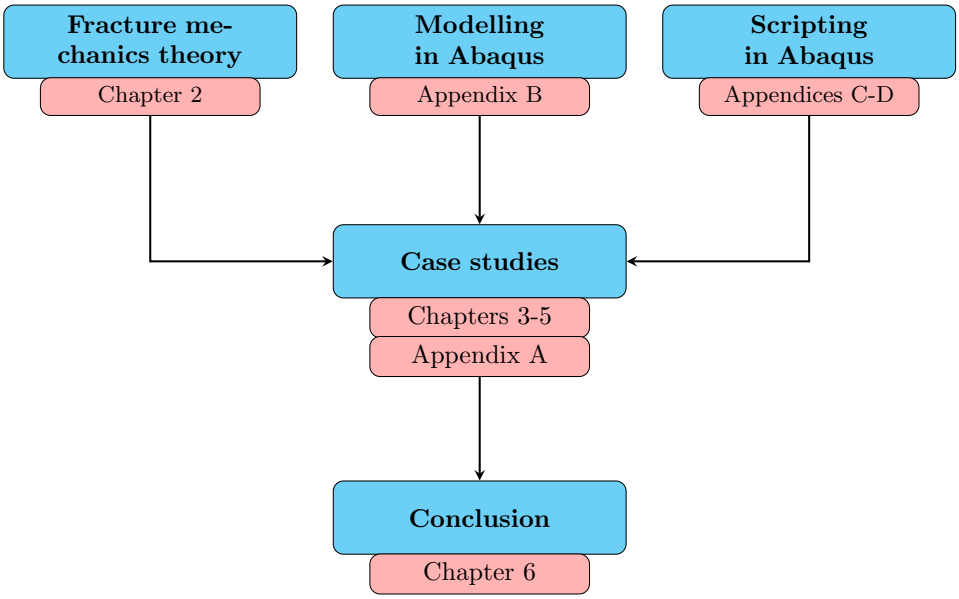


Figure 1.1: Structure of thesis report

2

Literature Review

This chapter presents the literature review needed for this thesis work. Firstly, the stages of a crack and crack growth are discussed. Secondly, the concept on linear elastic fracture mechanics and the stress intensity factor are introduced. Finally fatigue in steel orthotropic bridge decks and the use of design standards are treated.

2.1. Stages of a crack

Fatigue cracks are influenced by many aspects which do not have a constant effect on the crack throughout the service life. For example, the surface roughness of a material has great influence on a crack when it is still relatively small as opposed to relatively large cracks. Therefore, two periods may be considered in fatigue life, the initiation period and the crack growth period. In the initiation period, microcracks are formed and microcrack growth occurs. After the initiation period, macrocracks are formed and macrocrack growth takes place. In this crack growth period, macrocracks may grow until an unstable situation where final failure of the structure occurs. In the figure given below, both periods are presented with the corresponding factors which are treated in this literature overview [43].

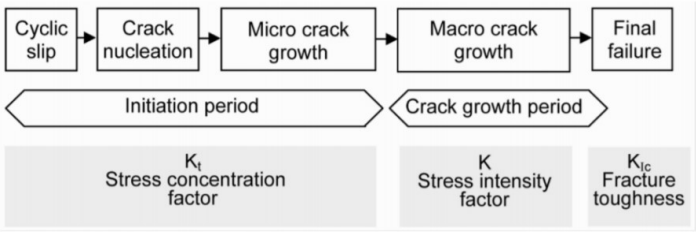


Figure 2.1: Phases in fatigue life with relevant factors [43]

2.1.1. Crack initiation

Fatigue crack initiation is a result of cyclic slip induced by cyclic shear stresses at microscopic level. As a consequence, microcracks are formed. The exact location where slip occurs depends on e.g. the grain size and shape, the elastic anisotropy and crystallographic orientation. The surface of the material has proven to have great influence on the crack initiation since this is where cracks generally start. The surface roughness and the presence of corrosion pits of the material enhance the initiation of surface cracks. Therefore, surface treatments may be applied to improve both the fatigue as corrosion resistance [26, 43].

Stress concentration factor
Microcracks are most often formed at locations where there is an imperfection or other disturbance in geometry. This disturbance in geometry leads to peak stresses at certain locations. To describe this phenomena mathematically, a stress concentration factor was introduced. The stress concentration factor K_t, as defined in [43], is the ratio between the peak stress at the root of the notch and the nominal stress which would be present if a stress concentration did not occur. This stress concentration factor has a prime role in designing against fatigue since these peak stresses often lead to cracks [43].

$$K_t = \frac{\sigma_{peak}}{\sigma_{nominal}} \tag{2.1}$$

For many structural details, the crack initiation phase plays a significant role in the service life. However, for welded details or details containing defects (i.e. due to fabrication tolerances), the crack growth (or crack propagation) phase dominates the fatigue life. Moreover, the resistance of the crack propagation stage for high strength steels may be lower than mild steel. Therefore, for welded structures, choosing a steel with a higher yield strength may not be beneficial for fatigue analysis [16, 37, 54].

2.1.2. Crack growth

In the crack growth period, material surface conditions are no longer of great importance. The formation of macrocracks and macrocrack growth is rather considered as a bulk material phenomenon. The growth direction is mainly perpendicular to the loading direction [43]. A widely accepted relation for macroscopic crack growth is the Paris law. The Paris law describes the crack growth rate as a function of the stress intensity factor range. The stress intensity factor is addressed in section 2.2.2. The equation is given by [39]:

$$\frac{da}{dn} = C(\Delta K)^m \tag{2.2}$$

Which can be rewritten as:

$$dn = \frac{da}{C(\Delta K)^m} \tag{2.3}$$

Herewith, we can calculate the number of cycles needed for a crack to increase with da. It should be noted that the stress intensity factor range ΔK changes each time dn is calculated due to its dependence on the crack size a. Therefore, an analytical expression of the total number of cycles is not intuitive. This can be solved by an numerical approach [26]. The factors C and m are two material constants. BS7910 gives recommended values as well as mean and mean plus 2 times the standard deviation values for C and m [13]. In the figure given below, the typical sigmoidal curve is presented for $\log(\frac{da}{dN})$ versus $\log(\Delta K)$. It should be noted that the Paris law defined above only holds for regime II (the Paris region) in the figure given below. At regime I the crack growth rate increases rapidly with increasing stress intensity factor (SIF) while at regime III the SIF grows until the critical value K_c. Moreover, there is a certain threshold stress intensity factor ΔK_{th} below which no (significant) crack growth occurs [26].

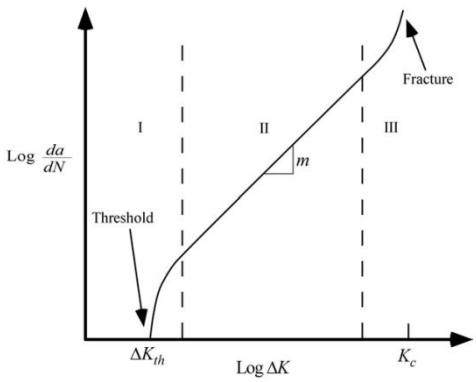


Figure 2.2: Fatigue crack growth curve [7]

The improved Forman relation [21] describes both regime II and III:

$$\frac{da}{dn} = \frac{C(\Delta K)^m}{(1-R)K_c - \Delta K} = \frac{C(\Delta K)^m}{(1-R)(K_c - K_{max})} \tag{2.4}$$

Here, the stress ratio R is included as well as the critical stress intensity factor K_c. The critical stress intensity factor K_c, described in the next section. The denominator approaches zero as K_{max} approaches K_c, leading to a vertical asymptote. Many other relations have been proposed, also including the entire curve, i.e. Erdogan and Ratwani [18] and ESACRACK [19].

2.1.3. Fracture toughness and critical crack length

The fracture toughness (or plane strain fracture toughness) K_{Ic} is a material property that describes the sensitivity for cracks under static loading. Fracture toughness is an important parameter to determine the residual strength of a structure. Furthermore, it can calculate the critical crack size or the critical stress in the appearance of a crack. The fracture toughness may be obtained with a so called Charpy test [26, 37]. For example, it could be said that fracture will take place if the stress intensity factor exceeds the fracture toughness [30, 48].

$$K_I \geq K_{Ic} \tag{2.5}$$

The stress intensity factor is discussed in section 2.2.2. The stress intensity factor at which fracture will occur is called the critical stress intensity factor. As explained below, this critical stress intensity factor is not a constant but depends on multiple factors, including temperature and specimen thickness [26]. We can rewrite the equation for the stress intensity factor (see equation 2.15) to obtain the critical crack length with the given fracture toughness and critical stress.

$$a_c = \frac{1}{\pi}\left(\frac{K_{Ic}}{Y\sigma_c}\right)^2 \tag{2.6}$$

where:

a_c = Critical crack length;
K_{Ic} = Fracture toughness;
Y = Geometrical factor;
σ_c = Critical stress.

Since, in this equation, the fracture toughness may be assumed a material property and the stress applied is assumed to be known as is the geometric factor Y, the critical crack length can easily be calculated.

Specimen thickness dependency on K_c

For relatively small thicknesses, the plane strain fracture toughness K_{Ic} (as described above) is a conservative value. Instead, the critical stress intensity factor might be higher. In the figure given below, the critical stress intensity factor is presented as a function of specimen thickness. K_c converges to the plane strain fracture toughness as the thickness increases. Therefore, in many applications, K_{Ic} is used for calculations. The increasing trend for fracture toughness for thin elements is due to the transition from plane strain to plane stress [26]. What should be noted here is that failure is more brittle in thick elements than in thin ones. Therefore, EN1993-1-10 gives upper values for element thicknesses [34].

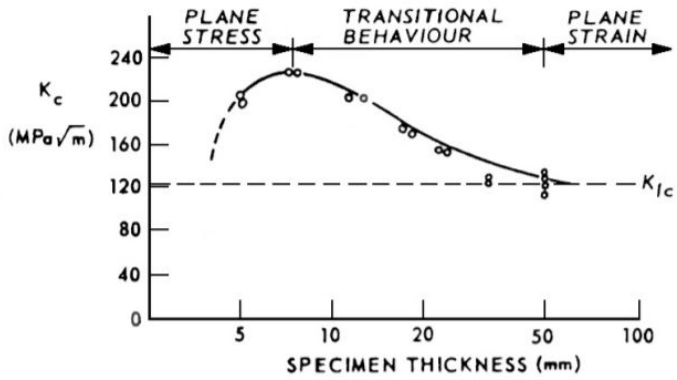


Figure 2.3: Critical stress intensity factor versus thickness [26]

Temperature and yield strength dependency on K_c
Together with specimen thickness, temperature and yield strength have an influence on fracture toughness. At relatively low service temperatures, steels fail more brittle. Furthermore, a higher yield strength leads to a lower fracture toughness. This is due to the smaller plastic zone in higher grade steels [26]. EN1993-1-10 provides a table from which the maximum thickness may be derived for reference temperatures, reference stress level, steel grades and steel qualities (which is related to the fracture toughness) [34].

2.2. Linear Elastic Fracture Mechanics

Fracture mechanics provides answers to several questions regarding fracture and fatigue. Some important questions can be [26]:

- What is the residual strength of a structure as a function of crack size;
- What crack size can be tolerated under service loading;
- How long does it take for a crack to grow from a certain initial size, for example the minimum detectable crack size, to the maximum permissible crack size;
- What is the service life of a structure when a crack-like flaw (e.g. a manufacturing defect) with a certain size is assumed to exist.

In linear elastic fracture mechanics (LEFM) we assume all the material to behave linear elastic and that the material is isotropic [6, 7, 30]. Locally, some plasticity may occur at the crack tip. However, in high cycle fatigue this region remains small and LEFM may still be used [7, 26, 30].

2.2.1. Elastic stress field approach

The elastic stress field approach, or stress intensity approach, is one of the most widely used methods. This method provides information on the stress field surrounding the crack tip as well as information about the crack propagation and the elastic deformation around the crack [6, 26, 42]. The elastic deformation will not be dealt within this thesis since it gives no further insight on the research questions. For information about linear elastic deformations in material around a crack, reference is made to [26].

Three different modes of crack surface displacements are described. Herewith, all possible stress systems can be examined. Within these modes, mode *I* is most common in engineering practice and important regarding crack growth. All three modes are presented in figure 2.4. Mode I, II and III represent the opening, sliding and tearing mode respectively [7, 26, 42, 43].

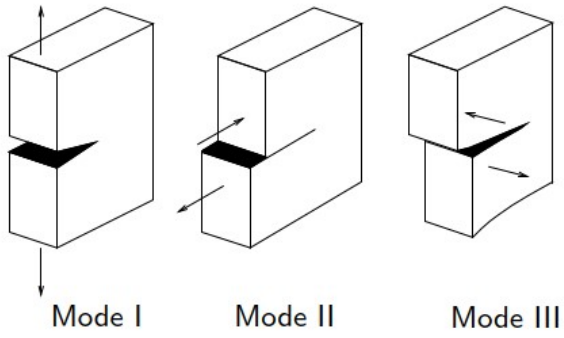


Figure 2.4: Three modes of crack surface displacements [44]

Expressions to describe the stress state close to the crack tip for a blunted tip (a tip with a finite radius) for all three modes were proposed by Paris and Creager [15]. Most research papers and study books state only the equations to mode *I* since this mode is most practical for engineering purposes. However, for the sake of completion, the equations for all three modes and the coordinate system are stated below [15, 43]. Furthermore, the local coordinate system at the crack tip is presented in figure 2.5.

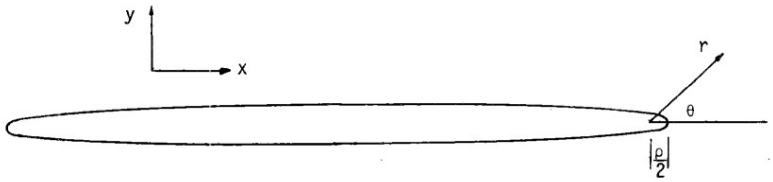


Figure 2.5: Coordinate system blunt crack [15]

Mode I:

$$\sigma_{xx} = \frac{K_I}{\sqrt{2\pi r}}\cos(\theta/2)[1-\sin(\theta/2)\sin(3\theta/2)] - \frac{K_I}{\sqrt{2\pi r}}\frac{\rho}{2r}\cos(3\theta/2) \tag{2.7}$$

$$\sigma_{yy} = \frac{K_I}{\sqrt{2\pi r}}\cos(\theta/2)[1+\sin(\theta/2)\sin(3\theta/2)] + \frac{K_I}{\sqrt{2\pi r}}\frac{\rho}{2r}\cos(3\theta/2) \tag{2.8}$$

$$\tau_{xy} = \frac{K_1}{\sqrt{2\pi r}}\sin(\theta/2)\cos(\theta/2)\cos(3\theta/2) - \frac{K_I}{\sqrt{2\pi r}}\frac{\rho}{2r}\sin(3\theta/2) \tag{2.9}$$

Mode II:

$$\sigma_{xx} = \frac{K_{II}}{\sqrt{2\pi r}}\sin(\theta/2)[2+\cos(\theta/2)\cos(3\theta/2)] + \frac{K_{II}}{\sqrt{2\pi r}}\frac{\rho}{2r}\sin(3\theta/2) \tag{2.10}$$

$$\sigma_{yy} = \frac{K_{II}}{\sqrt{2\pi r}}\sin(\theta/2)\cos(\theta/2)\cos(3\theta/2) - \frac{K_{II}}{\sqrt{2\pi r}}\frac{\rho}{2r}\sin(3\theta/2) \tag{2.11}$$

$$\tau_{xy} = \frac{K_{II}}{\sqrt{2\pi r}}\cos(\theta/2)[1-\sin(\theta/2)\sin(3\theta/2)] - \frac{K_{II}}{\sqrt{2\pi r}}\frac{\rho}{2r}\cos(3\theta/2) \tag{2.12}$$

Mode III:

$$\tau_{xz} = \frac{K_{III}}{\sqrt{2\pi r}}\sin(\theta/2) \tag{2.13}$$

$$\tau_{yz} = \frac{K_{III}}{\sqrt{2\pi r}}\cos(\theta/2) \tag{2.14}$$

In these equations, r and θ are polar coordinates as can be seen in figure 2.5. In case of a sharp tip, i.e. $\rho = 0$, the last term of mode I and II drops and the coordinate system's origin is at the crack tip (instead of at $\rho/2$ distance away from the crack tip) [26].

2.2.2. Stress intensity factor K

The elastic stress field approach is based on an important parameter within fracture mechanics, namely the stress intensity factor K. The stress intensity factor describes the magnitude of the elastic crack tip stress field. Herewith, the crack growth and fracture behaviour of materials can be predicted, provided that the stress field around the crack tip remains predominantly elastic [26].

As mentioned, the initial crack in fracture mechanics needs to be predefined. Therefore, in the equations given above, the crack tip radius ρ is a known parameter. Polar coordinates are used in these equations and thus, r and θ are input variables for the location of interest. Therefore, the stress intensity factor K is the one variable left to describe the stresses and specimen and crack geometry in these formula. Many studies have been performed to identify stress intensity factors in different situations and handbooks are formed for relatively simple situations. However, for complex geometries/loading conditions, these are not so easily determined and finite element analysis is inevitable [13, 26].

The stress intensity factor is often written in the form of a geometric function Y multiplied with the remote stress field (or nominal stress) and the square root of π multiplied with the crack length [7, 26, 48]. In BS7910, the general form is stated as [13]:

$$K_I = Y\sigma\sqrt{\pi a} \tag{2.15}$$

where:

K_I = Stress intensity factor mode I;
Y = Geometric function;
σ = Remote stress;
a = Crack length.

Superposition of stress intensity factors

Often, in engineering practice, structures are designed for multiple loads simultaneously. Therefore, stress intensity factors for such cases are needed. For equal modes (see figure 2.4), the superposition principle holds for LEFM as illustrated in figure 2.6. Therefore, if the stress intensity factors are known for each load separately, the total stress intensity factor can be determined for each mode [26, 43].

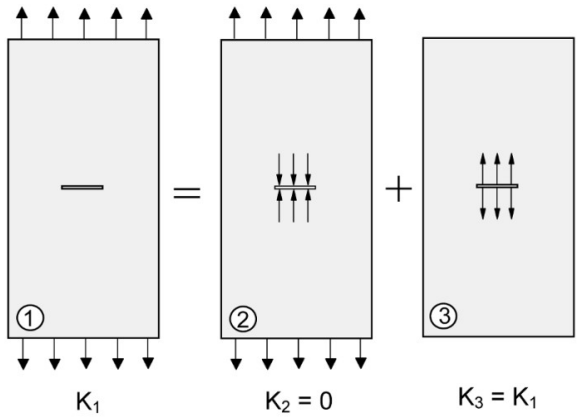


Figure 2.6: Application of the superposition principle for SIF [43]

2.2.3. J-Integral

The J-integral is a path independent line integral that takes a value of the decrease in potential energy per increment of crack area extension in both linear and nonlinear material. This line integral surrounds the crack tip and has a direction of counter clockwise as shown in the figure below. Due to its path independence, the user may evaluate the J-integral at different contours and find the same value. Furthermore, a path may be defined depending on the convenience for calculation. The J-integral is defined as [7, 26, 30, 48]:

$$J = -\frac{dU_p}{dA} \tag{2.16}$$

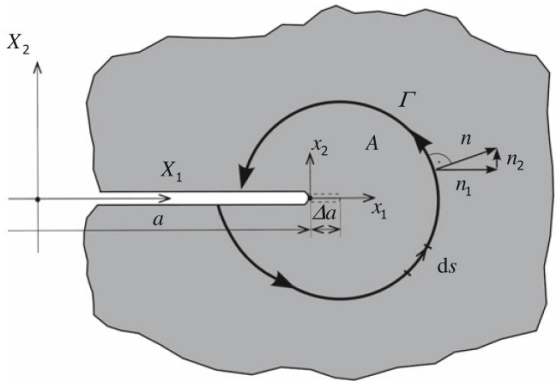


Figure 2.7: J as a line integral [30]

Here, U_p is the potential energy of the system. It should be noted that, in the denominator, the crack area close to the crack tip is denoted and not the crack length or depth. For a through thickness centre crack with unit thickness this results in $dA = 2da$ [7]. When considering linear elastic material, the J-integral equals the energy release rate G which is proportional to the stress intensity factor squared. Therefore, when the J-integral is known, the stress intensity factor can be calculated. Below, geometry-independent relations for plane stress and plane strain are stated [7, 26, 48].

$$Plane\ stress: \quad J = G = \frac{K_I^2}{E} \tag{2.17}$$

$$Plane\ strain: \quad J = G = \frac{K_I^2}{E}(1-\mu^2) \tag{2.18}$$

It should be noted that these relations are valid for mode I dominant cracks. Otherwise J can be dependent on all stress intensity factors. Here, E is the Young´s modulus and μ is the Poison's ratio. The strain energy release rate is not treated in this literature review as it will not be used in this thesis. For the reader's interest, the theory about strain energy release rate based on Griffith energy balance approach and developed by Irwin can be found in many literature, i.e. [7, 26, 48].

2.3. Extended Finite Element Method

With the extended finite element method (XFEM), discontinuities (such as cracks) can be modelled mesh-independently. This is contrary to the finite element method (FEM), where alignment of the mesh with the discontinuity is needed. With XFEM, local enrichment is used around the discontinuity through partition of unity. Using conventional FEM the approximated displacement field is given by [5, 31]:

$$FEM: \quad u_i^h = \sum_{I \in \Omega^{fem}} N_i^I u_i^I \tag{2.19}$$

In XFEM, the approximated displacement field holds an additional term taking into account the enriched nodes [5, 31]. The enriched nodes are visualised in figure 2.8.

$$XFEM: \quad u_i^h = u_i^{FEM} + u_i^{enr} = \sum_{I \in \Omega^{fem}} N_i^I u_i^I + \sum_{J \in \Omega^c} N_i^J \psi a_i^J \tag{2.20}$$

where:

- N_i are shape functions;
- u_i^I is the displacement at node i;
- ψ is the enrichment function;
- a_i^J are the enriched degree of freedoms;
- Ω^{fem} are set of all nodes;
- Ω^c are the enriched set of nodes.

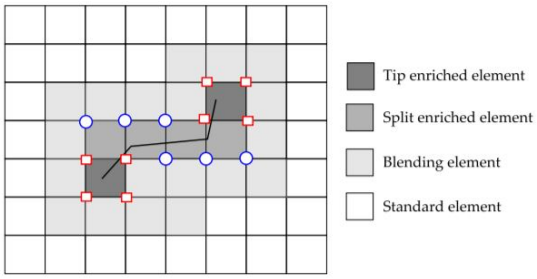


Figure 2.8: Enriched elements along crack discontinuity [33]

For the completely cracked elements, Heavyside functions (H) can be used as enrichment functions. At the crack tips however, other enrichment functions are needed as the element contains only a partial discontinuity. A $\sqrt{r}$ term is implemented to locate the singularity at the tip (see section 2.2.1). The four crack tip enrichment functions for an isotropic elastic material are [20, 33, 45]:

$$F_\alpha(r,\theta) = \left\{ \sqrt{r}\sin\left(\frac{\theta}{2}\right), \sqrt{r}\cos\left(\frac{\theta}{2}\right), \sqrt{r}\sin\left(\frac{\theta}{2}\right)\sin(\theta), \sqrt{r}\cos\left(\frac{\theta}{2}\right)\sin(\theta) \right\} \tag{2.21}$$

Here, r and θ are polar coordinates with the coordinate system at the crack tip. Having enrichment functions for both the completely cracked as well as the partially cracked elements, the approximated displacement field reads [22, 33, 45]:

$$u_i^h = \sum_{I \in \Omega^{fem}} N_i^I u_i^I \quad + \quad \sum_{J \in \Omega^c} N_i^J H a_i^J \quad + \quad \sum_{K \in \Omega^f} N_i^K \sum_{\alpha=1}^{4} F_\alpha b_i^{K\alpha} \tag{2.22}$$

Where, Ω^{fem} is the set of all nodes, Ω^c is the set of enriched nodes corresponding to the completely cut elements and Ω^f is the set of enriched nodes corresponding to the crack tip.

2.4. Fatigue in steel orthotropic decks

Numerous steel orthotropic decks have been constructed in the Netherlands since the 1960's [27, 32]. An orthotropic deck is stiffened in two orthogonal directions, namely the longitudinal and the transverse direction. This accommodates the transportation of local loads to the main girders. A typical longitudinal stiffener is a trough (or rib) and a transverse stiffener is a crossbeam [24]. A benefit of such an orthotropic deck is the relatively high stiffness with a relatively low weight. Therefore, such designs are usually found when self weight is important relative to the variable loads [24, 27]. Typical bridges where self weight of the deck is important are long span bridges and movable bridges. However, this design also has its disadvantages such as high fabrication costs and problems with fatigue. This design has resulted in various cracks needing repair in several bridges in the Netherlands [27, 32, 51, 52]. A typical section of a steel orthotropic deck is illustrated in figure 2.9.

The goal for this section is to determine where the critical locations are for steel orthotropic decks with respect to fatigue.

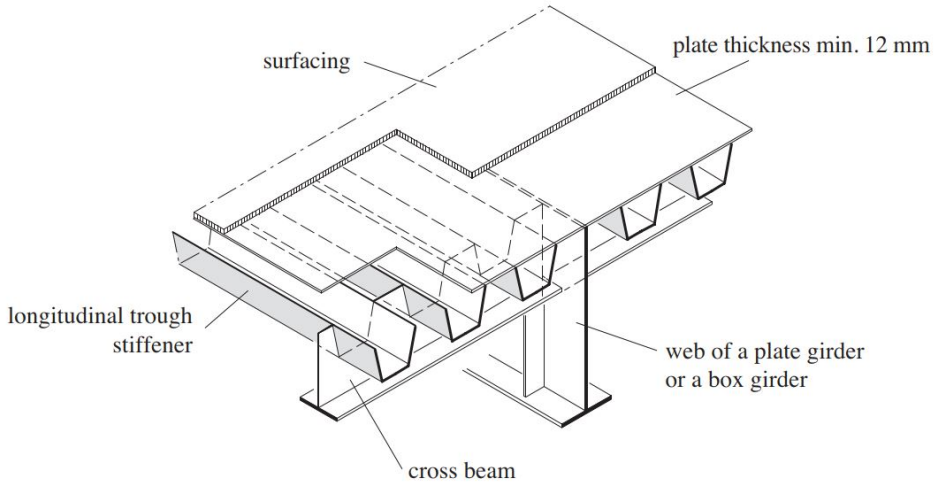


Figure 2.9: Section of steel orthotropic deck [24]

2.4.1. Locations prone to fatigue

The large amount of welds and geometric discontinuities make an orthotropic bridge deck sensitive for fatigue. De Jong [27] has investigated cracks in steel orthotropic bridge decks and divided these cracks into four distinct categories.

1. Cracks in the deck plate;
2. Cracks in the longitudinal weld between deck plate and rib web;
3. Cracks in rib splice joint;
4. Cracks in the connection between rib profile and crossbeam.

According to [27] and [32] the longitudinal crack in the deck plate, originating from the welded connection between rib and deck plate, is most severe. This is the case as this crack may cause a threat to traffic safety and thus human lives. Since this crack has shown to be most severe, this specific crack shall be investigated in more detail. These cracks can originate from the root of the weld, located on the inner side of the rib, making crack detection an arduous task. Therefore, visual inspection will only reveal these cracks when they have propagated through the deck plate [27, 29, 32]. This kind of a crack is illustrated in figure 2.10. According to Wu et al. [50], the crack initiates in the deck plate at the location of the crossbeam earlier than at the location between the crossbeams. High stress concentration occurs at the location of the weld when a concentrated load is applied between the webs of the rib at the deck plate at the location of the crossbeam. This occurs because the welded connection between the crossbeam, rib and deck plate creates a clamped support leading to a high bending stresses [27, 29, 35].

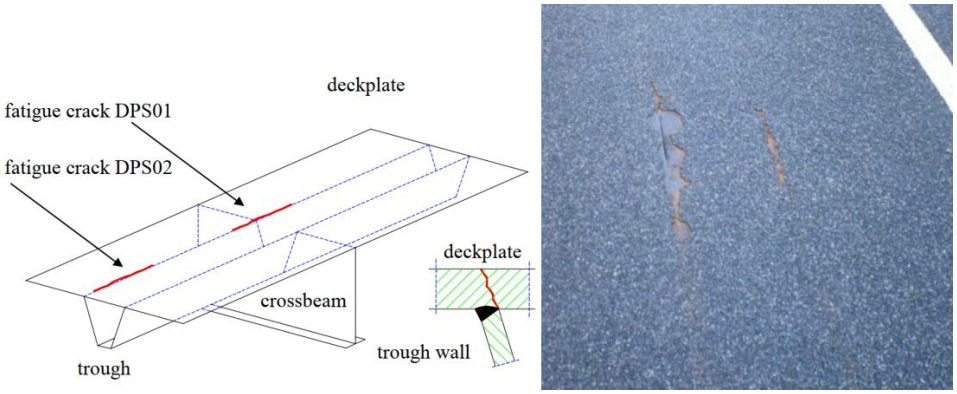


Figure 2.10: Cracks in deck plate originating from root of weld [27]

Such a crack has a semi-elliptical surface and grows vertically through the thickness of the deck plate. Once it has reached the top of the deck plate, the growth occurs in the longitudinal direction of the deck. In case of such a crack taking place in between crossbeams, vertical and longitudinal growth takes place simultaneously [27, 32]. De Jong [27] states that the length of the crack is approximately 4 times larger than the depth and a crack length starting of approximately 50 cm threats appear to traffic safety.

To improve the general fatigue behaviour of steel orthotropic decks the following recommendations were found [24, 27, 32, 37]:

- The deck plate should be thick enough to prevent cracks propagating through it;
- Continuous longitudinal stiffeners that pass through the transverse stiffeners should be applied;
- Only the transverse stiffener is to be welded to the longitudinal stiffener at the location of the webs of the longitudinal stiffener (see figure 2.11);
- Cope holes in the transverse stiffeners may be applied at the location of the base of the longitudinal stiffeners to reduce stress concentrations in the ribs [1] (see figure 2.11);
- Thicker surfacing helps to reduce the fatigue stresses but increases the overall self weight.

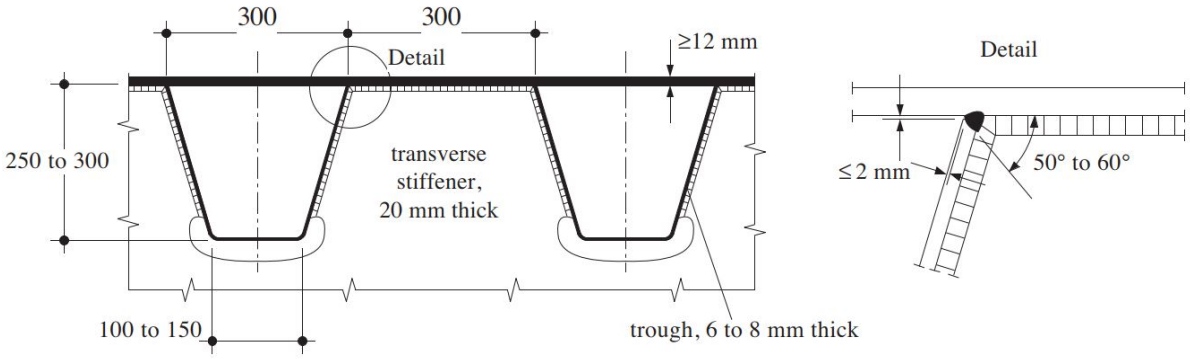


Figure 2.11: Detail connection between rib, crossbeam and deck plate [24]

[1] It should be noted that applying cope holes may have a positive effect for the ribs with respect to fatigue but may influence the crossbeam in a negative way.

2.4.2. Fatigue in welded structures

A known problem in designing against fatigue is weldments within a structure. The residual stresses due to manufacturing tend to reduce the service life of the structure. This is also implemented in the design categories of EN 1993-1-9 [36]. These residual stresses result in high tensile stresses in the vicinity of the weld that may exceed the yield stress. As a result, welded structural details that are in compression due to external loads, may well have high tensile stresses. Therefore, crack initiation and propagation are likely to occur even for global compression zones in a structure. K. Spyridoni [47] investigated this in the case of an orthotropic steel bridge deck and has shown that tensile stresses, of similar magnitude to that of the yield stress, occur at the root and toes of the welds due to residual stresses.

As described in chapter 2.4.1, cracks have occurred in steel orthotropic bridge decks originating from the root of the weld between rib and deck plate that propagate through the thickness of the deck. While these deck plates are designed to be in compression, cracks initiate and propagate due to residual stresses present through manufacturing.

2.5. Design standards

A short description is given here regarding both the Eurocode and the British Standards state how to design against fracture and fatigue.

2.5.1. British Standard 7910

The British Standard 7910 [13] (short: BS7910), is a guide to the acceptability of flaws in metallic structures. What should be noted here is that this document is not a code of practice but takes the form of a guide and recommendation.

Methods for assessment

Two methods are stated in the BS7910 for the assessment of planar flaws. The general procedure and the simplified procedure based on S-N curves (quality category procedure). While the methodologies of assessment differ, they are both based on fracture mechanics principles. The general procedure uses the cyclic stress intensity factor and crack growth relations. The simplified procedure uses S-N curves where the already performed fracture mechanics calculations are visualised graphically. The BS7910 states that the general procedure is the more accurate of the two and the quality category procedure the more simple. To decide which method can be used, a flaw must first be categorised [13].

Categorisation of flaws

The flaws in BS7910 are categorised as planar flaws, non-planar flaws or shape imperfections. Since cracks are categorised as planar flaws, the focus shall lie on these kind of flaws. The procedure for assessment of planar flaws can be through the general or the quality category procedure. Since the general procedure is the more accurate, this shall be treated here [13].

General procedure for planar flaws

The general procedure for planar flaws by BS7910 involves a definition of the input parameters and a fracture mechanics calculation. The steps to this procedure are given in figure 2.12. No specific crack growth law is stated which should be used but reference is made to various literature where the crack growth laws can be found. As an example, integration of the Paris law is shown in BS7910. The stress intensity factors can be determined as stated in Annex M of BS7910 [13]. It should be noted here that these stress intensity factor solutions only apply for relatively simple cases. As a more complex structure can be examined, solutions through a computational calculation must be found.

Step		Relevant clause references	Relevant figure(s) and table(s)
1	Determine cyclic stress range from P_m, k_{tm}, P_b, k_{tb}, Q	**6.4.1, 6.4.5, 8.2.1**	Figure 2b), Figure 3 and Table 9
2	Resolve flaw normal to maximum principal stress	**6.4.5**	Figure 4
3	Define flaw dimensions	**7.1.2**	Figure 10
4	Assess uninspectable regions	**6.3.5**	—
5	Define limit to crack growth:	—	—
a)	for unstable fracture	—	—
	Option 1	**7.3.3**	Figure 6
	Option 2	**7.3.4**	Figure 7
	Option 3	**7.3.5**	Figure 8
b)	other failure modes	Clause **10**	—
Planar flaws (general procedure)		—	—
6	Select values of A, m and ΔK_0	**8.2.3**	Table 10 to Table 12
7	Determine ΔK for cyclic stress range and flaw height and shape	**8.4.3,** Annex M	—
8	Calculate crack growth increments Δa and Δc for one stress cycle	**8.4.4**	—
9	Repeat steps 7 and 8 for crack height $a + \Delta a$ and continue until the limit to crack growth (step 5) or the specified design life is reached. The flaw is acceptable if the limit to crack growth is not exceeded in the design life.	**8.4.4, 8.4.5, 8.4.6**	—

Figure 2.12: General procedure for planar flaws [13]

2.5.2. Eurocode 1993-1-9

In EN 1993-1-9, fatigue assessment is based on S-N curves. Here, stress range (in MPa) versus the endurance (in cycles) is plotted on a log-log scale. Specific detail categories and the position on the graph specify the C, D and m parameters of the curve. These detail categories can be found in table 8.1 of EN 1993-1-9 [36]. When considering bridges, specifically orthotropic steel decks, more information about analysing the fatigue resistance can be found in chapter 9 of EN 1993-1-2 [35]. Hereunder, more details are presented where cracks can occur in orthotropic steel decks. A typical S-N curve can be found in figure 2.13.

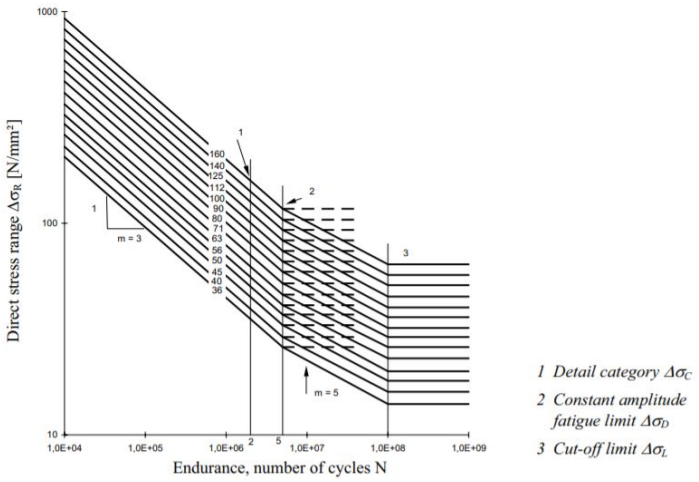


Figure 2.13: S-N curve [36]

With this S-N curve, the endurance can be determined for a given load situation. The applied number of load cycles divided by the endurance can be seen as the damage done by this load situation. For all load situation,

the cumulative value can serve as theoretical damage on the structural detail:

$$D = \sum_i D_i = \sum_i \frac{n_i}{N_i} \leq 1.0 \tag{2.23}$$

where:
D = Damage;
n = Applied number of cycles;
N = Maximum number of cycles that can be resisted.

Typical fatigue analysis methods that make use of S-N curves are the nominal stress method, the hot spot stress method and the effective notch method. From these the nominal stress method is the most simple but conservative method to perform [11]. What all these methods lack is the information in crack geometry throughout the service life. This is where the fracture mechanics approach has an advantage. The main difference in approach however is that an initial flaw is assumed in the fracture mechanics approach and thus neglecting the crack initiation stage in contrary to the S-N curves.

2.5.3. Eurocode 1993-1-10

The scope of EN 1993-1-10 lies in the selection of steel for fracture toughness and through thickness properties in welded elements. In welded elements, a risk may be present for lamellar tearing for which this code gives guidance in the design. The code provides the maximum allowed element thickness considering the following design parameters [34]:

- Steel grade;
- Toughness quality;
- Reference stress level;
- Reference temperature.

The resulting thicknesses (in mm) are presented in table 2.1 of EN 1993-1-10. A fracture mechanics approach is also available. The fracture mechanics parameters (i.e. J-integral and K_{Ic}) may be used to meet the toughness requirement. Here, a flaw must be introduced which complies with several conditions. One of which is that the crack increase should be determined during the next inspection interval [34]. Therefore, a propagated crack must be taken into account when choosing the fracture mechanics approach for meeting the toughness requirement.

3

Asymmetrical Four Point Bending Specimen

Through application of computational fracture mechanics, theoretical crack paths are determined in this chapter for an asymmetrical four point bending specimen. A similar procedure has been performed by Huang et al. [25] which serves as a reference model and is described in section 3.1. The main difference between the reference model and the model created for this thesis is that the former has used conventional finite element method while here the extended finite element method is used.

Preliminary to the study presented in this chapter, two underlying researches are conducted which are presented in appendices B and C. These comprises computational fracture mechanics in Abaqus and the integration of the Python scripting language with Abaqus. A comparison between the results of the current analysis and that of Huang et al. [25] is carried out. Finally, conclusions are drawn from the obtained results and comparison. The flow chard of the structure of this chapter is depicted in figure 3.1.

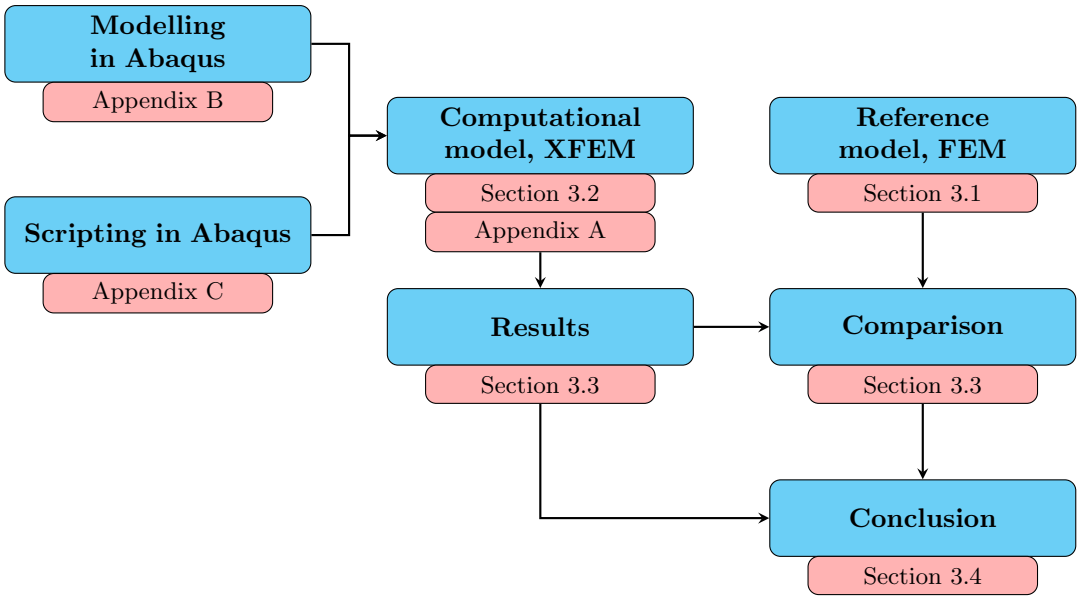
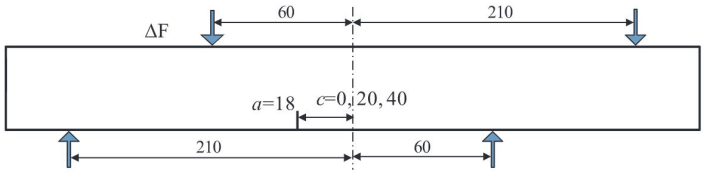


Figure 3.1: Structure of chapter 3

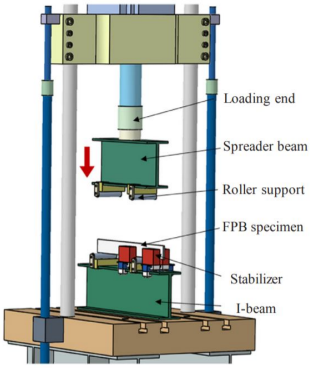
3.1. Reference Model - Huang et al.

For validation of the results obtained, a reference study is used which is presented in this section. An asymmetric four point bending test has been performed at a steel sheet with an initial crack by Huang et al. [25]. Conventional finite element method was used to model crack path and fatigue life assessment.

An initial through thickness crack, with a length of 18 mm, was implemented in the model after which crack propagation was modelled. In the figures presented below, the model and the test setup are illustrated. The initial crack offset is 20 mm from the midline of the specimen as presented in figure 3.2a. This specific test setup provides shear stresses in the region of the crack. As a result, mixed mode cracking may occur. In this specific setup, the crack will propagate with an angle. Therefore, the crack propagation angle has to be determined making the prediction of the crack path more challenging.



(a) Model including loading positions



(b) Test setup

Figure 3.2: Specimen of reference study [25]

3.1.1. Input variables

The input variables used in the reference model are listed below. The plate specimen has dimensions of 500x60x8.2 mm for width, height and thickness respectively. For representing steel in the Abaqus model, Young's modulus and Poisson's ratio were chosen 213 GPa and 0.3 respectively. The crack increment length per step in the analysis, which is discussed in the next section, was 1 mm.

Symbol	Parameter	Value	Unit
h	Height	60	$[mm]$
w	Width	500	$[mm]$
t	Thickness	8.2	$[mm]$
a	Crack increment length	1.0	$[mm]$
E	Young's modulus	213	$[GPa]$
μ	Poisson's ratio	0.3	$[-]$
ΔK_{th}	SIF threshold	148.60	$[MPa \cdot \sqrt{mm}]$
K_c	Critical SIF	4613.64	$[MPa \cdot \sqrt{mm}]$
ΔF_1	Loads closest to midline	460.98	$[MPa]$
ΔF_2	Loads furthest away from midline	131.71	$[MPa]$

Table 3.1: Input variables used in reference model [25]

At four positions, loads are applied on the modelled specimen. Two of which are located on the inner side of the plate and the remainder two more on the outside. The loads on the inside of the specimen, having a magnitude of 460.98 MPa, are spaced 60 mm away from the midline. The loads more on the outside of the specimen are spaced 210 mm away from the midline and have a magnitude of 131.71 MPa. In figure 3.3, the geometry of the specimen including the loading conditions are presented.

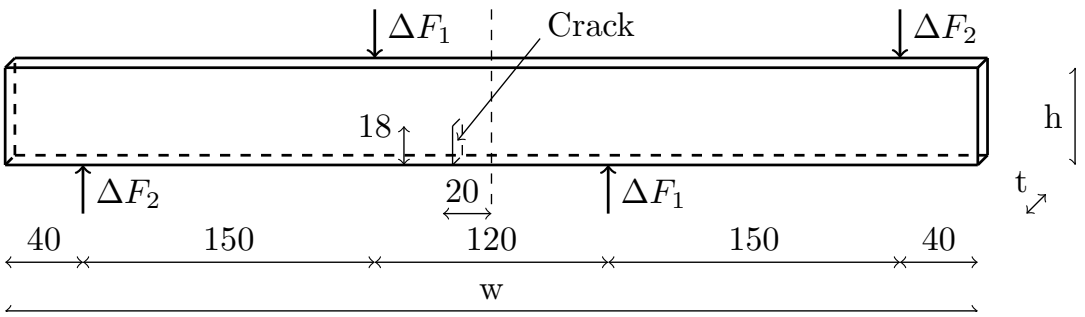


Figure 3.3: Loading conditions on specimen [25]

3.1.2. Calculation procedures

In the study performed by Huang et al. [25], conventional finite element method was applied for the calculation of contour integrals. From these contour integrals, stress intensity factors (SIF's) were obtained for each cracking mode (K_I, K_{II} and K_{III}) throughout the analysis. The crack propagation analysis is carried out using Python which can be used in the Abaqus pre- and post procedure.

In this report, a fixed crack increment size of 1 mm is assumed during the crack growth analysis. The analysis is limited within the region where the effective stress intensity factor ranges between the threshold- and the critical value ($K_{th} < K_{eff} < K_{cr}$). As treated in the literature review, the stress intensity factor threshold is the lower limit at which (theoretically) the crack starts to propagate. Below this value, no extension of the crack is assumed. Furthermore, the critical stress intensity factor is the upper limit of stable crack growth. From this value onward the crack may extend rapidly. Therefore, loading cycles are counted in the reference model for stress intensity factors between these two values. Finally, when the crack has grown up to 70% of the specimen height, the calculation is forced to stop as well.

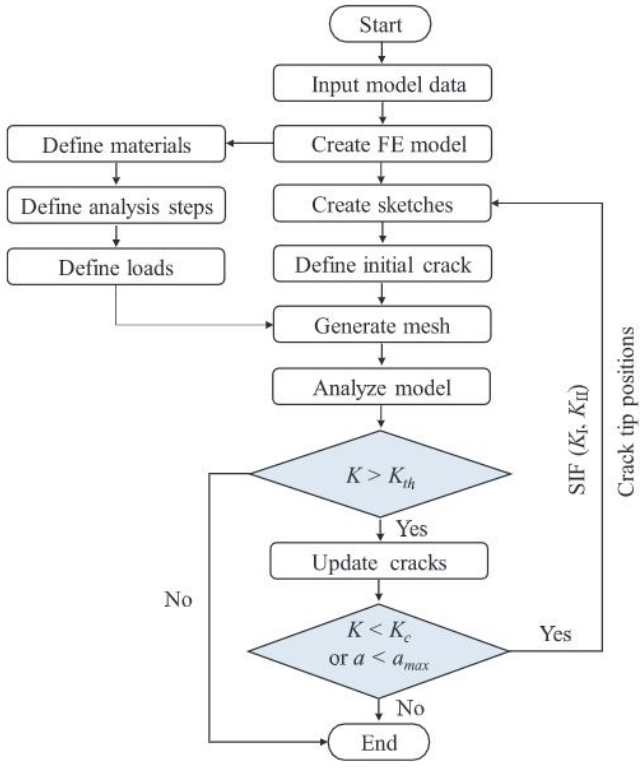


Figure 3.4: Flow chard of automatic crack propagation [25]

Crack propagation direction
Through Abaqus, the crack propagation direction can be determined through three different criteria. These criteria are, the maximum tangential stress criterion (MTS), the maximum energy release rate criterion and the $K_{II} = 0$ criterion (section B.1.4). However, Huang et al. [25] determined the crack propagation direction without the built in tool of Abaqus and has used the maximum circumferential criterion (MCSC):

$$\theta_0 = 2\arctan\left(\frac{K_I - \sqrt{K_I^2 + 8K_{II}^2}}{4K_{II}}\right) \tag{3.1}$$

Fatigue life assessment
To determine the fatigue life assessment, the NASGRO[1] model was used [25]:

$$\frac{\Delta a}{\Delta n} = -\frac{C(1-f)^m \Delta K_{eff}^m \left(1 - \frac{\Delta K_{th}}{\Delta K_{eff}}\right)^p}{(1-R)^m \left(1 - \frac{\Delta K_{eff}}{(1-R)K_c}\right)^q} \tag{3.2}$$

Various parameters that were used in this crack propagation relation by Huang et al. [25], are stated in the table below. The remainder, are variables in the formula ($\frac{\Delta a}{\Delta n}$ and ΔK_{eff}). The parameter f is defined as the crack opening function. The material parameters C and m are taken as $3.98 \cdot 10^{-13}$ and 2.88 respectively. The fracture toughness, K_c and threshold value, K_{th} are implemented in this relation where $K_c = 4170\ MPa\cdot\sqrt{mm}$ and $K_{th} = 148.6\ MPa\cdot\sqrt{mm}$. Finally, the stress ratio R was used 0.1 and the exponents, p and q, were taken 0.5.

Symbol	Value	Unit
C	$3.98 \cdot 10^{-13}$	[−]
m	2.88	[−]
ΔK_{th}	148.6	$[MPa\cdot\sqrt{mm}]$
K_c	4613.64	$[MPa\cdot\sqrt{mm}]$
R	0.1	[−]
p	0.5	[−]
q	0.5	[−]

Table 3.2: Constants used in NASGRO model

Finally, the effective stress intensity factor range was calculated by combining mode I and II through the following formula:

$$\Delta K_{eff} = \sqrt[4]{(\Delta K_I)^4 + 8(\Delta K_{II})^4} \tag{3.3}$$

3.1.3. Results

Huang et al. [25] has performed various calculations, some of which are presented in figure 3.5. These are results for various loading modes and initial crack offsets. The reference model used for this thesis is the crack trajectory named 'U-C-1' in figure 3.5. A more detailed graph of both the predicted and measured crack trajectory is shown in figure 3.6a.

[1] NASGRO provides software, used to anaylyse fracture and fatigue. This software is developed by SwRI and NASA. https://www.swri.org/consortia/nasgro

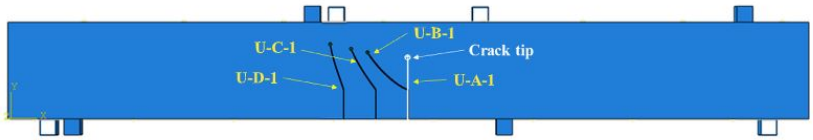
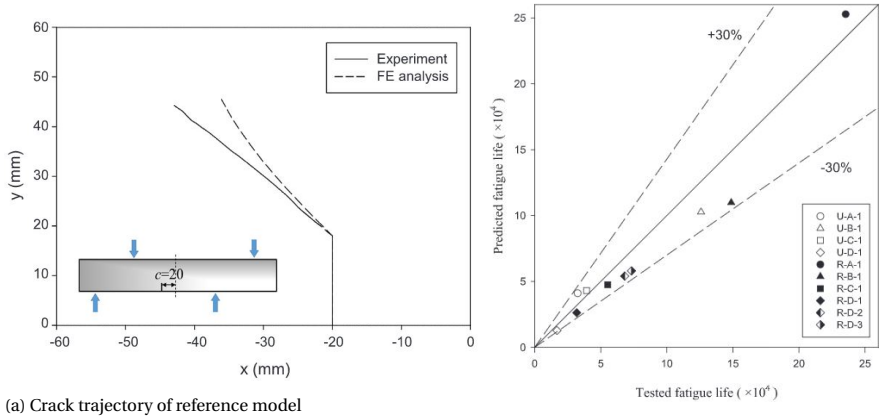


Figure 3.5: Crack trajectories of various specimens [25]



(a) Crack trajectory of reference model

(b) Fatigue life, predicted and tested

Figure 3.6: Results from reference model [25]

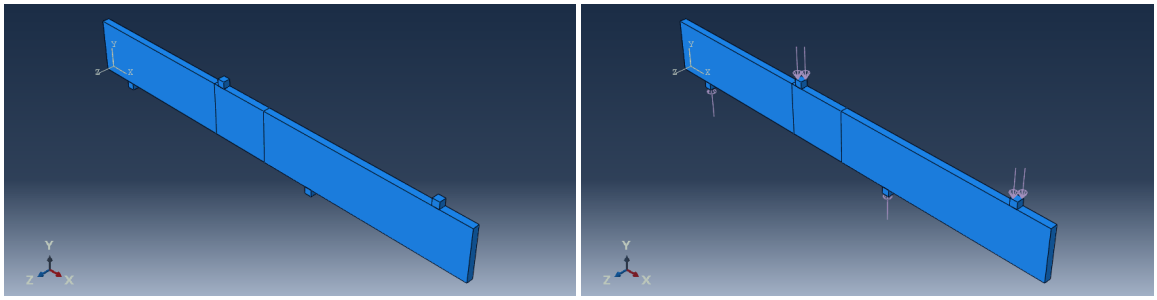
The modelled propagating crack by Huang et al. [25] showed good comparison to the experimental values. However, the difference increases with an increase of the crack length. The accumulative number of cycles determined from the finite element analysis up to failure was 43,113 cycles. From the experiment this was approximately 40,000 cycles [25] as can be seen from figure 3.6b (label U-C-1). The exact number of cycles was not mentioned in the research paper. The value of 40,000 has been extracted from the graph shown in figure 3.6b.

3.2. Computational Model

In this research, a similar model as described in the previous section was constructed. The main difference is that the extended finite element method (XFEM) was used instead of conventional finite element method (FEM). XFEM can be used for crack propagation performed by Abaqus itself and for the ability not to remesh the structure [1]. Since mesh refinement around the crack tip is expected to be needed for accurate results, remeshing the structure is inevitable (this is elaborated in chapter 3.2.2) and is therefore incorporated in a Python script. Whether remeshing is needed is investigated in section 3.3.4. Furthermore, contour integrals can only be evaluated for stationary cracks in Abaqus. Therefore, the built in crack growth option in Abaqus was not used but a stationary crack was modelled as is explained in more detail in chapter 3.2.3. More details into computational fracture mechanics in Abaqus and integrating Python with Abaqus is presented in appendices B and C.

3.2.1. Geometry, loading conditions and crack insertion

The geometric information of the model as adopted from Huang et al. [25]. The parameters are discussed in section 3.1.1. The plate specimen has a width, height and thickness of 500 mm, 60 mm, and 8.2 mm respectively. The loading points are represented by the four cubes which are connected to the main plate using tie constraints. Local partitioning of the cracked area was applied for mesh refinement. This is discussed in more depth in section 3.2.2. Similarly as described in section 3.1.1, the inner loads have a value of 460.98 MPa and the outer loads a value of 131.71 MPa.

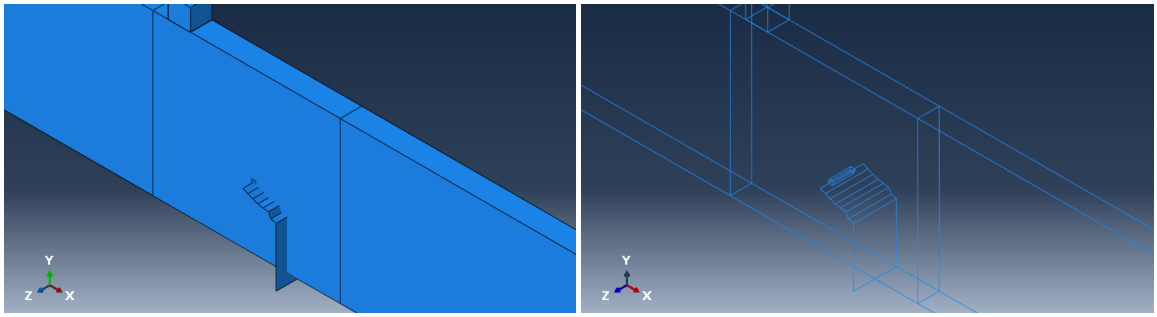


(a) Structure

(b) Structure with supports and loads

Figure 3.7: Abaqus model of the assembled structure

An extruded 3D shell part is modelled to represent the crack. This crack was inserted at the same location as in the reference model (see section 3.1). This 3D shell part is slightly wider (2x4 mm wider) than the sheet's thickness to ensure that at both sides the crack part extends out of the plate (see figure 3.8). Also the 3D shell part extends 4 mm out from the bottom of the plate. M. Levén [31] showed that modelling the crack in this particular manner leads to better convergence of the contour integral output and is therefore applied here as well. Additionally, this is recommended by the Abaqus/CAE user's guide [1].



(a) Shaded view

(b) Wireframe view

Figure 3.8: Assembled model of the specimen in Abaqus including a crack

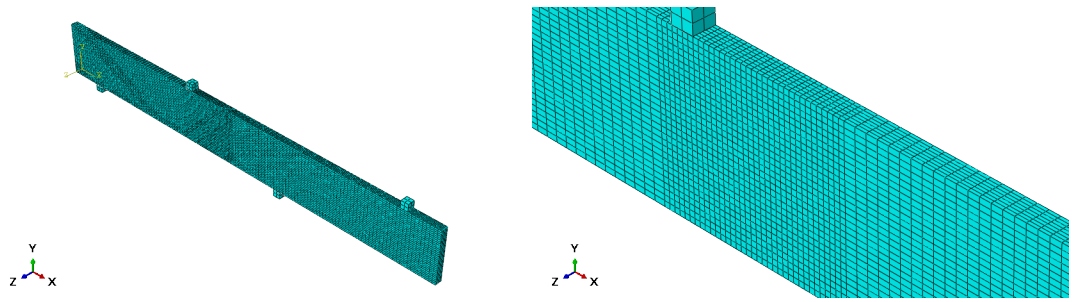
3.2.2. Meshing the structure

A research on convergence of contour integral output values of stationary XFEM crack analysis in Abaqus was performed by M. Levén [31]. The recommendations that followed from that research have been applied in this model. In order to have fine mesh around the crack tip, a partitioned zone in the sheet is created where locally the mesh can be refined. Here, the mesh will be more fine than the coarse mesh outside this zone (see figure 3.9). The most important recommendations by M. Levén are in short [31]:

- Use conformed mesh at the crack tip region;
- Apply high mesh density at the crack tip region;
- Apply elements with reduced integration since this has minor effect on the SIF output and reduces the computation time significantly;
- Use at least four contour integrals in the analysis output.

Brick elements of type C3D8R were used for the entire model. Only linear elements can be used for XFEM crack analysis in Abaqus. Moreover, no wedge elements can be used inside the contour evaluation region since this is not supported by Abaqus for XFEM cracks [1]. The structured meshing technique was used for meshing of the global model. Structured mesh showed significantly shorter computation time and similar results compared to the sweeping technique. At the mesh refinement region, sweeping was used with the advancing front algorithm. All elements are assigned hexagonal to prevent wedge elements in the contour integral region. Global mesh size was chosen 4 mm and local seeding was used in the mesh refinement region. This resulted in mesh sizes of approximately 2x2x1 mm for Δx, Δy and Δz respectively. Around the crack

front, local mesh was used of approximately 0.1x0.1x1 mm for Δx, Δy and Δz respectively. A mesh sensitivity study was performed and is presented in section 3.3.4. The meshed model and the mesh refinement region are presented in figure 3.9.

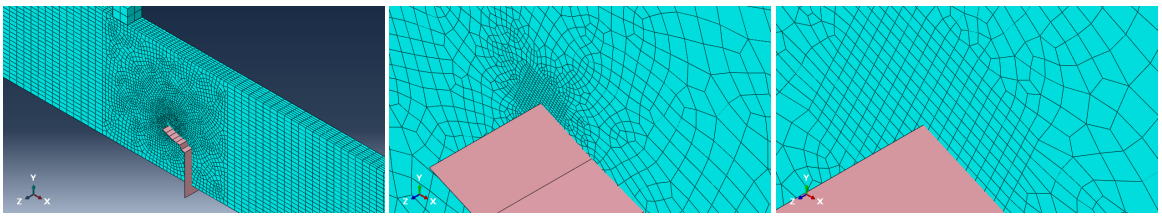


(a) Global view

(b) Zoomed at mesh refinement region

Figure 3.9: Meshed model

The elements around the crack tip are oriented in such a way that the elements are aligned with the crack front. This is illustrated in figure 3.10c. A partitioning was incorporated in the script to ensure local seeding and thus control over element size. This region around the crack tip where mesh alignment is applied is illustrated in figure 3.10c. Abaqus evaluates contour integrals around the crack front and are thus evaluated in this region as well [1].



(a) Zoomed at crack

(b) Zoomed at crack front

(c) Extremely zoomed at crack front

Figure 3.10: Abaqus model of the sheet, meshed

3.2.3. Methodology of crack propagation

As contour integrals can only be evaluated for stationary cracks, a method is needed to model crack propagation. A suitable method can be by using the Python scripting interface of Abaqus. Here, the crack can be updated with a crack increment after each calculation. This methodology has been followed in various studies. These studies include, CT-specimens [10], centre cracked specimens [17] and bending specimens [25]. A preliminary study on scripting with Abaqus can be found in appendix C.

The procedure of the script that has been composed is illustrated in figure 3.11. A CAE-file was constructed of an uncracked asymmetrical four point bending specimen. This file contains all the unchanged modelling options as parts, material properties, boundary conditions, loads, mesh size and type of elements. After running the script, this CAE-file is opened automatically. Afterwards, all the modifications coded in the script, are performed in this CAE-file. After opening the CAE-file, a part is created to represent the crack and is assigned an XFEM crack. Remeshing is performed to control the size and orientation of the elements around the crack front throughout the entire crack propagation analysis (as is explained in chapter 3.2.2). After generation of the mesh, the calculation in Abaqus is started automatically. Here, output results are stress intensity factors for all contour integrals. The new crack increment angle is then calculated with these stress intensity factors according to equation 3.1. Using a fixed crack increment size and a calculated crack increment angle, the newly formed crack is determined. Then, a 'for loop' reruns the script and the procedure is again followed. The number of loops is user defined in the Python script. The resulting number of cycles for the crack extension for each loop is calculated through integration of the Paris' law. This methodology is illustrated in a flow chard in figure 3.11.

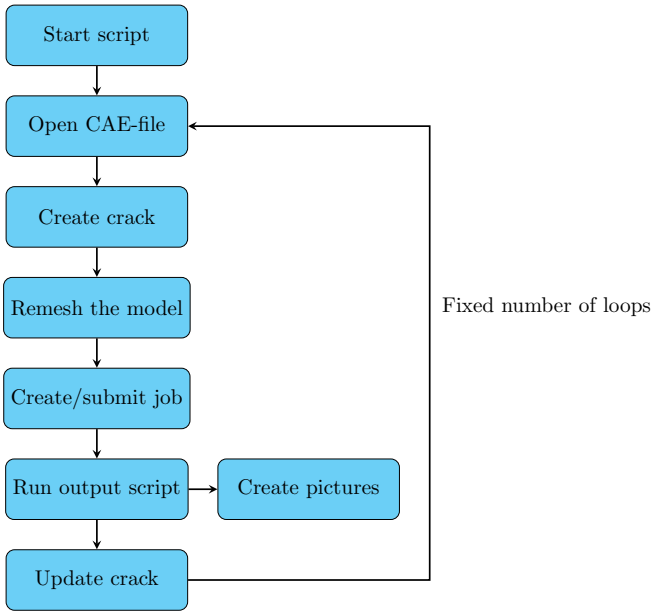


Figure 3.11: Flow chard of script

3.3. Results and Comparison

In this section, the results presented. A comparison study is performed between the results and the reference model for validation. The results of stress intensity factors, crack trajectories and fatigue life assessment are presented. Finally, a parametric study is performed.

3.3.1. Stress intensity factors

As stress intensity factors are used to determine both crack path as fatigue life assessment, these are treated first. As mentioned in [1] and [31], the first contour integral often shows unrealistic values. Therefore, it is recommended that the first contour integral values should be disregarded. The stress intensity factors for each contour integral number are plotted to investigate whether the first contour integrals gives unrealistic values. The resulting stress intensity factors are plotted in the graphs below.

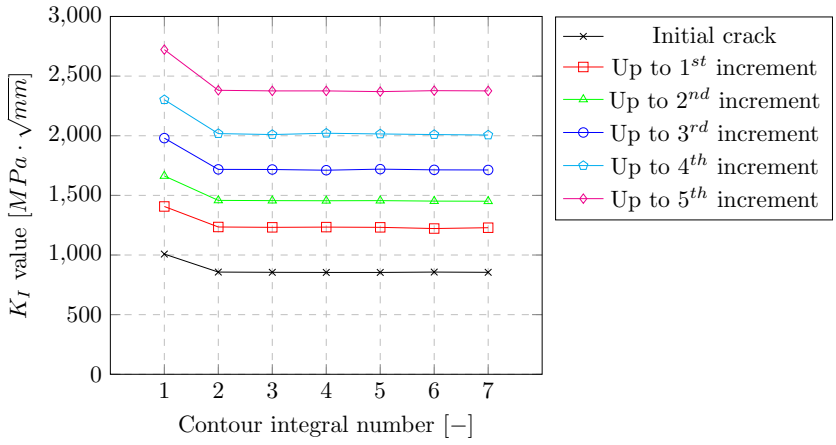


Figure 3.12: K_I per contour integral up to 5^{th} increment

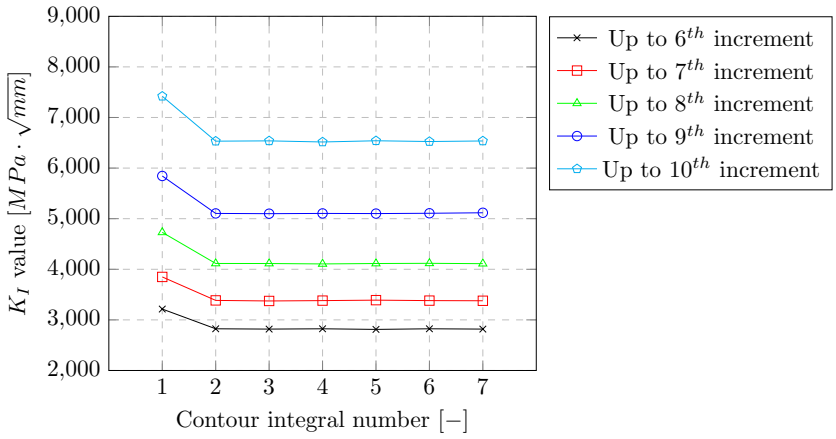


Figure 3.13: K_I per contour integral for increment 6 up to 10

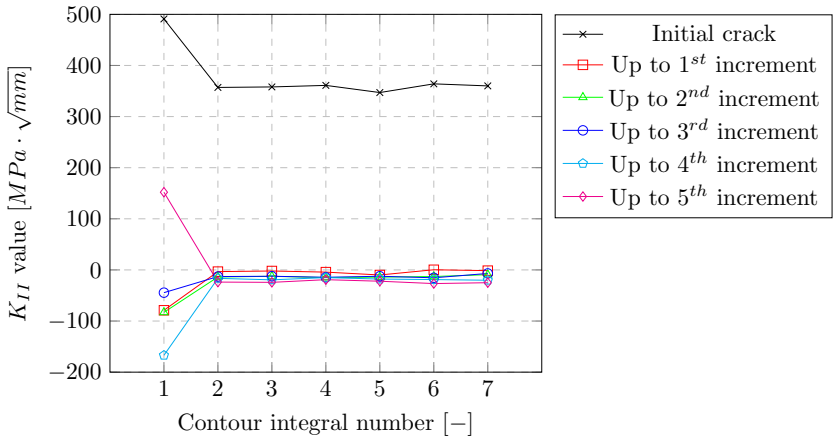


Figure 3.14: K_{II} per contour integral up to 5^{th} increment

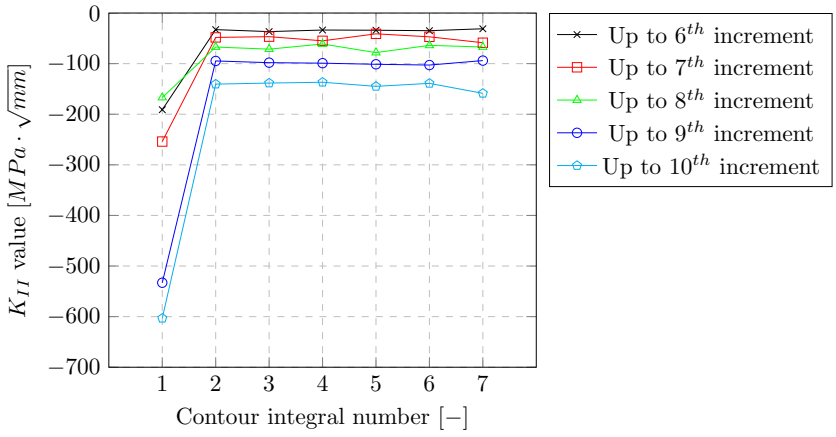


Figure 3.15: K_{II} per contour integral for increment 6 up to 10

From figure 3.12-3.15, the first contour integral shows deviation from the trend for all results. All other contour integrals showed approximately the same values for each calculation. Therefore, the mean average is taken from contour integral numbers 2-7. The resulting values of K_I and K_{II} are plotted versus the increment number in figure 3.16.

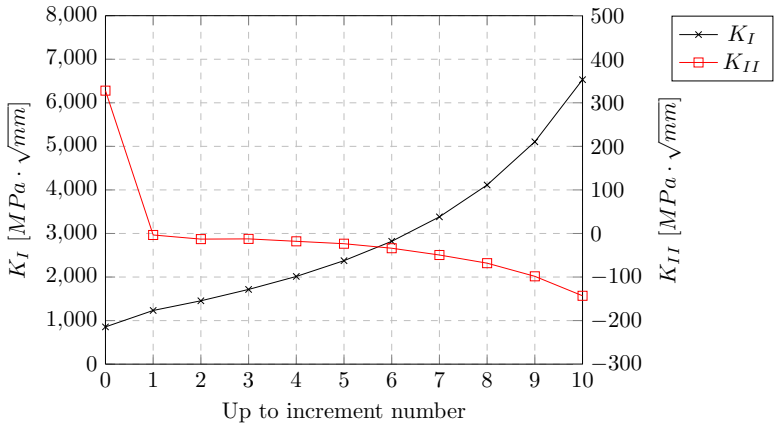


Figure 3.16: K_1 and K_{II} per increment number

Through equation 3.3, the effective stress intensity factors are determined; the results versus crack increment number are plotted in figure 3.17. Results regarding the initial crack is assigned increment number 0. K_{eff} follows a somewhat exponential trend leading to higher values of K_{eff} for a larger crack. These results for K_{eff} are used to analyse the fatigue life of this specimen. The results for crack trajectory and fatigue life assessment are presented in this order in the following sections. The exact values of K_{eff} can be found in table 3.3.

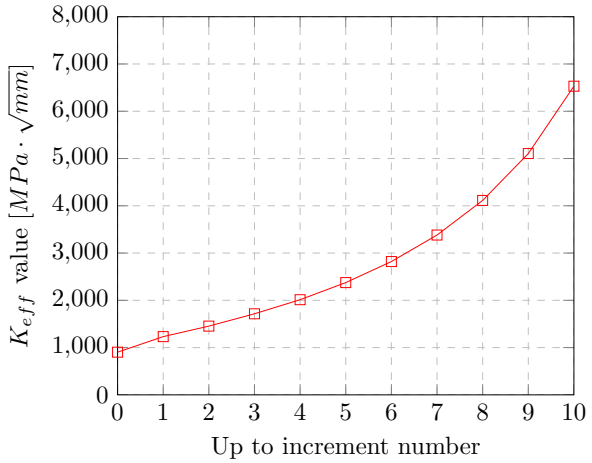
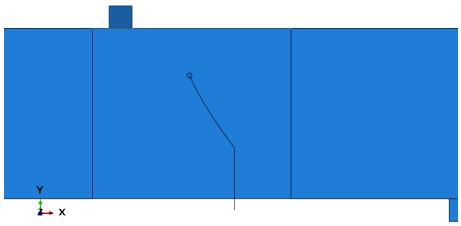


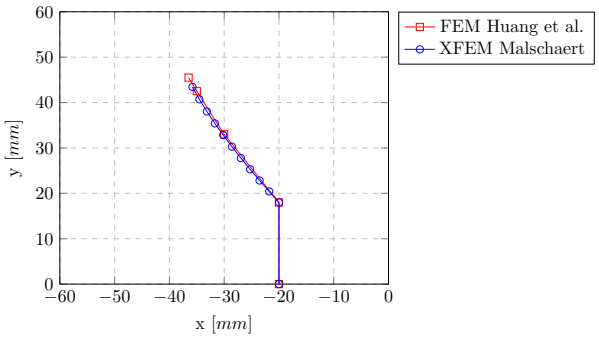
Figure 3.17: K_{eff} per increment number

3.3.2. Crack trajectory

Illustrations of the crack path per increment are presented in appendix A.2. The crack follows a smooth path towards the loading point as illustrated in figure 3.18. This was also found by Huang et al. [25]. Results of crack trajectories from Huang et al. and from XFEM analysis are presented in figure 3.18b. Good agreement was found between the two results. Therefore, crack trajectory is accurately determined through the XFEM analysis. The results of crack trajectory from Huang et al. [25] have been extracted from a graph; exact values could not be obtained. Therefore, no value is presented which quantifies the difference in results.



(a) Results of crack trajectory



(b) Comparison of results

Figure 3.18: Results of crack trajectory

3.3.3. Fatigue life assessment

A fatigue life assessment was performed from initial crack, up to a crack spanning 70% of specimen height (42 mm) as also performed by Huang et al. [25]. Paris' law has been used to determine the cycles needed for each crack increment:

$$\Delta n = \frac{\Delta a}{C(\Delta K_{eff})^m} \tag{3.4}$$

Parameters C and m are selected $3,98 \cdot 10^{-13}$ and 2.88 respectively, as chosen in [25] and recommended in [13]. The calculated effective SIF should be used to determine the number of cycles for the next increment. For example, K_{eff} regarding the initial crack is used to determine the number of cycles needed for the first crack increment.

Increment number [−]	ΔK_{eff} $[MPa \cdot \sqrt{mm}]$	Δa $[mm]$	Δn [−]
1	903	3	22,729
2	1232	3	9,468
3	1454	3	5,876
4	1715	3	3,652
5	2013	3	2,302
6	2375	3	1,430
7	2820	3	872
8	3381	3	517
9	4113	3	294
10	5105	3	158
	$\sum =$	30	47,298

Table 3.3: Results from fatigue cycles calculation

The fatigue life assessment data, as shown in table 3.3, are plotted in figure 3.19. The crack length shows an increasing trend where extreme crack growth is observed after 40,000 cycles.

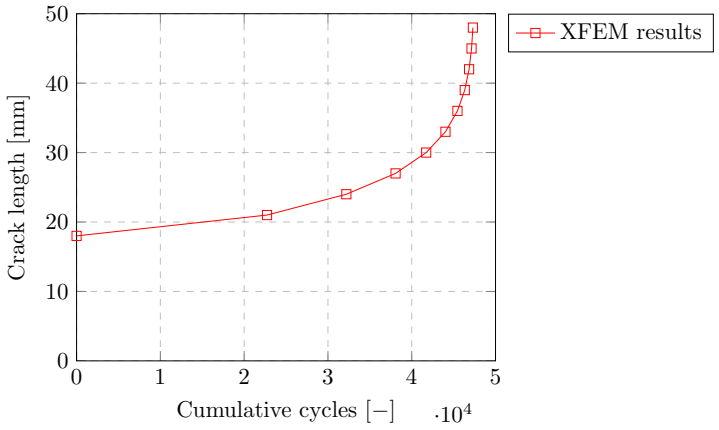


Figure 3.19: Fatigue life assessment

From the XFEM analysis, a total of 47,298 cycles in a crack reaching a height of 70% specimen height. From the conventional FEM calculation performed by Huang et al. [25], a total of 43,113 cycles were needed to reach the same height. From this the difference result in:

$$difference: \quad (47,298 - 43,113)/43,113 \cdot 100\% = 9.7\%$$

With a difference of 9.7%, the crack propagation using XFEM shows good agreement with the conventional FEM method with respect to fatigue life assessment. However, it should be noted that Paris' law was used for XFEM results while NASGRO model was used for conventional FEM by Huang et al. [25]. The reason for choosing Paris' law is that this will be applied in chapters 4 and 5. The main study in this thesis is crack propagation modelling in an orthotropic steel deck. To model fatigue, a simple relation is preferred. Therefore, it is of interest how the Paris law compares to the NASGRO model.

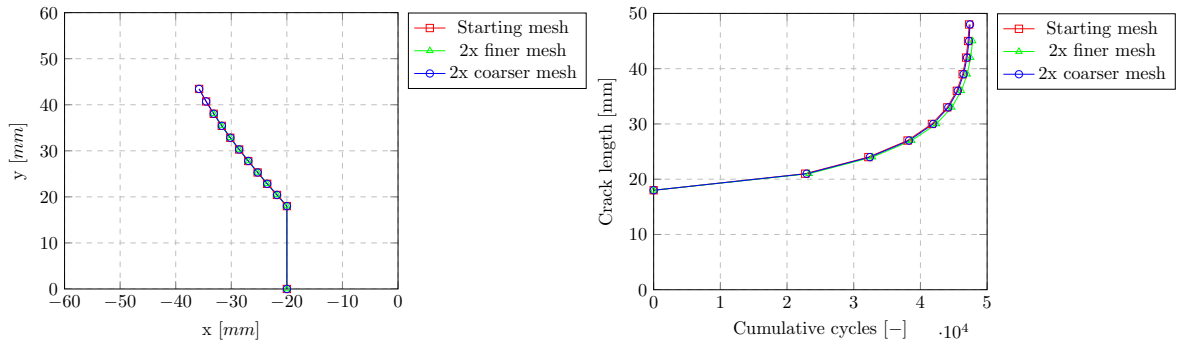
3.3.4. Crack trajectory - Parametric study

The effect of parameters on the results of crack trajectory and fatigue life assessment have been investigated. The following parameters were examined:

- Crack tip mesh size;
- Crack increment size;
- Mesh orientation at crack tip.

Crack tip mesh size

In the model, three types of regions are distinguished with respect to meshing. These regions are, the global mesh region, the fine mesh region and the crack tip mesh region (see section 3.2.2). As the contour integrals are evaluated inside the crack tip mesh region, this is of most interest for investigation. The starting mesh is chosen to be as explained in section 3.2.2. The results are presented in the graphs in figure 3.20. Results of crack trajectory did not show difference. Moreover, minor differences were observed in fatigue life assessment. As varying the mesh size around the crack tip did not result in significant differences, the crack tip mesh size was chosen fine enough for accurate results.



(a) Crack trajectories

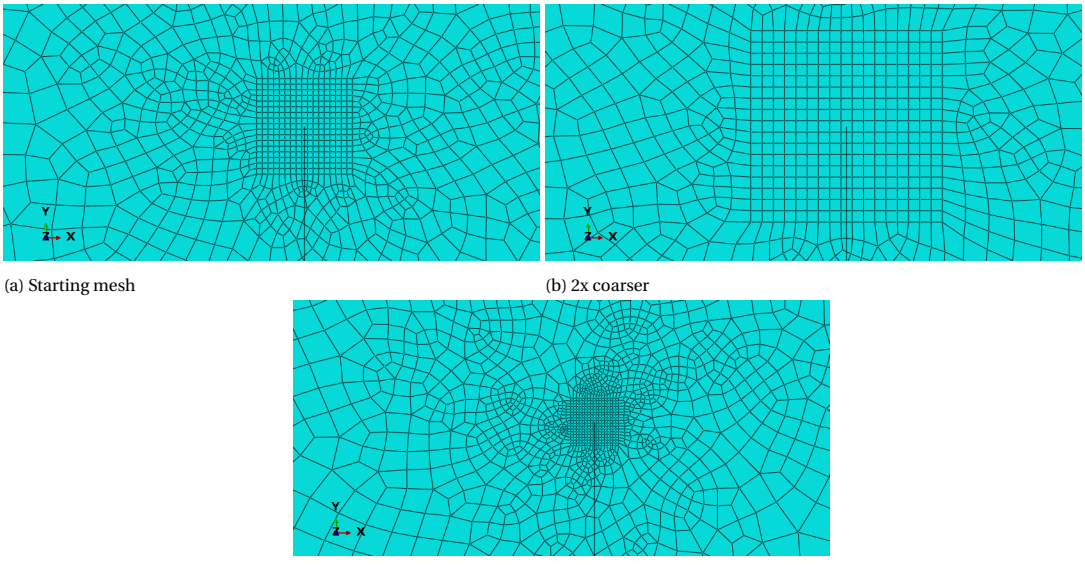
(b) Fatigue life assessment

Figure 3.20: Results for various mesh sizes

The exact mesh sizes, for elements surrounding the crack tip, are presented in table 3.4. In the z-direction, the element size was kept constant as this is in the thickness direction of the plate. The mesh surrounding the crack tip, for all calculations performed, are presented in figure 3.21.

Model	Δx **[mm]**	Δy **[mm]**	Δz **[mm]**
Starting mesh	0.088	0.088	1
2x finer mesh	0.044	0.044	1
2x coarser mesh	0.176	0.176	1

Table 3.4: Approximate crack tip element size per model



(a) Starting mesh

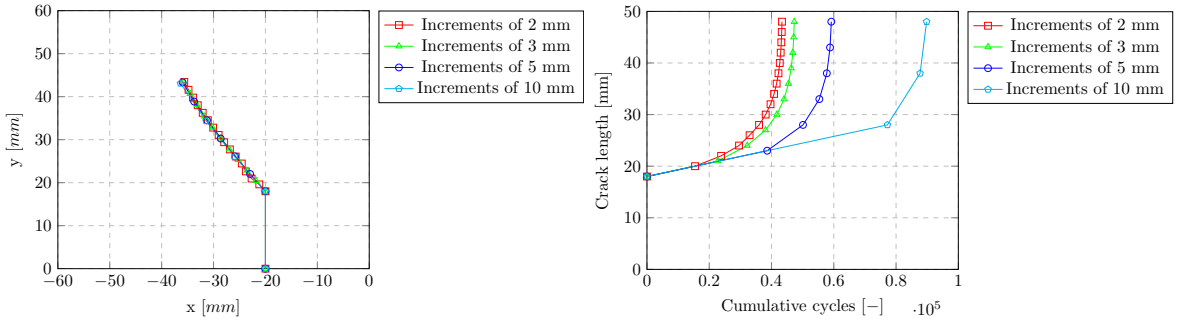
(b) 2x coarser

(c) 2x finer

Figure 3.21: Crack tip mesh sizes

Crack increment size

Crack increments sizes of 2, 3, 5 and 10 mm have been investigated. The results are plotted in figure 3.22. Varying crack increment sizes did not show significant differences for crack trajectory. As the crack path is (almost) a straight line, crack angles along the path remain constant. As a result, Varying the increment sizes does not deviate the crack path as the angle remains constant. However, for fatigue life assessment, the results where greatly affected by the increment size. Results of increment sizes of 2- and 3 mm showed best comparison whereas results for increment sizes of 5- and 10 mm showed significant differences.



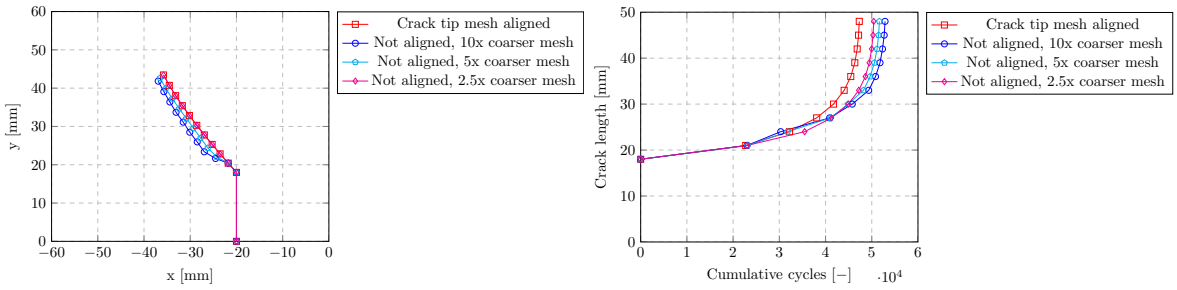
(a) Crack trajectories

(b) Fatigue life assessment

Figure 3.22: Results for various crack increment sizes

Mesh orientation at crack tip

Creating a mesh alignment as presented in figure 3.10c results in high effort regarding coding a Python script. Furthermore, remeshing after updating the crack is needed. Using XFEM, it might be possible to predict the crack path without remeshing the structure as the crack does not need to follow element boundaries (see section 2.3 for more details). Therefore, the crack tip box in which mesh is aligned with the crack, was discarded to investigate the crack trajectory without mesh alignment. As a larger region now holds refined mesh, a significantly higher computation time was observed. Therefore, more coarse mesh was investigated compared to the results from the aligned mesh. Approximately 10-, 5- and 2.5 times as coarse mesh were examined. The results are presented in the graphs in figure 3.23. Some differences where observed in both crack trajectory as well as fatigue life assessment. The results are converging to the results including mesh alignment as the more fine mesh was chosen. When the mesh was chosen 2.5 times as coarse, the results for crack trajectory complied with the model where mesh alignment with the crack tip was applied. Moreover, minor differences remained for fatigue life assessment (approximately 6.20%). As a result, XFEM can be used for contour integral evaluation without the need for remeshing. The mesh size used for the various models are summarised in table 3.5, the crack increment length was in all cases 3 mm. In figure 3.24 the crack tip mesh is illustrated for both with and without mesh alignment.



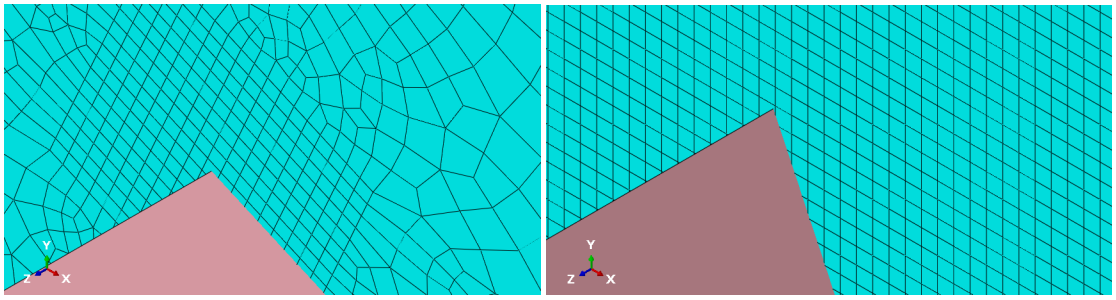
(a) Crack trajectories

(b) Fatigue life assessment

Figure 3.23: Crack trajectories without crack tip mesh alignment for various element sizes

Model	$\Delta x\ [mm]$	$\Delta y\ [mm]$	$\Delta z\ [mm]$
Crack tip mesh aligned	0.09	0.09	1.00
Not aligned, 10x coarser mesh	1.00	1.00	1.00
Not aligned, 5x coarser mesh	0.50	0.50	1.00
Not aligned, 2.5x coarser mesh	0.25	0.25	1.00

Table 3.5: Approximate element size per model



(a) Crack tip mesh aligned with crack

(b) Crack tip mesh not aligned with crack

Figure 3.24: Meshed model extremely zoomed at crack tip

3.4. Conclusion

Computational fracture mechanics was used for analysing an asymmetric four point bending specimen. Extended finite element method was applied for contour integral evaluation and for obtaining stress intensity factors for stationary cracks. Using Python scripting, crack increments were added to the model in order to model crack propagation. Validation was used by conventional FEM results of Huang et al. [25]. The following conclusions are drawn from this chapter:

- The obtained results for crack trajectory showed good agreement with the results from Huang et al. [25].
- Good stability was obtained in stress intensity factor output for element sizes of approximately 0.1 mm. Seven contours were evaluated in each step in the analysis. The first contour showed deviation from the remaining six. As a result, The first contour integral output was discarded from further analysis and the mean average was taken from contours 2 up to 7.
- Crack increment sizes of 2, 3, 5 and 10 mm are investigated. All results of the crack trajectory are approximately equal and followed (almost) a straight line. However, large differences were observed in fatigue life assessment. Increment sizes of 2- and 3 mm showed similar results while 5- and 10 mm resulted into significantly higher accumulative number of cycles for similar crack lengths.
- Element sizes around the crack tip has been investigated. In the first calculation element size around the crack tip was used of 0.088x0.088x1 mm for Δx, Δy and Δz respectively. 2x finer (0.044x0.044x1 mm) and 2x coarser (0.176x0.176x1 mm) mesh has investigated afterwards. All results showed similar crack trajectories and fatigue life assessment. As a result, element sizes of approximately 0.1 mm result into accurate results for this specific model.
- The effect of mesh alignment with the crack tip has been examined. Element sizes of approximately 0.1 mm were used for the model including mesh alignment (0.088x0.088x1 mm for Δx, Δy and Δz respectively). Element sizes of 2.5, 5 and 10 times as coarse were investigated at which no mesh alignment was applied. Differences in both crack trajectory and fatigue life assessment were observed. The results converged to the results including mesh alignment when more fine mesh was used. Choosing an element size of 0.25 mm yielded in very close results for crack trajectory and a maximum difference of 6.20% was observed for fatigue life. At this stage, the crack had propagated for approximately 30 mm. As a result, no remeshing is needed for modelling crack propagation using XFEM.
- Fatigue life was analysed using Paris' law. Validation was performed by conventional FEM performed

by Huang et al. [25] using the NASGRO propagation relation. A difference of 9.7% was observed after a crack growth of approximately 30 mm (initial crack length is 18 mm). Therefore, the fatigue life assessment, using XFEM with Paris law, is in good agreement with the results from conventional FEM using NASGRO propagation relation.

4

Semi-Elliptical Centre Cracked Plate

In chapter 3, a through thickness crack was examined in an asymmetrical four point bending specimen. Stable results were obtained in contour integral output. Furthermore, a Python script was composed in order to mimic crack propagation in Abaqus. In this chapter, a semi-elliptical surface crack in a centre cracked plate is examined as well as 3D crack propagation. The structure including a surface crack is presented in figure 4.1.

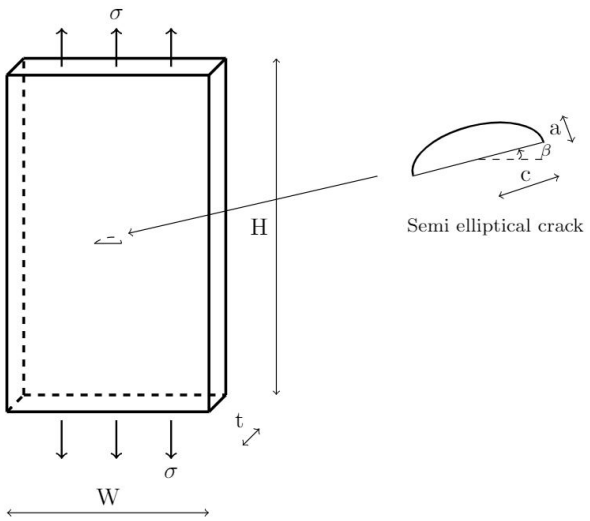


Figure 4.1: Semi-elliptical centre cracked plate

For defining a point along the crack front, θ is introduced as shown in figure 4.2. To obtain angle θ, a single local coordinate of the crack front (y) and the crack depth a are needed. Subsequently, it should be known at which side of the y-axis the point is located as for each side a point is located which holds an equal y-value.

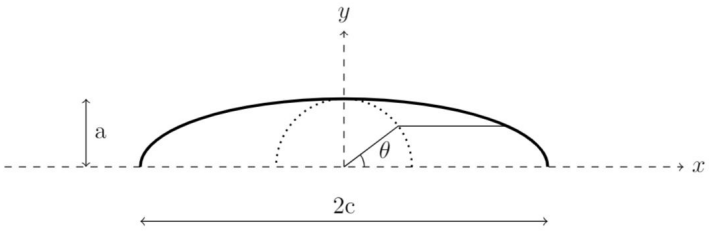


Figure 4.2: Definition angle theta in crack

In this chapter, a stationary surface crack is firstly examined. These results are validated using British Standard 7910 (BS7910) [13]. Secondly, a stationary inclined surface crack is treated. Here, validation is obtained through research papers as well as fracture and fatigue software Franc3D [14]. Finally, a propagating inclined surface crack is examined. Moreover, a methodology is followed which is explained in detail in appendix D. The results are validated with Franc3D [14]. The structure of this chapter is illustrated in figure 4.3.

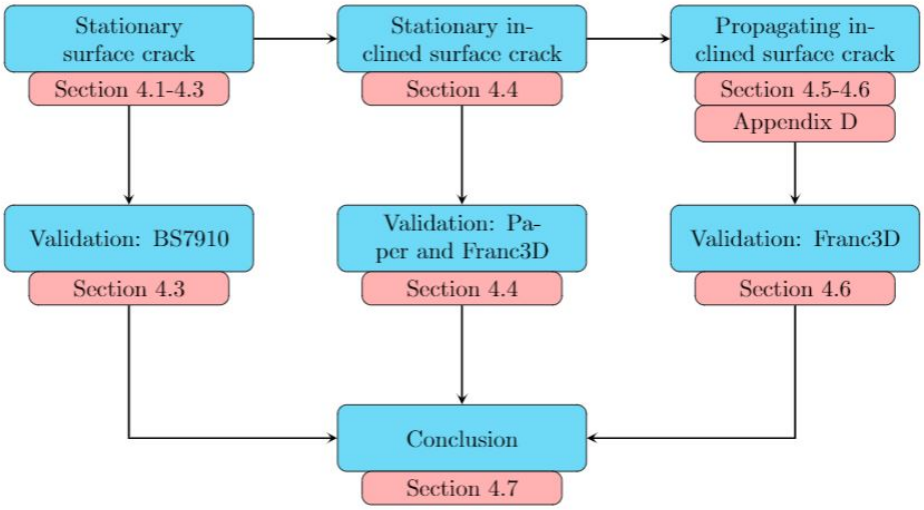


Figure 4.3: Structure of chapter 4

4.1. BS7910 solution

For obtaining an understanding of surface flaws and mesh size sensitivity, a semi-elliptical surface flaw is examined as explained in the introduction of this chapter. The input parameters are as presented in table 4.1 (see figure 4.1 for definitions of geometrical parameters).

Symbol	Parameter	Value	Unit
H	Height of plate	100	$[mm]$
W	Width of plate	100	$[mm]$
t	Thickness of plate	15	$[mm]$
E	Young's modulus	210,000	$[MPa]$
μ	Poisson's ratio	0.3	$[-]$
a	Crack depth	2	$[mm]$
c	Half crack length	4	$[mm]$
σ	Remote tensile stress	100	$[MPa]$
β	Initial angle	0	$[^\circ]$

Table 4.1: Input parameters

For verification of results obtained by computational fracture mechanics, analytical solutions are used. These solutions, were extracted from BS7910 [13]. Considering a semi-elliptical surface crack in a plate subjected to tension for $0 < a/2c < 0,5$, the following equations apply:

$$K_I = Y\sigma\sqrt{\pi a} \tag{4.1}$$

$$Y\sigma = M_m\sigma \tag{4.2}$$

$$M_m = \left[M_1 + M_2\left(\frac{a}{t}\right)^2 + M_3\left(\frac{a}{t}\right)^4\right]\frac{g\,f_\theta}{\Phi} \tag{4.3}$$

$$M_1 = 1.13 - 0.09\left(\frac{a}{c}\right) \tag{4.4}$$

$$M_2 = \left[\frac{0.89}{0.2 + (a/c)}\right] - 0.54 \tag{4.5}$$

$$M_3 = 0.5 - \frac{1.0}{0.65 + (a/c)} + 14\left(1 - \frac{a}{c}\right)^{24} \tag{4.6}$$

$$g = 1 + \left[0.1 + 0.35\left(\frac{a}{t}\right)^2\right](1 - \sin\theta)^2 \tag{4.7}$$

$$f_\theta = \left[\left(\frac{a}{c}\right)^2 \cos^2\theta + \sin^2\theta\right]^{0,25} \tag{4.8}$$

$$\Phi = \left[1 + 1.464\left(\frac{a}{c}\right)^{1.65}\right]^{0.5} \tag{4.9}$$

where:

K_I = Stress intensity factor for opening mode in $MPa \cdot \sqrt{mm}$;
Y = Geometric function;
σ = Remote tensile stress in MPa;
a = Crack depth in mm;
c = Half crack length in mm;
t = Plate thickness in mm;
θ = Surface crack angle for specifying position at crack front (see figure 4.2 for more details).

Following these relations and the input parameters as presented in table 4.1, the results are obtained as presented in figure 4.4. The symmetry of the structure and the crack shape is reflected in the results for stress intensity factor K_I. Moreover, the values for K_I around $\theta = 90°$ are highest. As a result, the crack tends to propagate in the thickness direction of the plate faster than in the width direction for this specific situation. The definition of θ for a semi-elliptical crack can be found in figure 4.2.

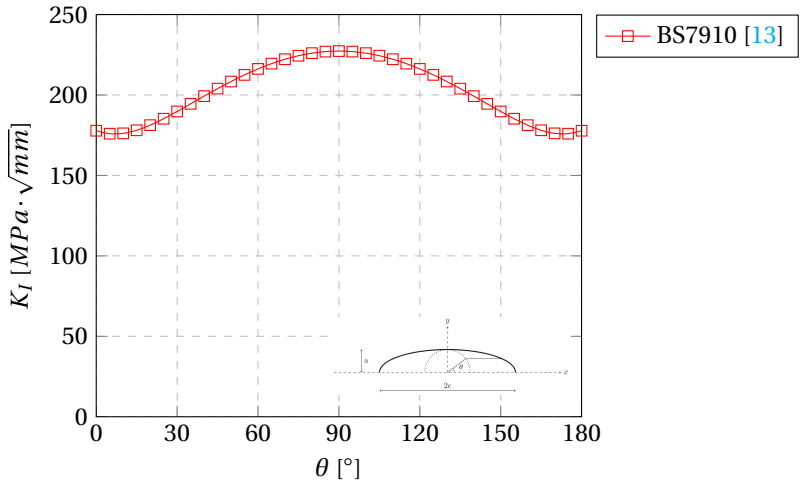


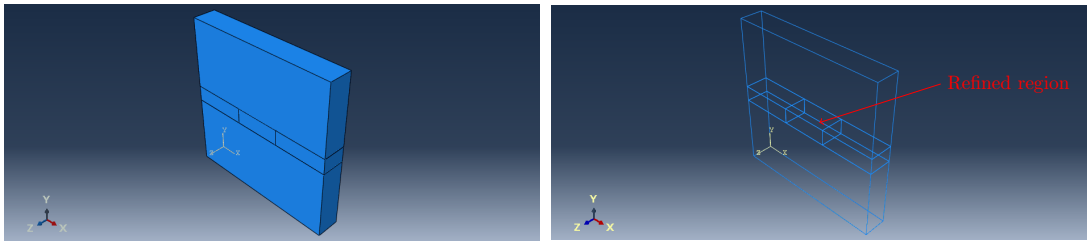
Figure 4.4: K_I results along crack front according to BS7910

4.2. Computational model

Similar to the studies presented in chapter 3 and in appendices B and C, the extended finite element method (XFEM) in Abaqus is used for modelling. In this section the model itself is presented. Firstly the geometry and boundary conditions are discussed and subsequently, meshing of the structure. All parameters that are used are listed in table 4.1. In figure 4.1 the corresponding geometry parameters are visualised.

4.2.1. Plate geometry and boundary conditions

A 3D extruded solid plate is modelled to represent the plate as presented in figure 4.5. For enabling mesh refinement, various parts were modelled and connected by tie constraints. This refined region, is located at the centre of the plate as is depicted in figure 4.5b. Linear elastic material with Young's modulus and Poisson's ratio of 210,000 MPa and 0.3 respectively, was selected for representing steel.



(a) Shaded view

(b) Wireframe view

Figure 4.5: Plate model in Abaqus

The plate is restricted at the bottom surface for translational motion and a tensile surface load of 100 MPa is applied at the top; consequently, leading to a uniform tensile stress in the sheet of 100 MPa. The Abaqus model, including these boundary conditions, is shown in figure 4.6.

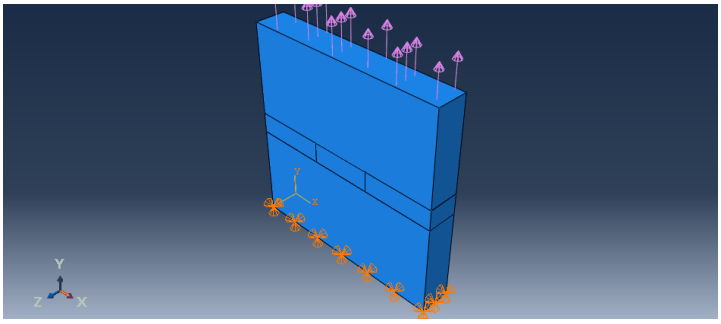


Figure 4.6: Boundary conditions

4.2.2. Meshing

The meshed model is presented in figure 4.7. Globally, the mesh has an approximate size of 10x10x7,5 mm for Δx, Δy and Δz respectively. Mesh refinement was used at the centre of the plate in which the crack is located. The element size varies per calculation which are listed in table 4.2. All elements outside the mesh refinement region are quadratic hexagonal elements of type C3D20R. At the mesh refinement region, linear hexagonal elements of type C3D8R were used as no quadratic elements in the cracked region are supported by Abaqus [1]. Structured meshing technique was used for generating the mesh.

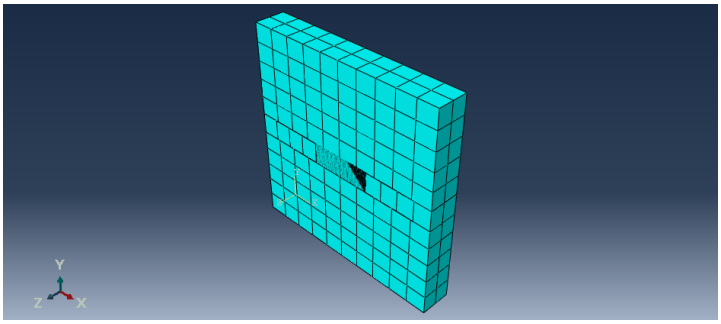
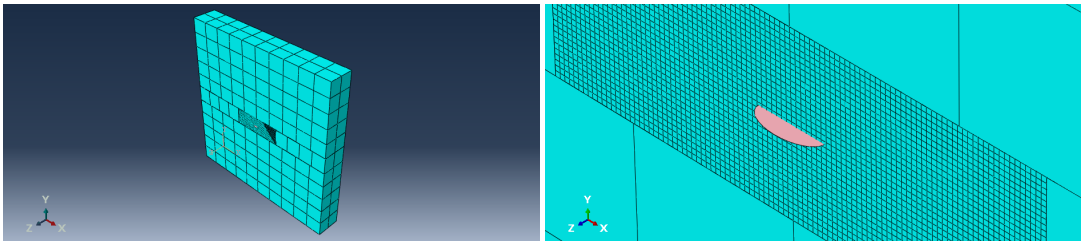


Figure 4.7: Meshed model

Calculation	Δx [mm]	Δy [mm]	Δz [mm]
1	0.4	0.4	0.4
2	0.3	0.3	0.3
3	0.2	0.2	0.2
4	0.1	0.1	0.1

Table 4.2: Mesh sizes around crack part per calculation

As was concluded from chapter 3, no complex meshing is needed for obtaining stable contour integral results. However, a refined mesh was recommended as this leads to higher accuracy. An element size of 0.10 mm was found to give stable results in chapter 3 with an increment size of 3 mm. Therefore, approximately an equal size was investigated for a semi-elliptical crack. The meshed model of the plate including the crack is presented in the figures below.



(a) Global view (b) Zoomed at crack

Figure 4.8: Meshed structure including crack

4.3. Results and comparison

In this section, results from the XFEM analysis are presented. As a 3D crack was investigated, results along the crack front are shown. For defining a position along the crack front, parameter θ was introduced at the beginning op this chapter (see figure 4.2).

4.3.1. Stress intensity factors

In the following graphs, stress intensity factors are presented versus the contour integral number for various positions along the crack front. The results are plotted for all calculations stated in table 4.2.

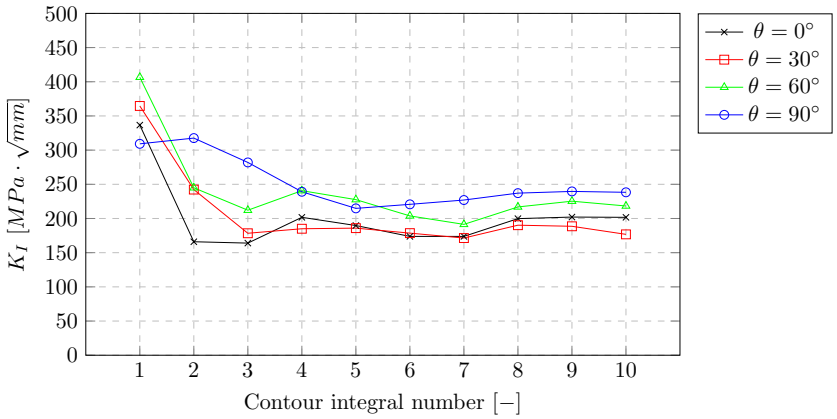


Figure 4.9: K_I for calculation 1 (0,40 mm)

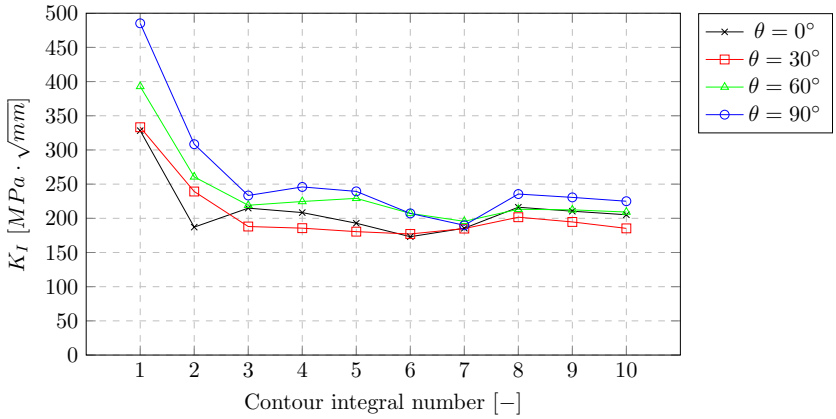


Figure 4.10: K_I for calculation 2 (0,30 mm)

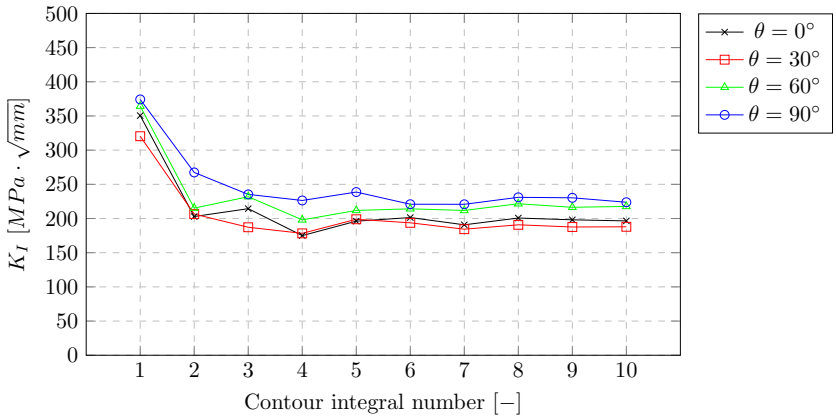


Figure 4.11: K_I for calculation 3 (0,20 mm)

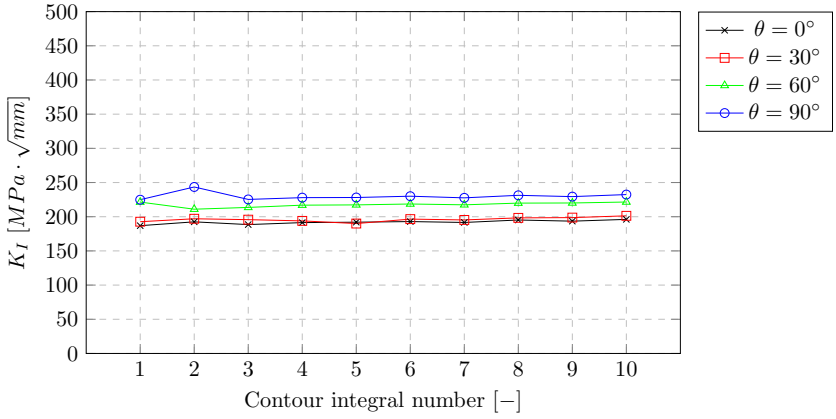


Figure 4.12: K_I for calculation 4 (0,10 mm)

In the graphs presented in figure 4.9-4.12, the values for K_I are shown for each contour integral number. The stability of the results increased as the element size decreased. Therefore, calculation 4 (element size of 0.10 mm) revealed the highest stability (see figure 4.12). Furthermore, calculation 3 (element size of 0.20 mm) showed stable results as well (see figure 4.11). For these two calculations, the values of K_I along the crack front are presented in figure 4.13. Similar as in chapter 3, the first two contour integrals are disregarded when determining the mean average. The results for calculation 3 showed oscillating behaviour along the crack front while this was not observed for calculation 4.

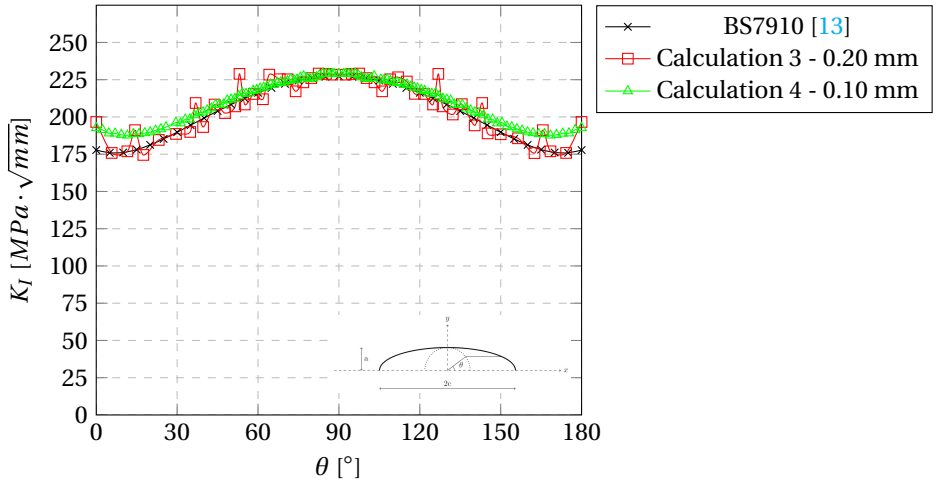


Figure 4.13: Mean average K_I values along crack front

The exact results are presented in the tables below. Here, the differences between the results from Abaqus and BS7910 are calculated in percentages. The largest and smallest difference between Abaqus analysis and BS7910 results are presented boldfaced.

θ [°]	$\mathbf{K_{I,BS7910}}$ [MPa$\sqrt{\text{mm}}$]	$\mathbf{K_{I,Abaqus}}$ [MPa$\sqrt{\text{mm}}$]	**Difference** [%]
0	177.76	197.32	**9.91**
5.74	175.81	175.76	0.03
11.54	176.55	178.72	1.22
14.39	177.72	192.11	7.49
17.46	179.46	176.14	1.89
23.20	183.71	186.38	1.43
29.60	189.36	190.59	0.64
34.85	194.33	191.80	1.32
36.87	196.26	210.07	6.57
39.64	198.90	195.21	1.89
43.91	202.90	209.73	3.26
47.77	206.39	204.03	1.16
51.54	209.63	209.81	**0.08**
53.13	210.95	230.99	8.68
55.08	212.51	210.92	0.75
58.43	215.05	215.40	0.16
61.82	217.41	214.99	1.13
64.16	218.92	231.47	5.42
68.05	221.17	227.91	2.96
71.00	222.66	225.64	1.32
73.95	223.96	218.71	2.40
76.78	225.01	225.64	0.32
79.74	225.89	229.41	1.53
82.52	226.53	232.67	2.57
85.20	226.95	232.93	2.57
87.44	227.17	232.73	2.39
		Average:	**2.66**

Table 4.3: Comparison of results of calculation 3

Table 4.4: Comparison of results of calculation 4

θ [°]	$K_{I,BS7910}$ [MPa$\sqrt{mm}$]	$K_{I,Abaqus}$ [MPa$\sqrt{mm}$]	**Difference** [%]
0.00	177.76	192.63	7.72
2.35	176.61	191.47	**7.76**
4.67	175.96	188.60	6.70
7.01	175.76	189.03	7.02
9.29	175.98	189.03	6.32
11.54	176.55	187.23	5.71
13.74	177.42	189.28	6.27
15.87	178.51	188.81	5.46
18.00	179.82	188.92	4.82
20.03	181.23	190.27	4.75
22.02	182.75	191.48	4.56
23.99	184.36	192.32	4.14
25.88	185.99	193.26	3.76
27.71	187.62	194.03	3.30
29.54	189.30	196.30	3.57
31.30	190.95	196.46	2.80
33.02	192.58	199.82	3.62
34.72	194.20	198.87	2.35
36.37	195.78	201.77	2.97
37.99	197.33	202.89	2.74
39.61	198.87	204.22	2.62
41.15	200.32	204.57	2.07
42.69	201.77	206.54	2.31
44.23	203.19	209.23	2.89
45.68	204.52	208.46	1.89
47.18	205.87	209.29	1.64
48.59	207.11	212.80	2.67
50.00	208.33	211.73	1.61
51.44	209.56	213.08	1.65
52.84	210.72	214.99	1.99
54.19	211.81	215.90	1.90
55.54	212.87	216.18	1.53
56.88	213.89	217.09	1.47
58.21	214.89	218.39	1.60
59.49	215.80	217.37	0.72
60.81	216.73	220.13	1.55
62.13	217.62	221.31	1.67
63.38	218.43	221.22	1.26
64.69	219.24	223.21	1.78
65.92	219.98	221.27	0.58
67.15	220.68	223.43	1.23
68.43	221.38	224.81	1.53
69.64	222.00	223.67	0.75
70.91	222.62	225.42	1.24
72.08	223.16	227.01	1.70
73.34	223.71	224.86	0.51
74.47	224.17	225.18	0.45
75.70	224.63	226.87	0.99
76.91	225.05	228.34	1.44
78.10	225.43	227.92	1.09
79.27	225.77	227.69	0.84
80.40	226.06	227.10	0.46

Continued on next page

Table 4.4 – *Continued from previous page*

θ [°]	$K_{I,BS7910}$ [MPa$\sqrt{mm}$]	$K_{I,Abaqus}$ [MPa$\sqrt{mm}$]	**Difference** [%]
81.69	226.36	227.22	**0.38**
82.75	226.57	228.77	0.96
83.99	226.78	229.92	1.36
85.20	226.95	230.71	1.63
86.38	227.08	230.73	1.58
87.44	227.17	230.67	1.52
88.19	227.21	230.82	1.57
90.00	227.25	230.61	1.46
		Average:	**2.57**

The obtained results showed similar values to that from BS7910. The results with an element size of 0.1 mm showed a smaller difference with the analytical results than the results with 0.2 mm element size. The greatest error, for the results with element size of 0.1 mm were at the surface. This error decreased as θ approached 90°.Moreover, the stability was greatly improved as was the shape of the K_I curve when applying a smaller element size. Calculation 4 (mesh size of 0.1 mm), granted the most accurate and stable results. Therefore, when considering SIF, mesh size of approximately 0.1 mm or smaller is recommended when examining a surface flaw with parameters a = 1 mm and c = 2 mm (crack depth and half crack length respectively). In calculation 3 (0.20 mm) strong oscillating behaviour was observed. This oscillating behaviour was also observed by other researchers [22, 23, 28, 40, 41]. The largest error was observed at the surface/boundary of the plate (for $\theta = 0°$ *or* $\theta = 180°$). This problem was also observed by [9, 22, 41]. A. O. Ayhan [9] neglected the SIFs near the boundaries when determining the trend polynomial and thus neglecting the effect the free boundary has on the SIF trend.

4.3.2. Crack propagation directions

The crack propagation directions are calculated through the maximum tangential stress (MTS) criterion. This was selected as Abaqus provides a built in tool to determine the crack propagation angle (see also appendix B). In section 4.6, the results are validated with results from Franc3D [14]. As Franc3D also provides a definition for crack propagation direction which differs from the one stated in Abaqus, the results may therefore be influenced. However, as will be presented in section 4.6, the difference remains small. In figures 4.14-4.17, the results per calculation are presented. The crack propagation direction is calculated through [2]:

$$\theta = \arccos\left(\frac{3K_{II}^2 + \sqrt{K_I^4 + 8K_I^2K_{II}^2}}{K_I^2 + 9K_{II}^2}\right) \tag{4.10}$$

where:

$$\theta < 0 \quad \text{if} \quad K_{II} > 0 \qquad \text{and} \qquad \theta > 0 \quad \text{if} \quad K_{II} < 0$$

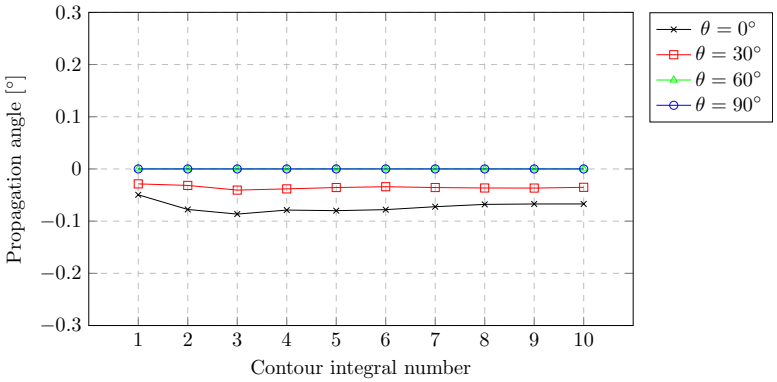


Figure 4.14: Crack propagation angles for calculation 1 (0,40 mm)

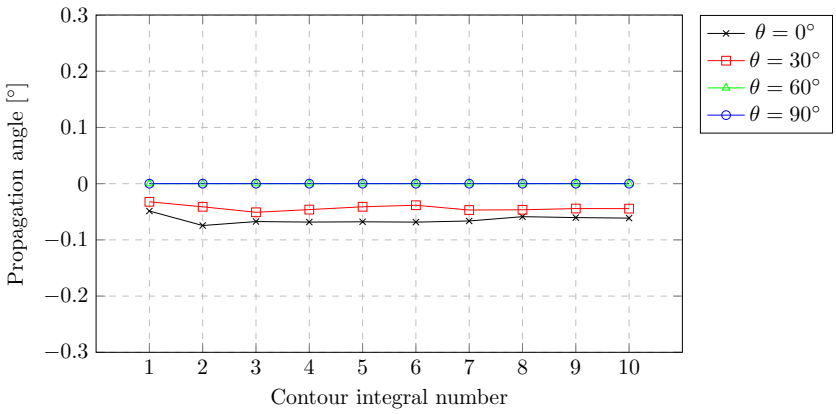


Figure 4.15: Crack propagation angles for calculation 2 (0,30 mm)

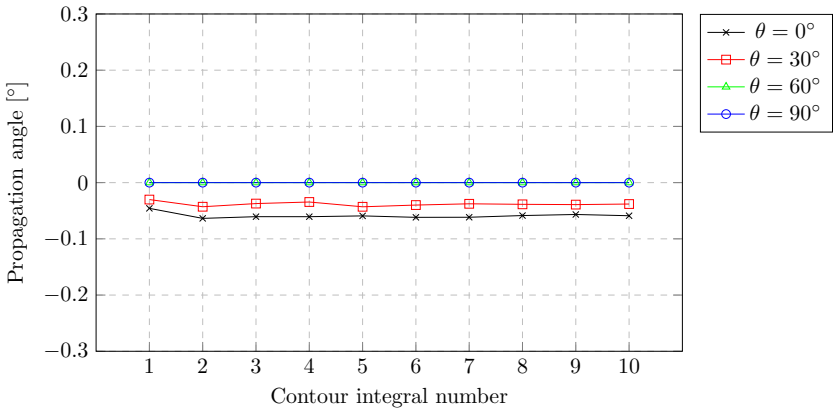


Figure 4.16: Crack propagation angles for calculation 3 (0,20 mm)

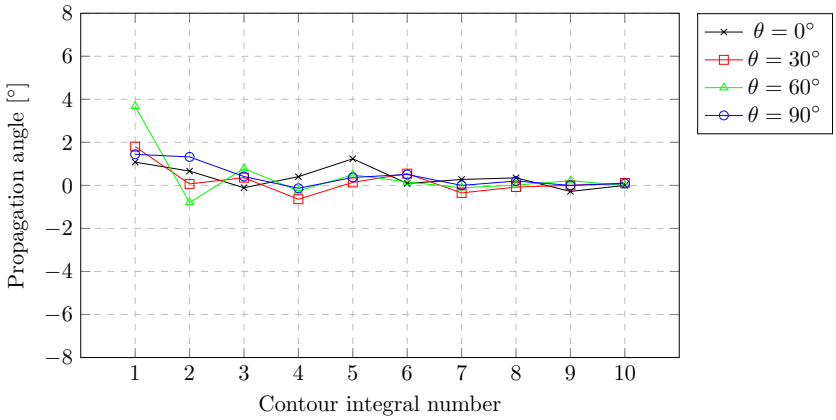
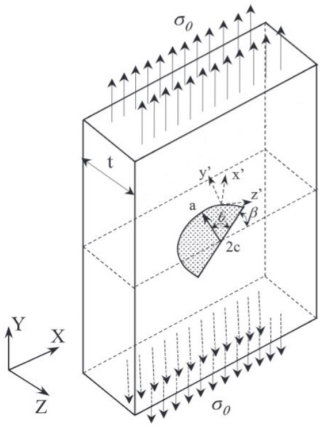


Figure 4.17: Crack propagation angles for calculation 4 (0,10 mm)

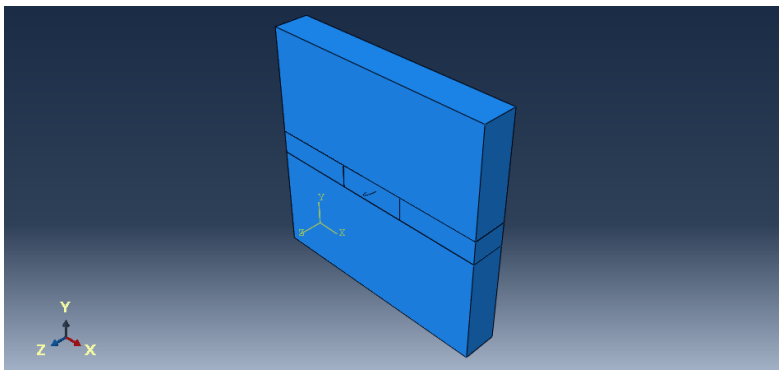
As the crack specimen is loaded in tension and the crack is perpendicular to this loading direction, the expected propagation direction is 0°; provoking mode I cracking. All calculated angles are within a range of '$-1° < angle < 4°$' and thus close to 0. Furthermore, all calculations gave stable results for the crack propagation angle. Calculation 4 (element size of 0.1 mm) gave most stable and most accurate results with respect to SIF and gives also stable and accurate results with respect to crack propagation angles. Therefore, for the next sections, this element size was used.

4.4. Inclined semi-circular crack

Prior to treating a propagating crack, an initially inclined semi-circular crack is investigated. In the previous section was shown that an element size of 0.1 mm leads to both stable and accurate results considering that crack geometry parameters of $a = 2$ and $c = 4$ mm were selected. Therefore, this element size will also be used in this analysis. The structural parameters are presented in the table below (see figure 4.1 for more details). The inclined crack is visualised in figure 4.18.



(a) Parameters visualised [9]



(b) Abaqus model assembled

Figure 4.18: Inclined semi-elliptical centre cracked plate

Symbol	Parameter	Value	Unit
H	Height of plate	100	$[mm]$
W	Width of plate	100	$[mm]$
t	Thickness of plate	15	$[mm]$
E	Young's modulus	210,000	$[MPa]$
μ	Poisson's ratio	0.3	$[-]$
a	Initial crack depth	3	$[mm]$
c	Initial half crack length	3	$[mm]$
σ	Tensile stress	100	$[MPa]$
β	Initial angle	30	$[°]$

Table 4.5: Input parameters

4.4.1. Results

Firstly, the results for K_I, K_{II}, K_{III} and crack propagation direction are presented for each contour integral for several points along the crack front. Subsequently, the mean average is taken and the results are presented along the entire crack front.

Contour integral output versus contour integral number

In figures 4.19-4.22, the results are presented per contour integral number. A total of 10 contours were evaluated in Abaqus. The results for K_I, K_{II}, K_{III} and crack propagation direction are shown in the following graphs.

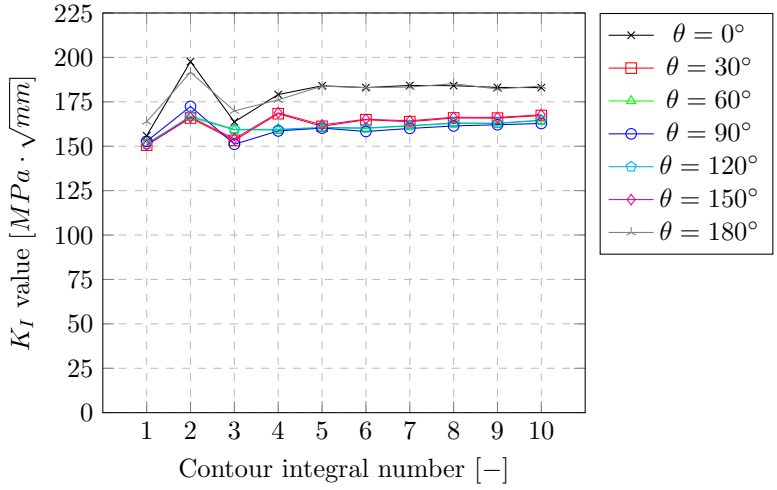


Figure 4.19: K_I for initial crack

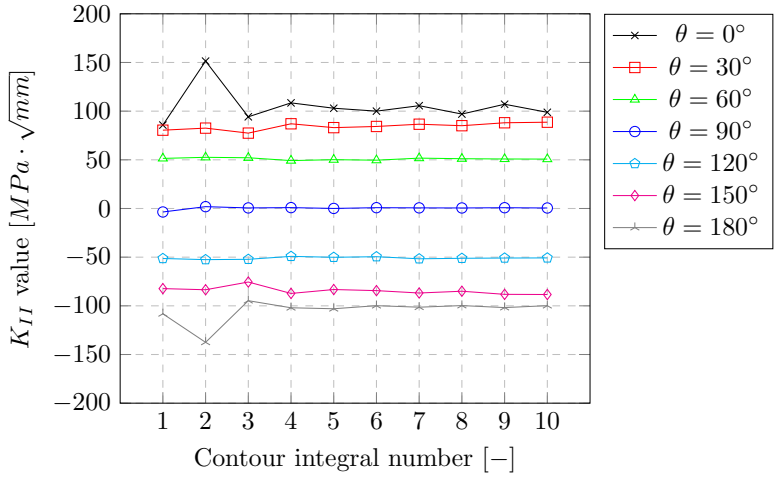


Figure 4.20: K_{II} for initial crack

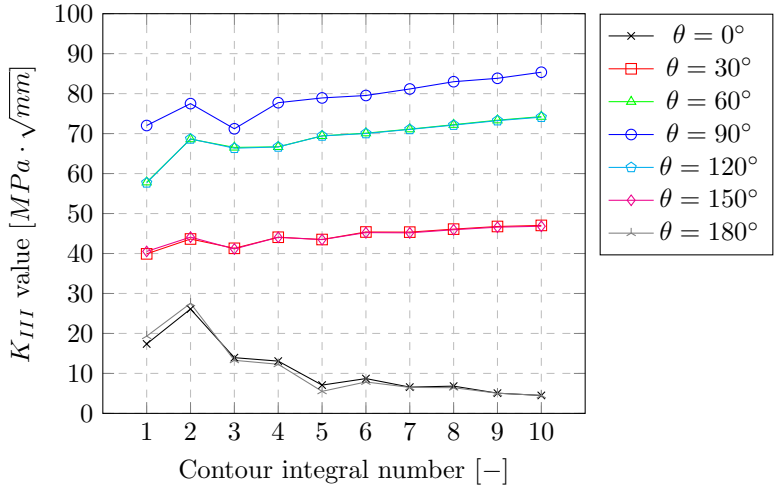


Figure 4.21: K_{III} for initial crack

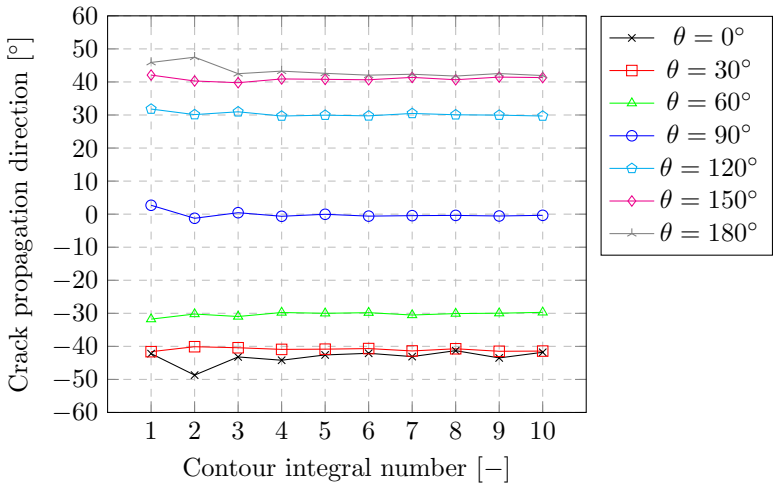


Figure 4.22: Crack propagation angles for initial crack

In the graphs presented in figures 4.19-4.22, the stress intensity factors are presented for each contour integral number. In these results, the first five contour integrals showed in some cases unstable behaviour. After contour integral number five, the remainder is approximately constant. Therefore, the mean average is taken of contour numbers 6, 7, 8, 9 and 10 for further analysis.

Mean averaged parameters along crack front

Below, the results are presented along the entire crack front for each contour integral output parameter (K_I, K_{II}, K_{III} and crack propagation direction). The mean average results show a smooth relation between the stress intensity factors and angle θ. It should be noted that no regression is performed; these results are the raw data output from the Abaqus analysis. In the next section, these results are validated.

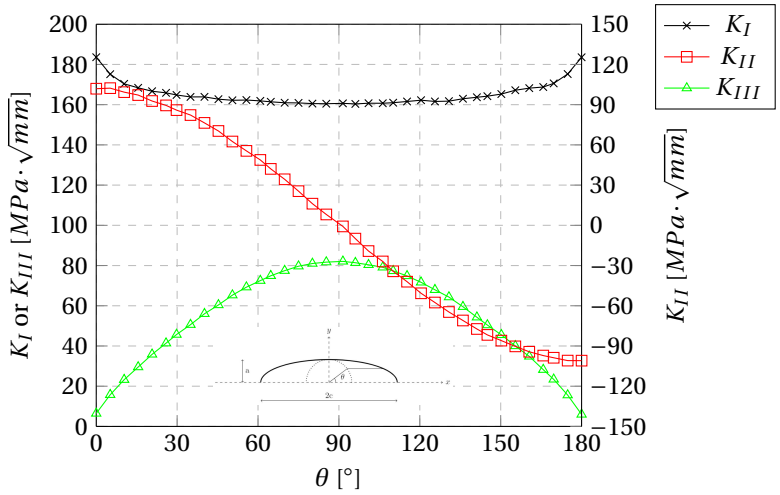


Figure 4.23: Stress intensity factors for initial crack

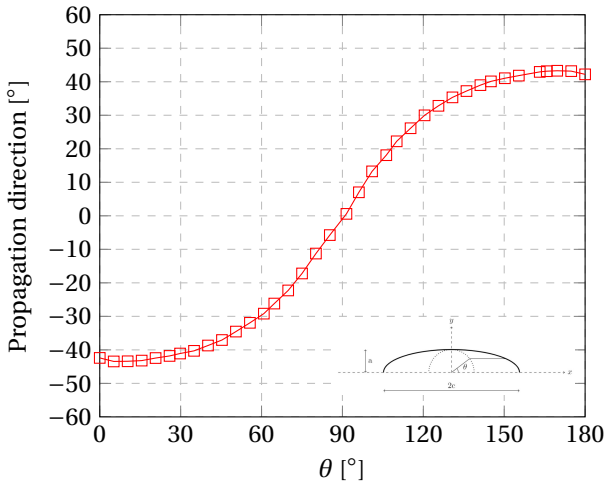
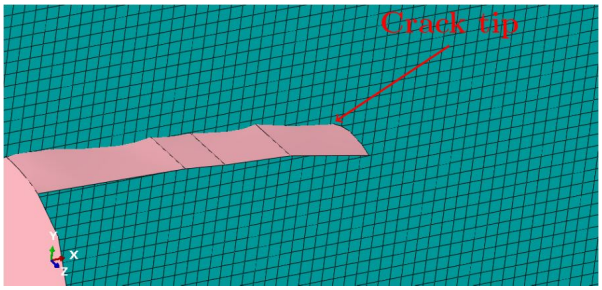


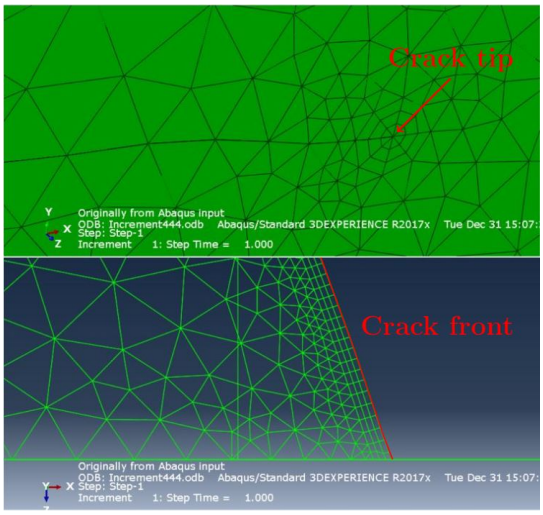
Figure 4.24: Crack propagation directions for initial crack

4.4.2. Validation of results

For validation of the results obtained, two different sources are used. These contain, results obtained from Franc3D and from research papers. Franc3D [14] is a commercial software that analyses fracture and fatigue. An existing Abaqus model can be imported as well as a crack geometry. This makes this program valuable for comparison of results. Franc3D uses conventional FEM including remeshing with aligned mesh around the crack front. Furthermore, polynomials are used for smoothing of the data (SIF's and crack geometry). The difference between mesh used with XFEM in Abaqus and FEM in Franc3D is shown in figure 4.25.



(a) Abaqus XFEM constant mesh



(b) Franc3D FEM mesh alignment

Figure 4.25: Difference in mesh used with Abaqus and Franc3D

Ali O. Ayhan [8] has investigated stress intensity factors for semi-circular cracks. Hereby, a finite element approach was followed to obtain stress intensity factors. Afterwards, a non-linear regression was performed on the data to obtain analytical formulas for stress intensity factors along the entire crack front [9]. The following formulas were found:

$$K_i = K_i^N \cdot K_R \tag{4.11}$$

$$K_R = \sigma_0 \sqrt{\frac{\pi a}{Q}} \tag{4.12}$$

$$Q = 1 + 1.464 \cdot (a/c)^{1.65} \tag{4.13}$$

$$K_I^N = \left(a_1 + b_1(a/t)^2 + c_1(a/t)^4\right) \cdot \sin^2\left(d_1 + e_1\beta\right) \cdot \cosh\left(f_1(\theta - \pi/2)\right) \tag{4.14}$$

$$K_{II}^N = \left(a_2 + b_2(a/t)^2\right) \cdot \sin\left(c_2\beta\right) \cdot \cos\left(d_2\theta\right) \tag{4.15}$$

$$K_{III}^N = \frac{\left(a_3 + b_3(a/t)^2\right) \cdot \sin\left(c_3\beta\right)}{\cosh\left((\theta - \pi/2)d_3\right)} \tag{4.16}$$

where:

σ_0 = Remote tensile stress in MPa;
t = Plate thickness in mm;
a = Crack depth in mm;
c = Half crack length in mm;
θ = Position specifier (see figure 4.2).

Here, a_i, b_i, c_i, d_i, e_i and f_i are parameters depending on the type of crack (surface or corner and deflected or inclined) and the mode of stress intensity factor [9]. The other parameters are visualised in figure 4.26.

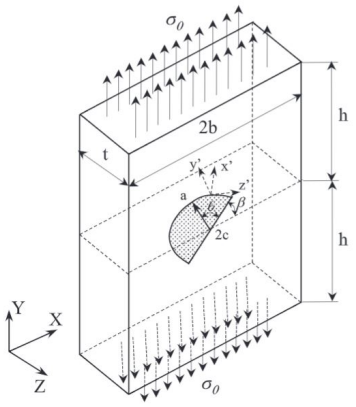


Figure 4.26: Inclined semi-elliptical crack parameters [9]

The results from Franc3D and from application of the formulas presented by Ali O. Ayhan [9] are compared to the results obtained through Abaqus. These are presented in figures 4.27-4.29.

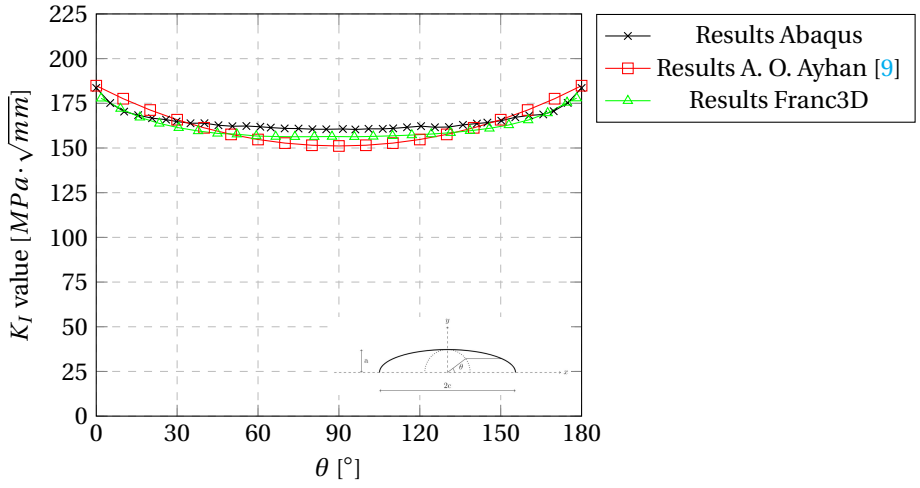


Figure 4.27: Comparison of K_I results

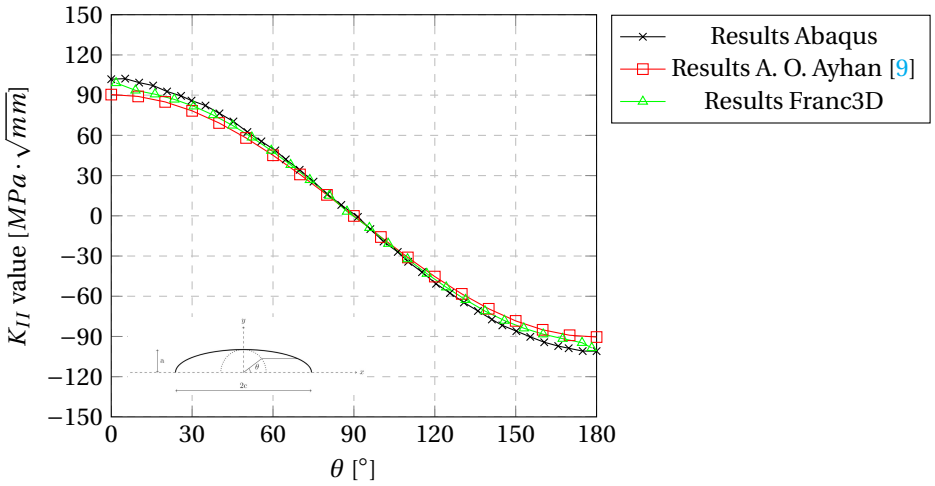


Figure 4.28: Comparison of K_{II} results

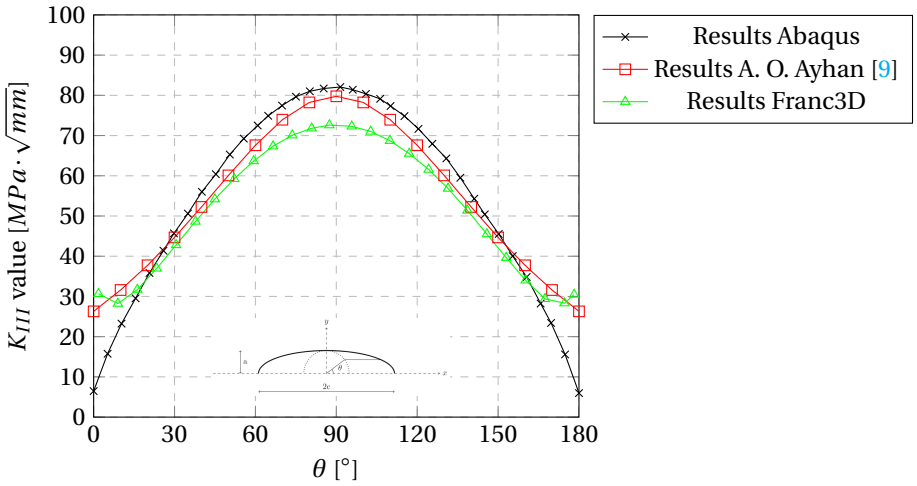


Figure 4.29: Comparison of K_{III} results

The results obtained through XFEM in Abaqus are in good agreement with the results of Franc3D and A.O. Ayhan [9]. For K_{III}, the values at the surface (plain stress region) show lower values. In the next sections, 3D crack propagation is examined.

4.5. 3D crack propagation methodology

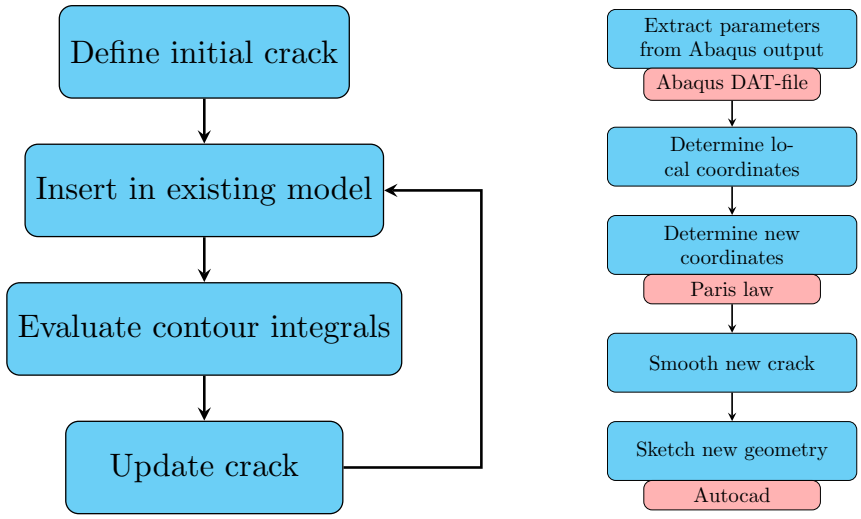
This and the next section, is dedicated to the propagation of 3D cracks. A semi-circular surface crack is assumed as initial crack after which the crack will propagate to certain directions. The initial crack is inclined with an initial angle β making the crack propagation direction a valuable output to investigate (see figure 4.1). In table 4.5, the input parameters are listed.

In this section, the methodology followed to predict crack propagation is presented. A more detailed explanation on the methodology of analysis is presented in appendix D. A semi-elliptical crack can be constructed according to the method presented in section 4.2. After determining the initial crack, this is inserted in an existing model. This model contains the geometry of the structure including boundary conditions and mesh. Therefore, no re-meshing is used in these calculations. When the analysis is completed, the crack can be updated for which, a Python script is used. The flow chard for updating the crack is shown in figure 4.30b. A detailed explanation of the script is presented in appendix D. In short, the script starts with extracting certain parameters from the Abaqus DAT-file. These contain:

- Global coordinates of crack front;

- Stress intensity factors;
- Crack propagation angles.

After reading the DAT-file, the script determines the local coordinates (according to coordinate system as shown in figure 4.2) of the crack front corresponding to the global coordinates. The Paris law is applied to determine the local coordinates of the new crack front; these are used to sketch the new crack geometry in Autocad. From Autocad, the sketched crack is imported into Abaqus for a new analysis. The flow chard for the global analysis and updating the crack is shown in figure 4.30. Determining the new crack front is visualised in figure 4.31.



(a) Global analysis

(b) Updating crack

Figure 4.30: Flow chard of python code analysis

For all data points obtained through Abaqus, the new points are determined using both the Paris law and the crack propagation angle; this is visualised in figure 4.31. The old data points are points along the crack front as inputted into Abaqus while the new data points are obtained through a Python script. Here, Δa is obtained through application of the Paris law for a certain predefined number of cycles as shown in equation 4.17. The crack propagation angle θ is obtained through Abaqus and ϕ is determined by determining the orthogonal direction of the polynomial through the three data points at the location of interest. The exact formulas used for determining ϕ are stated in appendix D.

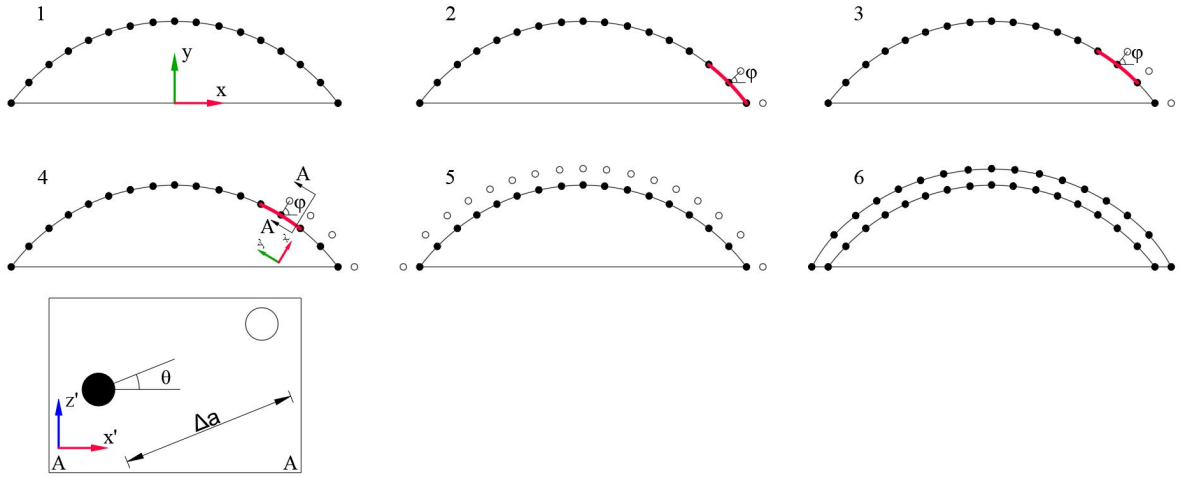


Figure 4.31: Determining new crack front

4.6. Results - 3D crack propagation

The results for crack propagation are presented in this section. The corresponding input parameters are as listed in table 4.5 and figure 4.1. Firstly, the results of the stress intensity factors are presented for each increment followed by the crack trajectory. Subsequently, the relation between the number of cycles versus crack depth increase is discussed.

4.6.1. Stress intensity factors

Below, the stress intensity factors for all three modes are plotted for each crack increment. These SIF's are extracted from the DAT-file from Abaqus; no regression is performed on these data. In the subfigures on the right hand side of the graphs, the top views are shown of the crack. Here, the corresponding increment is indicated.

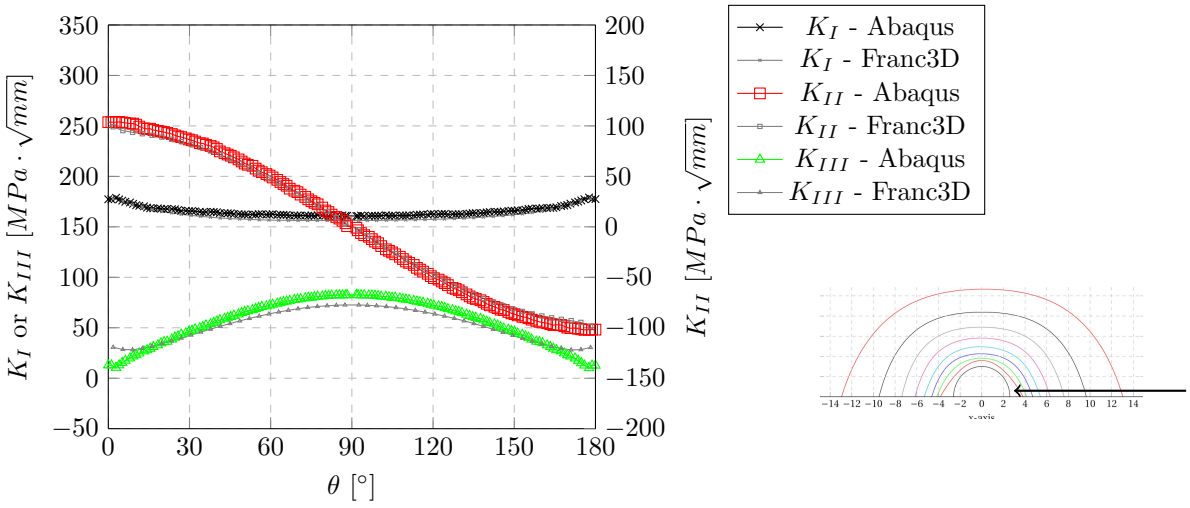


Figure 4.32: Results initial crack

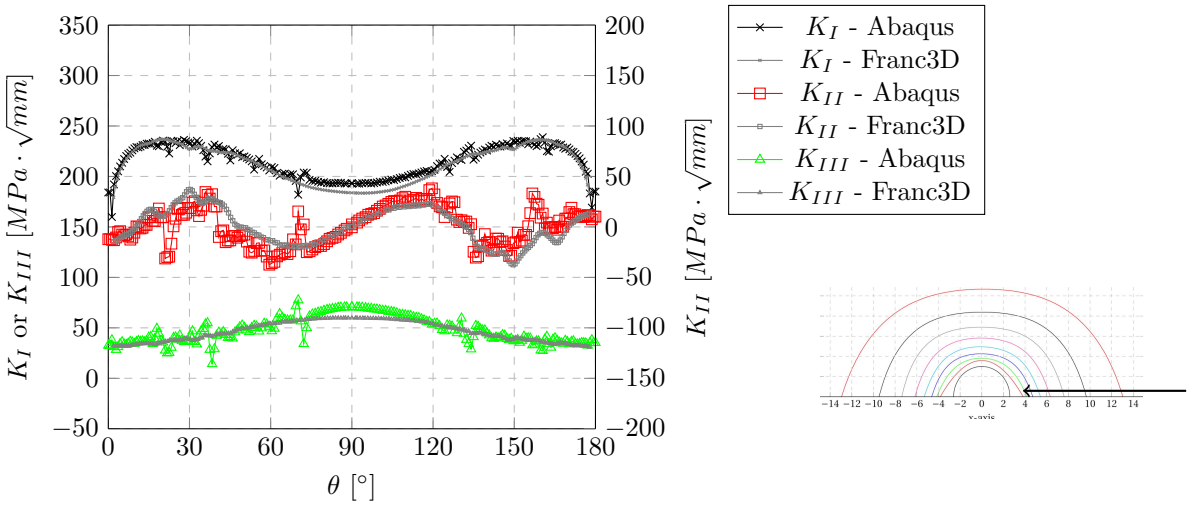


Figure 4.33: Results increment 1

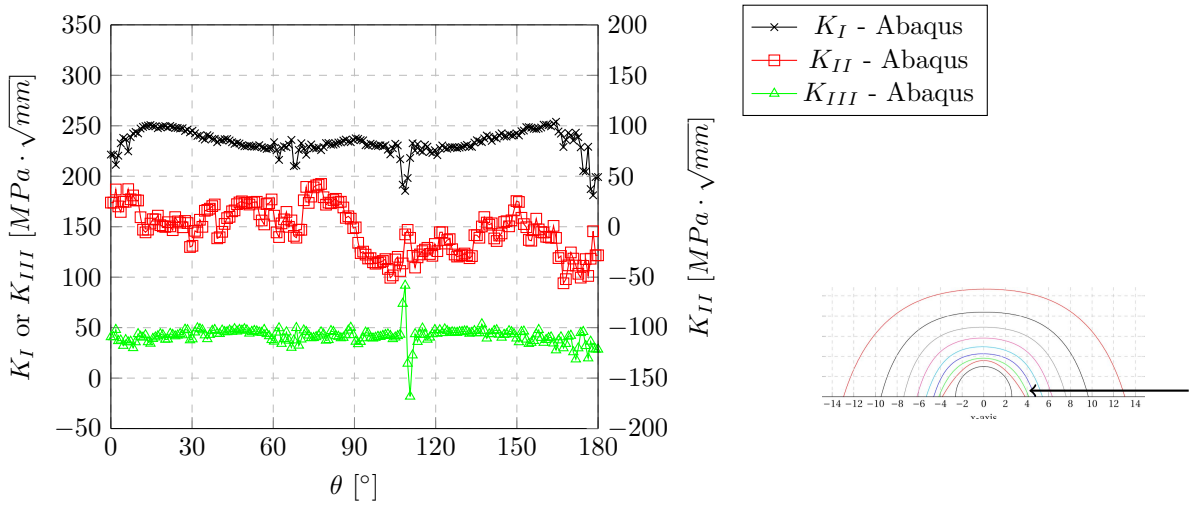


Figure 4.34: Results increment 2

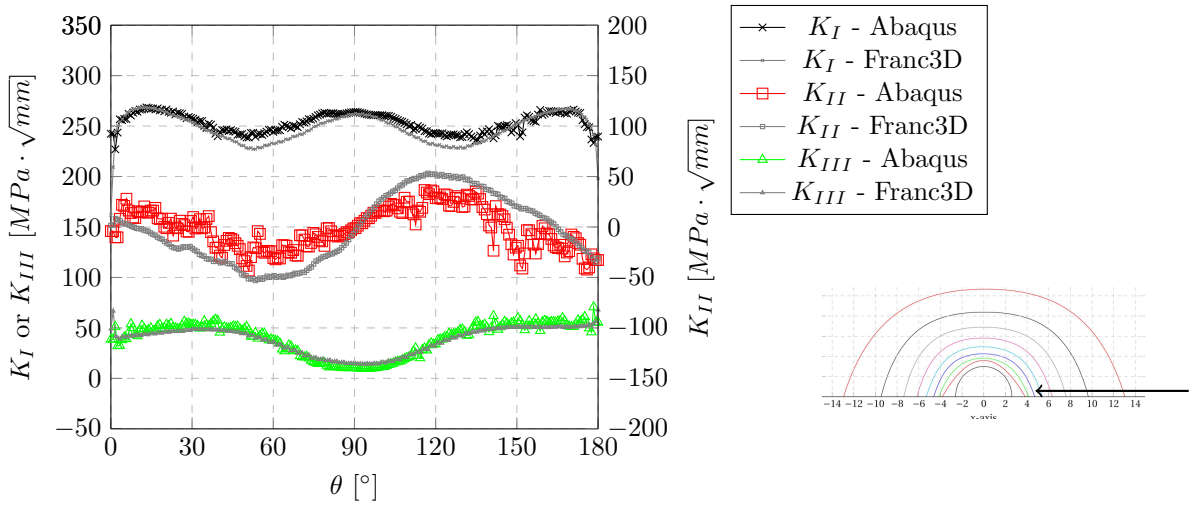


Figure 4.35: Results increment 3

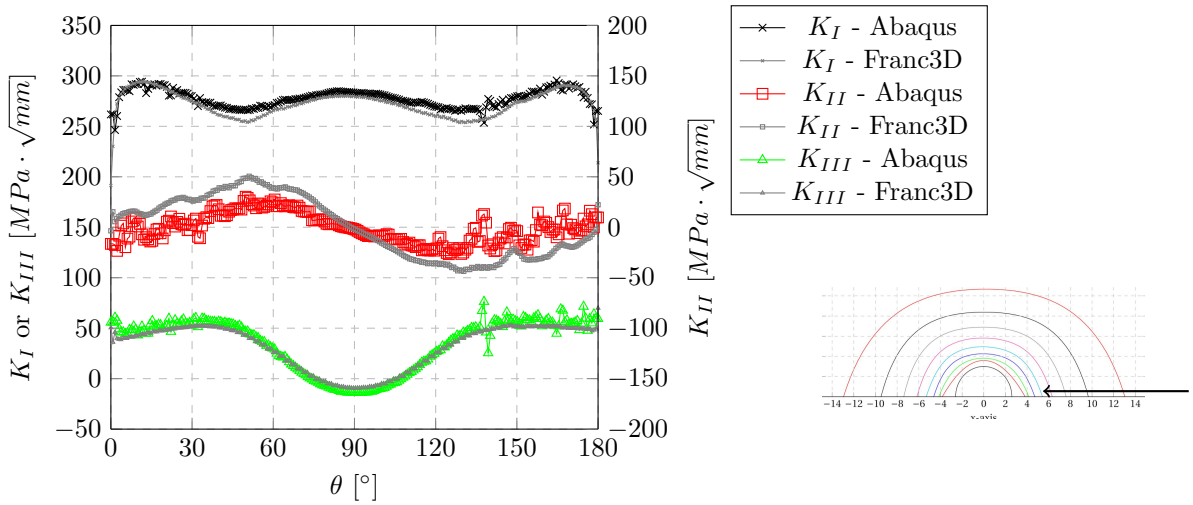


Figure 4.36: Results increment 4

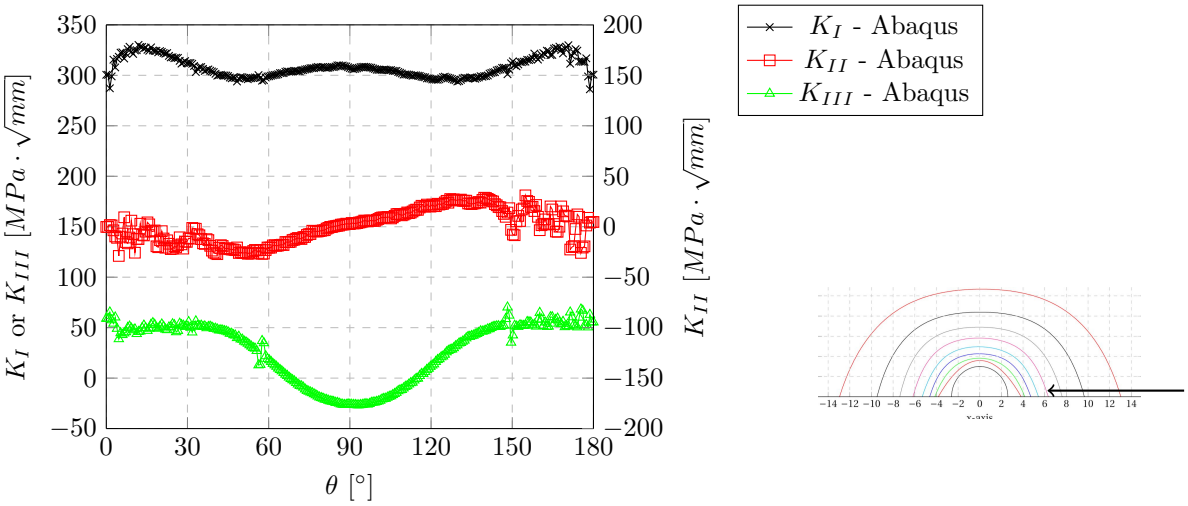


Figure 4.37: Results increment 5

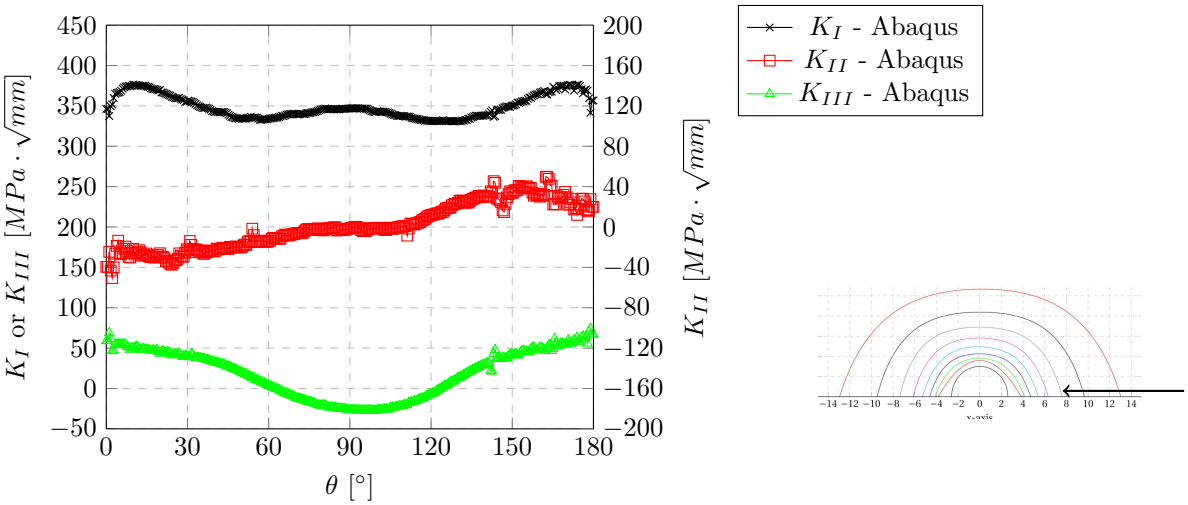


Figure 4.38: Results increment 6

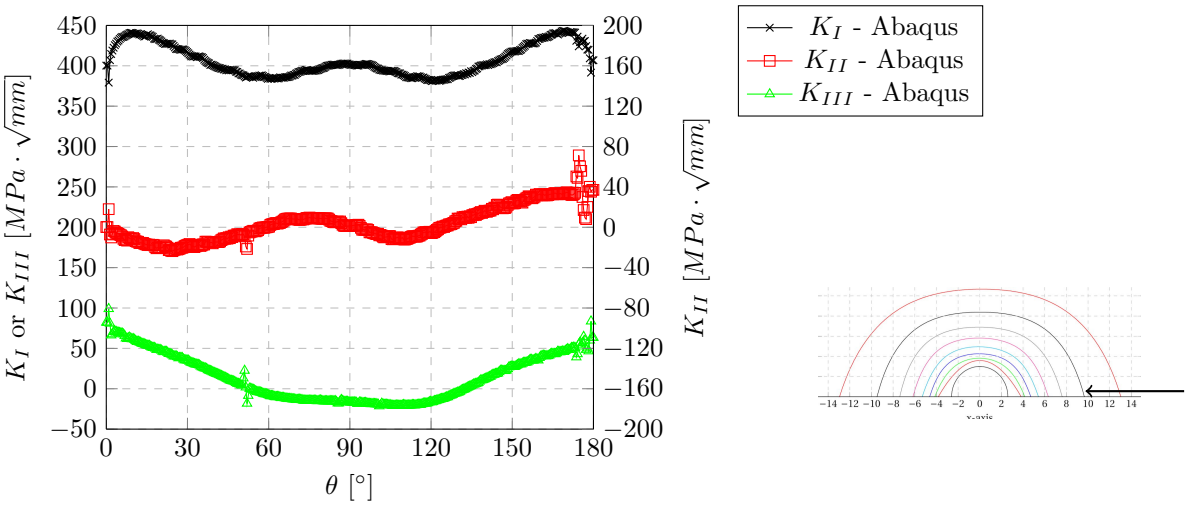


Figure 4.39: Results increment 7

The results of stress intensity factors are stable; some oscillation occurs but mostly for K_{II}. However, it should be noted that K_{II} is plotted on a smaller scale and the oscillations are therefore illustrated larger compared to K_I and K_{III}. Furthermore, the oscillating behaviour of the SIF results reduces as the crack grows in size. For the last 2 increments (6 and 7), almost no oscillation is present. Moreover, K_I shows an increasing trend and is largest of all three modes throughout the entire analysis. Therefore, the crack propagates to a more dominant mode I. As a result, the crack converges to a perpendicular configuration with respect to the applied tensile load as is shown in the next section.

4.6.2. Crack trajectory

With a methodology followed as explained in section 4.5, the geometry of the propagating crack is determined. Below, all the crack increments are illustrated. Each increment is the result of 200,000 cycles integrated in the Paris law; only the first increment is a result of 500,000 cycles. The initial crack is assumed to exist prior to analysing the structure.

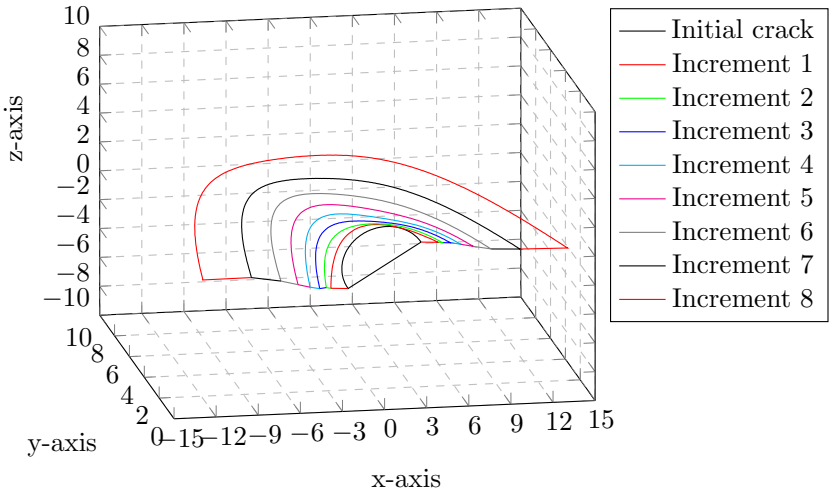


Figure 4.40: Isometric view of crack trajectories

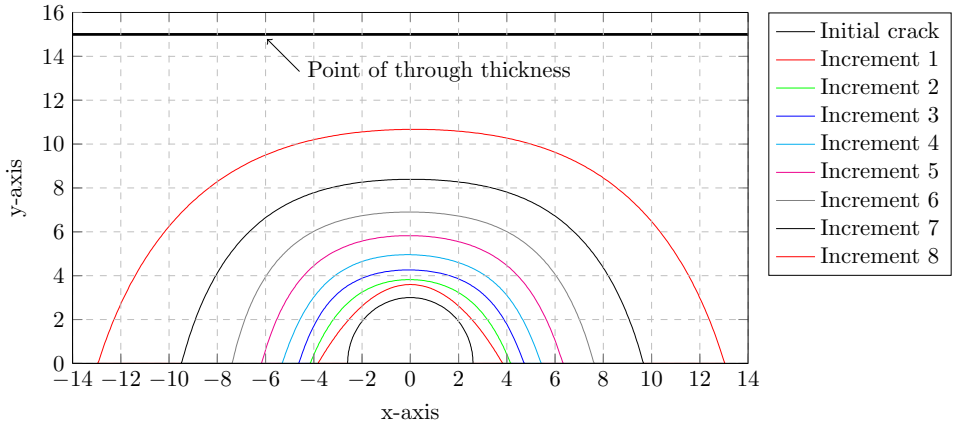


Figure 4.41: Top view of crack trajectories

The propagating crack shows a steady semi-circular crack front as presented in figure 4.41; this was also observed by [40]. Only the first increment shows a more parabola shape; this is due to the fact that a second order polynomial was used for this increment for regression. Afterwards, a 5th order polynomial was used to catch the semi-circular shape the crack is shown to follow. The crack increment size is increasing with the number of cycles; this is a result from the increase of the effective stress intensity factor K_{eff}. It should be noted that the first increment was a result from 500,000 cycles while the other increments from 200,000 cycles. Therefore, the increase in crack for the first increment is larger than for example increment 2. In figure 4.40, the propagating crack is illustrated in 3D. The crack is converging to a state perpendicular to the applied

tensile load which works in the z-direction. This is in agreement with the assumed maximum tangential stress criterion (MTS) used to determine the propagation direction.

For validation of the resulting crack trajectory, a calculation with Franc3D is performed. The crack trajectory at the surface of the plate is presented in figure 4.42. The x-axis is in the width direction and the z-axis in the height direction of the plate. All increments from Franc3D are a results of 200.000 cycles while the first increment from the script with Abaqus was a results of 500,000 cycles and the remainder of 200,000 cycles.

In the first stage of the crack, for approximately $-6 < x < 6$, good agreement was found between the results obtained through XFEM in Abaqus and Franc3D. Afterwards, small differences of approximately 1 mm were observed. Both XFEM and Franc3D results show a propagating crack which is perpendicular to the loading direction (x-direction).

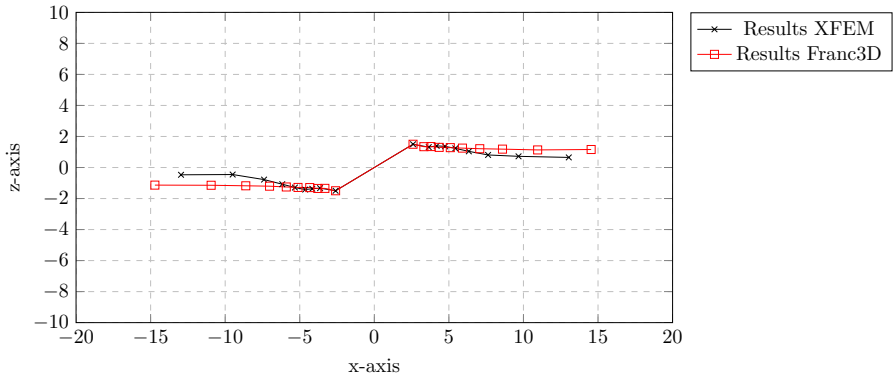
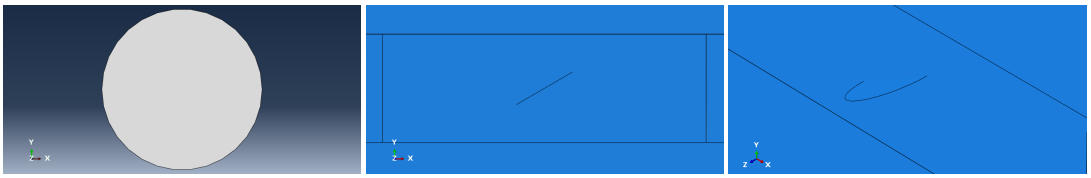


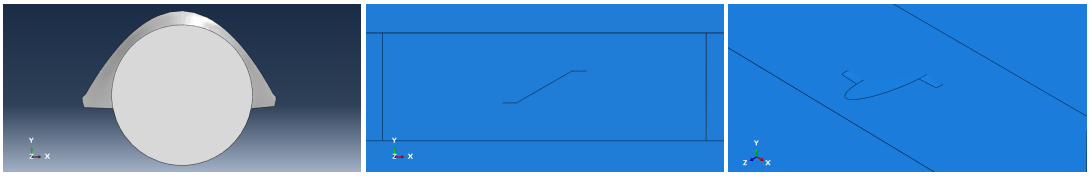
Figure 4.42: Comparison of crack propagation with Franc3D results

In the figures presented below, all crack increments are visualised from the Abaqus modelling.



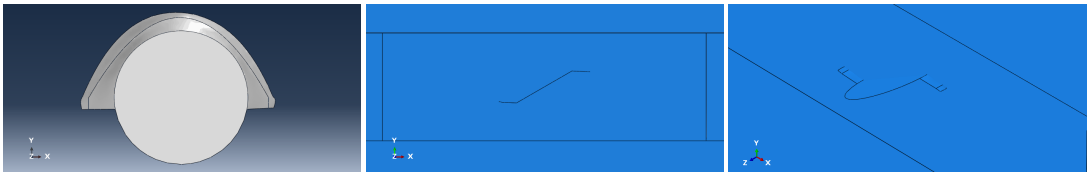
(a) Abaqus crack part (b) Abaqus assembled XY (c) Abaqus assembled ISO

Figure 4.43: Initial crack



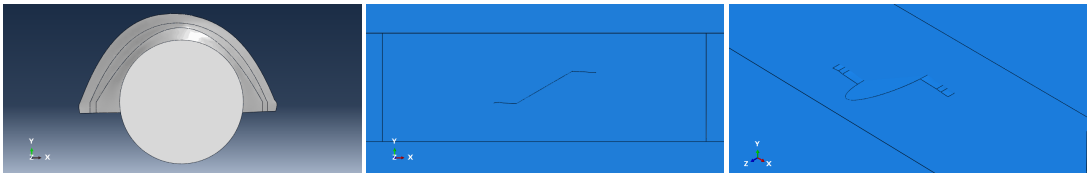
(a) Abaqus crack part (b) Abaqus assembled XY (c) Abaqus assembled ISO

Figure 4.44: Increment 1



(a) Abaqus crack part (b) Abaqus assembled XY (c) Abaqus assembled ISO

Figure 4.45: Increment 2

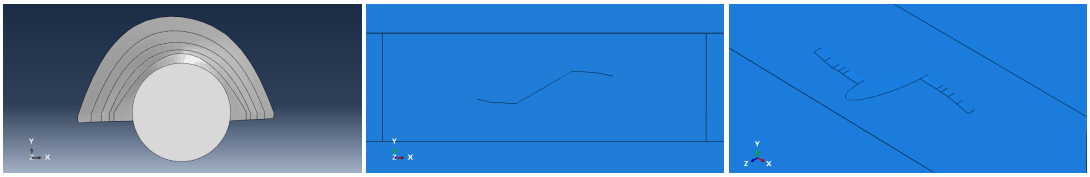


(a) Abaqus crack part (b) Abaqus assembled XY (c) Abaqus assembled ISO

Figure 4.46: Increment 3

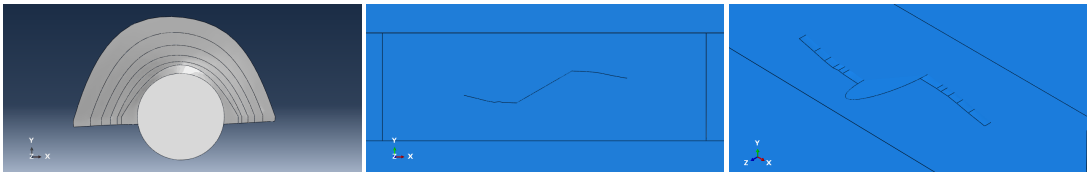
(a) Abaqus crack part (b) Abaqus assembled XY (c) Abaqus assembled ISO

Figure 4.47: Increment 4



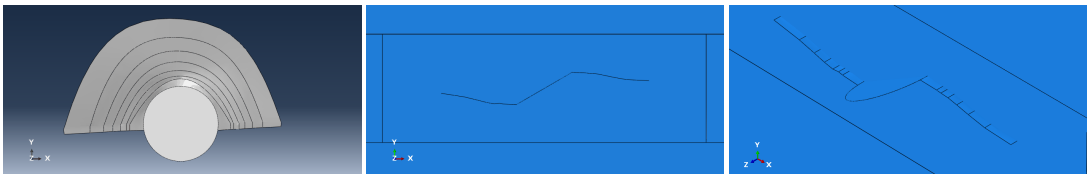
(a) Abaqus crack part (b) Abaqus assembled XY (c) Abaqus assembled ISO

Figure 4.48: Increment 5



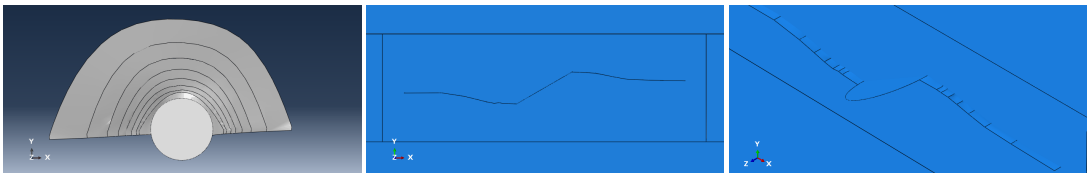
(a) Abaqus crack part (b) Abaqus assembled XY (c) Abaqus assembled ISO

Figure 4.49: Increment 6



(a) Abaqus crack part (b) Abaqus assembled XY (c) Abaqus assembled ISO

Figure 4.50: Increment 7



(a) Abaqus crack part (b) Abaqus assembled XY (c) Abaqus assembled ISO

Figure 4.51: Increment 8

Some of the coordinates along the crack front, that were used to draw the crack in Autocad, are presented in the table below. The local coordinate system that has been used is shown in figure 4.2. Only 15 coordinates are presented per increment of the analysis. In the actual calculations, much more (100-400) had been used to obtain a more smooth sketch of the crack front.

Table 4.6: Coordinates along crack front

Initial crack			**Increment 1**			**Increment 2**		
X	Y	Z	X	Y	Z	X	Y	Z
3.00	0.00	0.00	3.73	0.00	-0.66	4.19	0.00	-0.92
2.92	0.67	0.00	3.20	1.33	-0.50	3.59	1.41	-0.82
2.70	1.30	0.00	2.66	2.02	-0.42	2.98	2.29	-0.73
2.35	1.87	0.00	2.13	2.59	-0.33	2.37	2.91	-0.62
1.87	2.35	0.00	1.60	3.03	-0.25	1.77	3.34	-0.48
1.30	2.70	0.00	1.07	3.34	-0.17	1.16	3.61	-0.33
0.67	2.92	0.00	0.54	3.53	-0.08	0.56	3.77	-0.16
0.00	3.00	0.00	0.00	3.60	0.00	-0.05	3.82	0.02
-0.67	2.92	0.00	-0.53	3.53	0.08	-0.65	3.77	0.19
-1.30	2.70	0.00	-1.06	3.35	0.16	-1.26	3.60	0.36
-1.87	2.35	0.00	-1.59	3.03	0.25	-1.87	3.30	0.50
-2.35	1.87	0.00	-2.12	2.60	0.33	-2.47	2.83	0.63
-2.70	1.30	0.00	-2.66	2.03	0.41	-3.08	2.16	0.74
-2.92	0.67	0.00	-3.19	1.34	0.50	-3.68	1.22	0.84
-3.00	0.00	0.00	-3.72	0.00	0.65	-4.29	0.00	0.95

Increment 3			**Increment 4**			**Increment 5**		
X	Y	Z	X	Y	Z	X	Y	Z
4.67	0.00	-1.19	5.25	0.00	-1.67	5.81	0.00	-2.35
4.01	1.59	-1.23	4.51	1.78	-1.68	4.97	2.36	-2.23
3.34	2.59	-1.16	3.77	2.95	-1.58	4.14	3.68	-2.00
2.67	3.28	-1.01	3.03	3.75	-1.39	3.30	4.59	-1.70
2.01	3.74	-0.80	2.29	4.30	-1.13	2.46	5.18	-1.33
1.34	4.03	-0.56	1.55	4.65	-0.81	1.62	5.56	-0.93
0.67	4.20	-0.30	0.81	4.86	-0.45	0.78	5.76	-0.50
0.01	4.26	-0.02	0.07	4.95	-0.07	-0.05	5.83	-0.04
-0.66	4.22	0.26	-0.67	4.92	0.32	-0.89	5.76	0.43
-1.33	4.07	0.54	-1.41	4.75	0.70	-1.73	5.54	0.89
-1.99	3.78	0.79	-2.15	4.41	1.05	-2.57	5.13	1.34
-2.66	3.31	1.00	-2.89	3.86	1.34	-3.41	4.46	1.74
-3.33	2.59	1.15	-3.63	3.04	1.55	-4.24	3.44	2.06
-3.99	1.56	1.21	-4.37	1.87	1.65	-5.08	1.96	2.24
-4.66	0.00	1.14	-5.11	0.00	1.61	-5.92	0.00	2.22

Increment 6			**Increment 7**			**Increment 8**		
X	Y	Z	X	Y	Z	X	Y	Z
7.08	0.00	-3.36	8.47	0.00	-3.72	10.69	0.00	-5.37
6.10	2.34	-3.01	7.36	3.48	-3.75	9.18	5.52	-4.43
5.12	4.15	-2.61	6.25	5.39	-3.13	7.67	7.60	-3.53
4.14	5.35	-2.17	5.14	6.65	-2.59	6.16	8.98	-2.88
3.16	6.11	-1.70	4.03	7.46	-2.08	4.65	9.85	-2.29
2.18	6.57	-1.22	2.92	7.96	-1.56	3.14	10.35	-1.65
1.21	6.81	-0.73	1.81	8.24	-1.02	1.63	10.61	-0.94
0.23	6.90	-0.22	0.70	8.39	-0.45	0.12	10.69	-0.19
-0.75	6.88	0.31	-0.41	8.41	0.13	-1.39	10.62	0.58
-1.73	6.71	0.84	-1.52	8.32	0.71	-2.90	10.36	1.31

Continued on next page

Table 4.6 – *Continued from previous page*

X	Y	Z	X	Y	Z	X	Y	Z
-2.71	6.36	1.38	-2.63	8.07	1.28	-4.41	9.87	1.99
-3.69	5.72	1.92	-3.74	7.58	1.83	-5.92	9.03	2.64
-4.67	4.66	2.43	-4.85	6.76	2.40	-7.43	7.68	3.37
-5.65	2.99	2.88	-5.96	5.47	3.00	-8.94	5.61	4.34
-6.63	0.00	3.24	-7.07	0.00	3.78	-10.46	0.00	5.67

4.6.3. Fatigue life assessment

For assessing the fatigue life, the Paris law was used. The obtained results are presented in this section. The equation for both the Paris law as the effective stress intensity factor ΔK_{eff}, is as follows:

$$\Delta a = \Delta n \cdot C \cdot \Delta K_{eff}^{m} \tag{4.17}$$

$$\Delta K_{eff} = \sqrt[4]{(\Delta K_I)^4 + 8(\Delta K_{II})^4} \tag{4.18}$$

where:

Δa = Crack extension in mm;
Δn = Number of cycles responsible for Δa;
ΔK_I = Stress intensity factor range corresponding to mode I in $MPa \cdot \sqrt{mm}$;
ΔK_{II} = Stress intensity factor range corresponding to mode II in $MPa \cdot \sqrt{mm}$.

The parameters C and m have been chosen as $3.98 \cdot 10^{-13}$ and 2.88 respectively. These values have been recommended for surface flaws by BS7910 [13] and were also selected in chapter 3. After the initial crack, 500,000 cycles have been applied to determine the first increment. Afterwards, a fixed number of 200,000 cycles have been applied. The reason to choose lower amount of cycles is to maintain small increment sizes in order to obtain more accurate results. It should be noted that the values presented here only correspond to a crack in the Paris regime (see section 2.1.2). The cumulative number of cycles is presented below versus the crack depth.

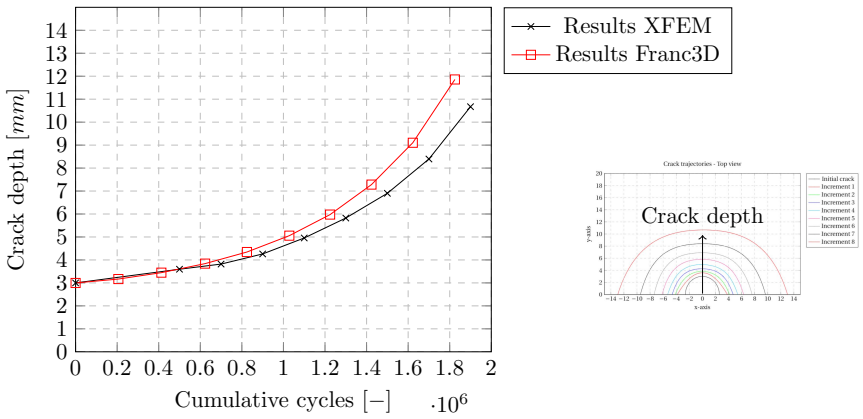


Figure 4.52: Cumulative cycles versus crack depth

Between the results through Abaqus and the results trough Franc3D, are some differences in analysis. Franc3D uses conventional FEM including mesh alignment with the crack front while results through Abaqus with XFEM uses a constant mesh (for mesh comparison see figure 4.25). Nonetheless, the results are close and therefore a good agreement was found in fatigue life assessment. In table 4.7 all data are shown from the fatigue analysis. In crack geometry, a 5th order polynomial is used to fit the data of the crack geometry. Therefore, the Δa values that are shown in this table may differ slightly from pure application of equation 4.17. The results are presented for both Franc3D (F) as for the script (S) composed in Python. Using linear interpolation, the results can be compared for an equal value of cumulative cycles. For example, the difference at 800,000 cycles is 7.7% and 24.4% at 1,800,000 cycles. For example, the difference at 800,000 is determined as follows:

$$3.82 + 0.5(4.26 - 3.82) = 4.04 \text{ mm}$$

$$(4.35 - 4.04)/4.04 \cdot 100\% = 7.7\%$$

The fatigue life assessment is very sensitive to the stress intensity factors. Therefore, a small difference in K_{eff} may lead to significant difference in fatigue life calculations.

Step	**Δn [Cycles]**		**ΔK_{eff} [MPa$\cdot\sqrt{mm}$]**		**Δa [mm]**		**a [mm]**		**N [Cycles]**	
	F	S	F	S	F	S	F	S	F	S
1	200,000	500,000	156.42	160.70	0.168	0.596	3.17	3.60	200,000	500,000
2	200,000	200,000	184.96	193.20	0.278	0.224	3.44	3.82	400,000	700,000
3	200,000	200,000	207.41	236.18	0.394,	0.441	3.84	4.26	600,000	900,000
4	200,000	200,000	229.66	264.08	0.505	0.695	4.35	4.96	800,000	1,100,000
5	200,000	200,000	257.46	283.08	0.712	0.868	5.06	5.82	1,000,000	1,300,000
6	200,000	200,000	285.13	307.26	0.919	1.077	5.98	6.90	1,200,000	1,500,000
7	200,000	200,000	320.64	346.96	1.305	1.492	7.28	8.39	1,400,000	1,700,000
8	200,000	200,000	360.11	401.14	1.822	2.278	9.10	10.67	1,600,000	1,900,000
9	200,000	-	412.96	-	2.756	-	11.86	-	1,800,000	-

Table 4.7: Data fatigue analysis

In figure 4.53 a graph is shown of the crack propagation rate $\frac{da}{dn}$ versus the effective stress intensity factor ΔK_{eff} on log-log scale. While the Paris law was used to determine the new crack fronts, a 5th order polynomial was used to fit the newly determined crack front. Therefore, the trend is plotted to show that the crack behaves as in the Paris region while using a polynomial regression function. Minor deviations are found between XFEM results and those from Franc3D; these may have occurred due to the polynomial fit.

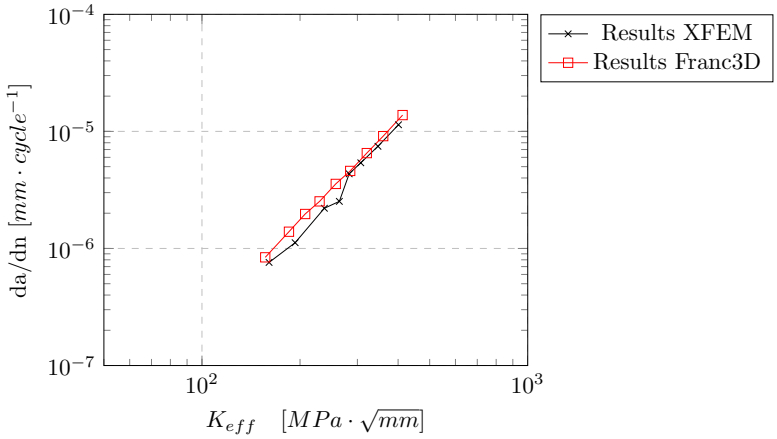


Figure 4.53: Crack propagation rate

4.7. Conclusion

In this chapter, a semi-elliptical centre cracked plate was examined. The extended finite element method in Abaqus was used for contour integral evaluation. A Python code was constructed in order to read the Abaqus output and to model crack propagation. As the analysis proved to be complex, two different studies were performed; these are treated separately in this conclusion as well.

Semi-elliptical stationary crack
Firstly, a semi-elliptical crack was examined with crack depth of 2 mm and half crack length of 4 mm. Here, K_I and crack propagation angles are results that were obtained along the crack front. The results were compared with the analytical solutions from BS7910 [13]. Four calculations were carried out with element sizes of 0.4 mm, 0.3 mm, 0.2 mm and 0.1 mm. The following conclusions are drawn.

- In total, 10 contour integrals were evaluated along the crack front for each calculation. From the resulting K_I values, element sizes of 0.2 and 0.1 showed most stable results. The finer mesh was used, the more stable the results were with respect to K_I
- The stress intensity factor in mode I, K_I, shows good agreement between the results from XFEM and BS7910 [13]. When the element sizes are smaller or equal as 0.2 mm, the differences are within 10% with an average difference around 2.6%
- When considering the crack propagation angle, all calculations were stable and show a good accuracy. The calculated propagation angles were approximately between -1 and 4 degrees.

Inclined semi-circular propagating crack
Secondly, an inclined semi-circular crack was investigated in a centre cracked plate. The surface flaw, with parameters $a = c = 3$ mm, was inclined with an angle of 30°. The results were validated with Franc3D and with a research performed by A. O. Ayhan [8, 9]. 3D crack propagation was investigated using a python script. This script reads the DAT-file from Abaqus and determines the new crack front. Paris law was integrated in the script for fatigue analysis. The following conclusions are drawn.

- For both the initial crack as for the increments, the stress intensity factors from XFEM complied with the results of Franc3D and A.O. Ayhan.
- K_I was increasing with the number of increments while K_{II} and K_{III} were decreasing representing a dominant mode I cracking behaviour.
- Oscillating behaviour was observed for stress intensity factor distributions along the crack front; these oscillations became smaller as the crack grew in size. The crack fronts are smoothed by a 5th order polynomial in order to minimise the effect of the oscillation.
- The crack trajectory complied with that from Franc3D. The crack converged to a state perpendicular to that of the applied tensile load as a result from the MTS-criterion. The difference between the two calculations remained for the whole crack trajectory below 0.8 mm.
- The fatigue curve showed similar results to that of Franc3D. Some difference was observed however after 1,700.000 cycles were applied. The difference after 800,000 cycles was approximately 7.7% and after 1,800,000 cycles 24.4%. Here, 800,000 cycles led to a crack depth of 4/15 through thickness and 1,800,000 cycles to a crack depth of 9.5/15 through thickness for XFEM results. Nonetheless, the shapes of the two curves followed the same trend. The crack propagation rate versus ΔK_{eff} showed a linear trend on the log-log scale representing the Paris region.

In this chapter is concluded that XFEM can provide good accuracy of stress intensity factors compered with conventional FEM. 3D crack propagation can be simulated by using a Python script to pre- and post-process the models and results in Abaqus. The next chapter is dedicated to a more complex structure, an orthotropic steel deck.

5

Orthotropic Steel Deck

In chapter 3, a through thickness crack was examined in an asymmetrical four point bending specimen. Accurate results were obtained for stress intensity factors (SIF's) and crack trajectory. Subsequently, in chapter 4, a more complex 3D crack was examined in a simple structure. The analysis of the cracked structure has proved to be more complex. Nonetheless , accurate results were also obtained for SIF, crack trajectory and fatigue life assessment. To conclude this research on the application of fracture mechanics, a 3D crack shall be examined in a complex structure, namely an orthotropic steel deck (OSD). This shall be the focus of this current chapter.

The crack to be examined is located at the lower weld toe of the rib-to-crossbeam joint of the OSD. When cope holes are applied, a stress concentration occurs at the weld toe induced by wheel loads. From there, cracks may arise and propagate through the rib. The location of this stress concentration is shown in figure 5.1. Furthermore, residual stresses due to welding may affect the cracking behaviour. The residual stresses however, are not included in the analysis but the weld shape itself is (see section 5.2.2 for more details). As a result of redistribution of stresses, this crack may not propagate until unstable propagation but rather reduces its propagation rate [27, 53].

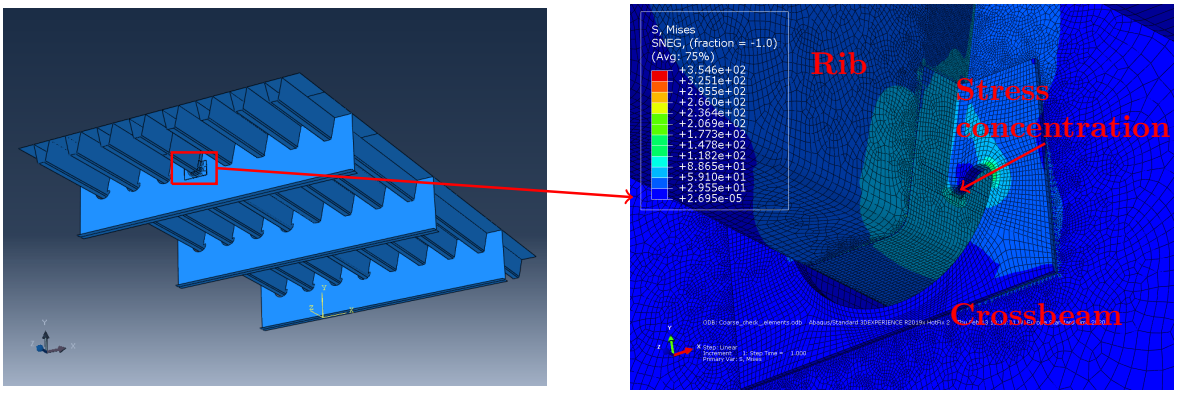


Figure 5.1: Global model of OSD (left) and location of cope hole (right)

5.1. Introduction of the specimen

At the TU Delft laboratory, an out of plane fatigue loading experiment on an orthotropic steel deck is performed and examined by W. Wu [49]. The experimental results of the fatigue tests are used to validate the XFEM calculation results.

5.1.1. Set-up

The orthotropic steel deck consists of eight ribs and three crossbeams. There are a total eight ribs, among which the left four contain Haibach shape cutouts and the right four are fully welded. In this research, a crack on the right side of rib five is studied. The experimental set-up is shown in figure 5.2. The rib is loaded with cyclic loads on either side of the crossbeam at rib five with an combined load of 500 kN [49] (2x250 kN). The crack propagates through the rib in the longitudinal direction on both the free edge side and the in-span side.

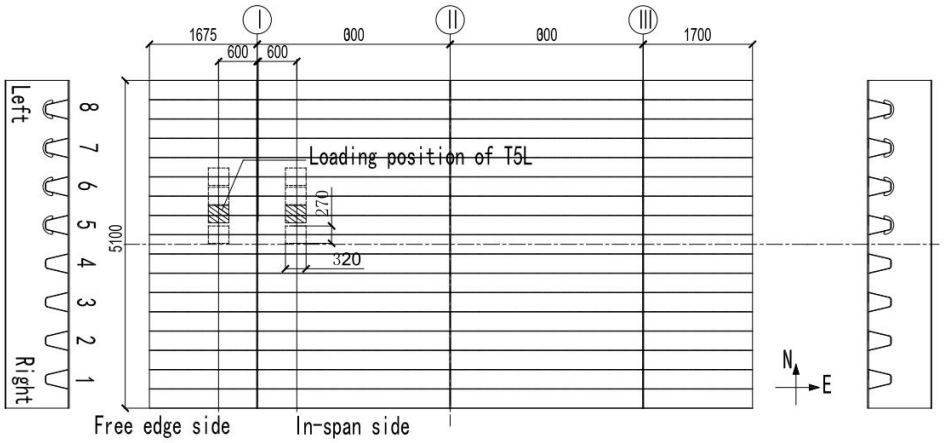


Figure 5.2: Overview OSD experimental setup [49]

5.1.2. Results

Measurements have been taken of the crack at the surface of the rib. It was observed that the crack tends to propagate first along the weld toe and subsequently in the longitudinal direction of the rib; this detail is visualised in figure 5.3. A simplified weld geometry is used as the actual weld, and has a more uneven geometry (see figure 5.9 for more details).

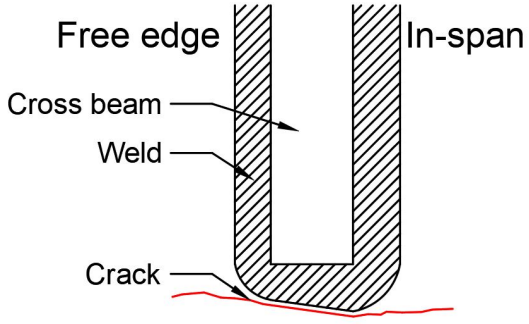


Figure 5.3: Rib-crossbeam detail including crack

As the crack is measured at several locations, the crack profile can be plotted and the crack lengths can be determined. However, the depth of the crack remains unknown as this was not measured. Therefore, assumptions must be made for the crack depth as discussed in section 5.2.5. The resulting crack profile is presented in figure 5.4, and the coordinates are listed in table 5.1. The local coordinate system as shown in figure 5.4, is used to present results obtained.

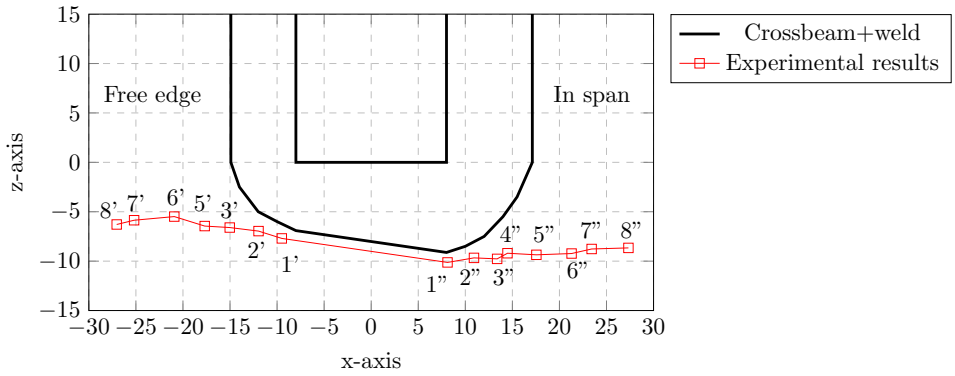


Figure 5.4: Experimental results of crack trajectory

In this figure, various points are labelled; crack tip coordinates and applied number of cycles are recorded at such measured positions and are listed in table 5.1. Linear interpolation was used between marks to obtain the crack profile. Labels with one apostrophe correspond to the free edge side whereas labels with two apostrophes correspond to the in-span side.

Mark	X-coordinate in mm	Z-coordinate in mm
8’	-27.04	-6.29
7’	-25.19	-5.86
6’	-20.92	-5.49
5’	-17.69	-6.45
3’	-15.03	-6.60
2’	-11.97	-6.96
1’	-9.50	-7.70
1”	8.11	-10.13
2”	10.92	-9.67
3”	13.38	-9.77
4”	14.48	-9.20
5”	17.55	-9.36
6”	21.29	-9.23
7”	23.42	-8.77
8”	27.33	-8.66

Table 5.1: Local coordinates along crack

As the load cycles are known at all marks, the fatigue life assessment can be performed. In table 5.2, the results are listed. Here, the number of cycles are obtained from the lab report [49] and the crack length was measured directly from the crack profile at the surface of the rib.

Marks	Crack length in mm	Cumulative cycles
1’-1”	17.78	8.89E+04
2’-2”	23.21	1.26E+05
3’-3”	28.75	1.41E+05
3’-4”	29.99	1.59E+05
5’-5”	35.72	1.96E+05
6’-6”	42.83	2.30E+05
7’-7”	49.30	2.66E+05
8’-8”	55.11	3.13E+05

Table 5.2: Fatigue data experiments of rib 5 right side

In figure 5.5, the fatigue life is plotted. Here we can see a somewhat linear trend between the crack length and the cumulative number of cycles.

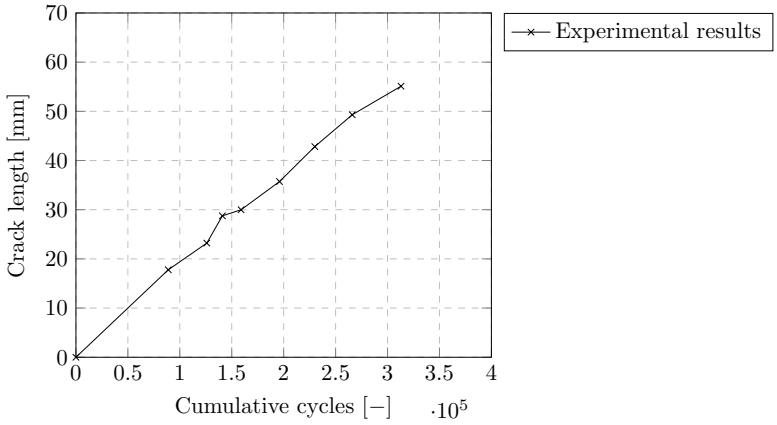


Figure 5.5: Experimental results of fatigue assessment

5.2. Computational model

In this section, the computational model is discussed. Abaqus is used to model the crack initiating from the lower weld toe and subsequent propagation in the rib. Crack extension is modelled using stationary XFEM crack analysis combined with a Python script, as described in chapter 4. This script is used to read the results from the calculation performed through Abaqus and determine the next crack shape for the analysis.

5.2.1. Meshing

The model comprises both plate and solid elements. Furthermore, the element size varies within the model (see table 5.3). At the location of the crack, a fine mesh was used while outside this region a rather coarse mesh was used. Three regions can be easily discerned. The first region consists of all plate elements, and contains the coarser mesh. The second region comprises the detail of interest, but excludes the crack region. The final and third region is that in which the crack is located and is composed of very fine mesh. The size of this region is restricted as the computation time highly depends on the number of elements used. From these three regions, regions 1 and 2 use quadratic elements and region 3 uses linear elements, given that quadratic elements are not available for XFEM analysis in Abaqus [1]. The three different regions are visualised in the figure below.

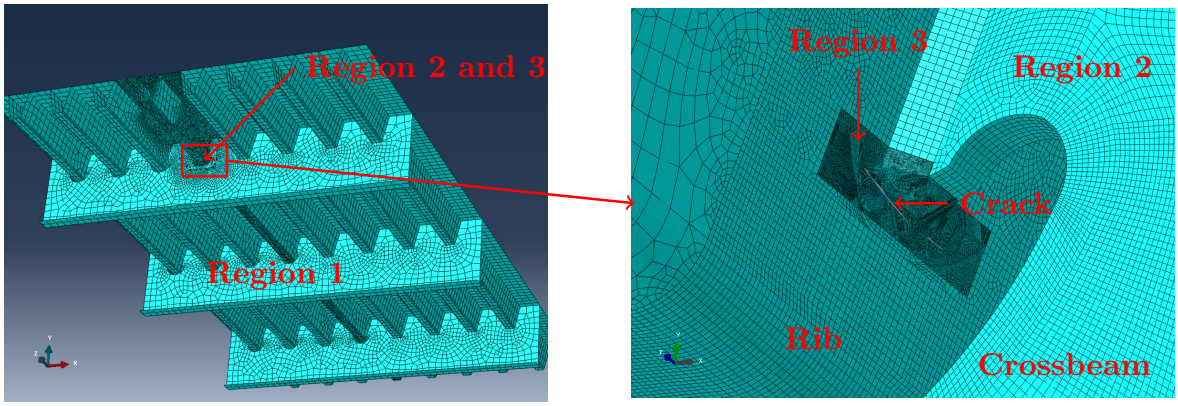


Figure 5.6: Various mesh regions in OSD model

The three regions are connected by use of tie constraints. The meshing information of these regions is summarised in table 5.3.

Region	Element type	Geometric order	Approx. mesh size in mm
1	Shell S8R	Quadratic	100
2	Solid C3D20R	Quadratic	2.00
3	Solid C3D8R	Linear	0.20-0.50

Table 5.3: Mesh information of OSD model

A mesh sensitivity study for region three is performed in order to ensure the mesh is fine enough for stable results. A range of element sizes have been used around the crack, from 0.5 to 0.2 mm. In the previous chapters (chapter 3 and 4), a local element size of 0.1 mm was found to yield the best results. However, the size of this OSD model is significantly larger and applying a mesh size of 0.1 mm will increase the computation time considerably and is therefore not applied here. To ensure accurate results, the stability of the stress intensity factors is checked in every step along the crack propagation. A mesh sensitivity analysis is performed for $\theta = 30°$ and $\theta = 90°$. The definition of θ is as presented in figure 4.2. For this mesh sensitivity analysis, K_I is examined given that mode I is dominant for this crack, as is shown in section E.3. In the two graphs below, the results are plotted for 10 contours.

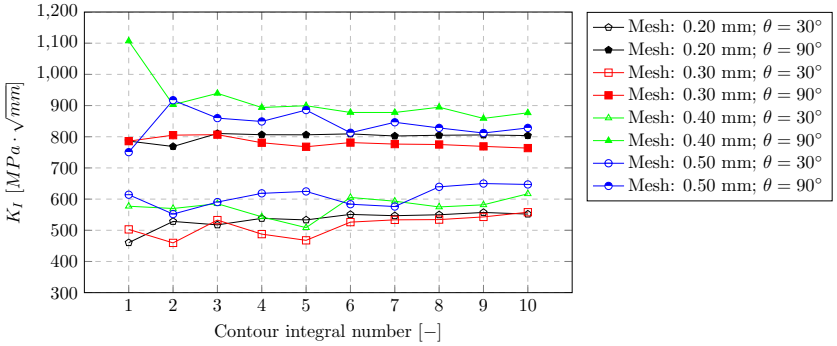
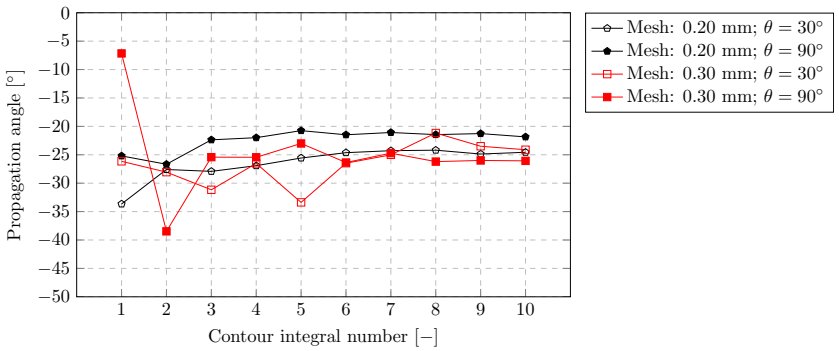
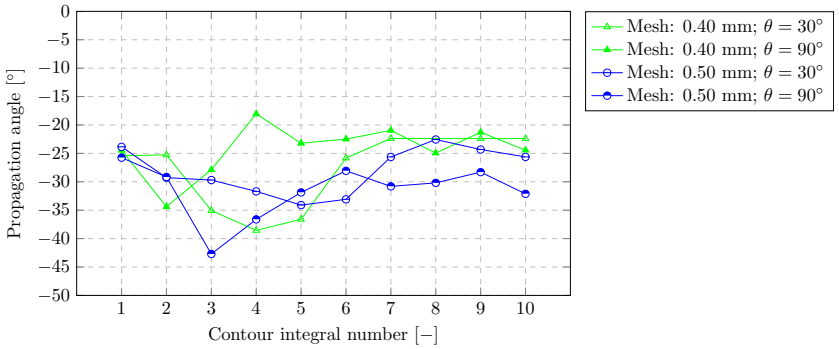


Figure 5.7: Mesh sensitivity analysis for K_I

In this graph, the values for K_I are much more stable for a mesh size of 0.20 mm than for a mesh size of 0.50 mm. Moreover, mesh size of 0.30 mm showed stable results for contour integral number 6-10. Generally, the smaller the elements, the more stable the contour integral results (as found in chapter 4). As the crack geometry along the crack propagation analysis is of importance, the stability of the crack propagation angle should be investigated. Therefore, a mesh sensitivity analysis is also performed on the crack propagation angle. The results are shown in the following graph.



(a) Mesh size of 0.20 mm and 0,30 mm



(b) Mesh size of 0.40 mm and 0,50 mm

Figure 5.8: Mesh sensitivity analysis for propagation angle

As shown in figure 5.8, the results for a mesh size of 0.20 and 0.30 mm proved stable while other mesh sizes (0.40 mm up to 0.50 mm) showed unstable behaviour. The first few integral numbers were mostly unstable and the last ones showed a higher stability. Therefore, the mean average is taken for contour numbers 6-10. A mesh size of 0.25 mm is used around the crack region to ensure stable results for the contour integrals. Such mesh size was chosen over 0.20 mm to reduce the computational time.

5.2.2. Modelling the weld geometry

Given that the crack originates from the weld toe, the geometry of the weld should be modelled with accuracy. Therefore, the geometry of the weld was measured from the experimental setup. A suitable model may then by created and used for numerical analysis. The measured results are shown in figure 5.10. As can be seen, the weld toe below the crossbeam is somewhat inclined. This can also be seen from the measured crack results (see section 5.1.2). Due to measurement conditions, a deviation of 1 mm in weld shape may occur as locally there are imperfections. The weld shape in experimental set-up is presented in figure 5.9.

(a) Free edge side

(b) In-span side

Figure 5.9: Photos of cracked detail in rib-to-crossbeam joint

The welded shape is modelled to mimic the measured cracked shape with an offset of 1 mm. Therefore, the weld shape can be determined below the crossbeam from the crack coordinates as listed in table 5.2. The distance from the crossbeam to the weld toe on the free edge side set to 6.91 mm. An equal value is used for the distance from crossbeam to weld toe below the crossbeam on the free edge side. This is similar to the measurements and can be modelled in Abaqus. Similarly, the distance between the weld toe and the crossbeam at the in-span side was set to 9.12 mm. The model of the weld used in the Abaqus model is shown in figure 5.10.

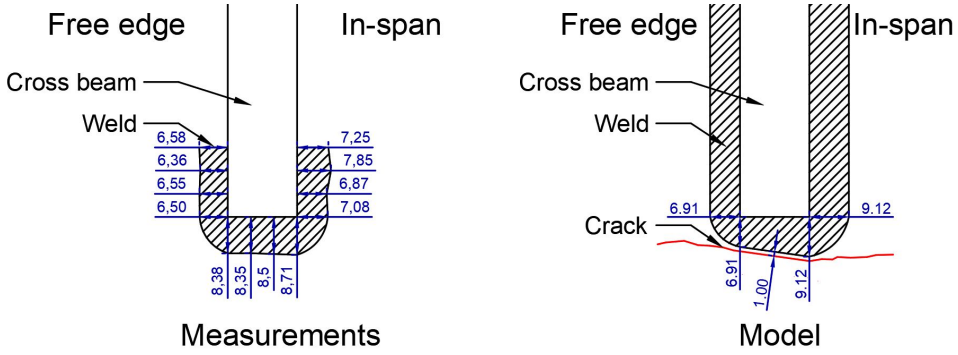


Figure 5.10: Weld detail, measurements (left) versus model (right)

5.2.3. Validation of model

In this chapter, the validation of the FEM model is performed. In the previous chapter, the new weld geometry was proposed. Therefore, a validation is needed to prove that the model represents the experimental set-up. The strains of the uncracked model have been computed and transformed into stresses by multiplication with the Young's modulus ($E = 210,000$ MPa). Both the global model and the sub-model were investigated to ensure that the entire model accurately represents the experimental setup. The results are plotted in the two graphs below. The finite element results resemble to the experimental results and this model can thus be used for further analysis.

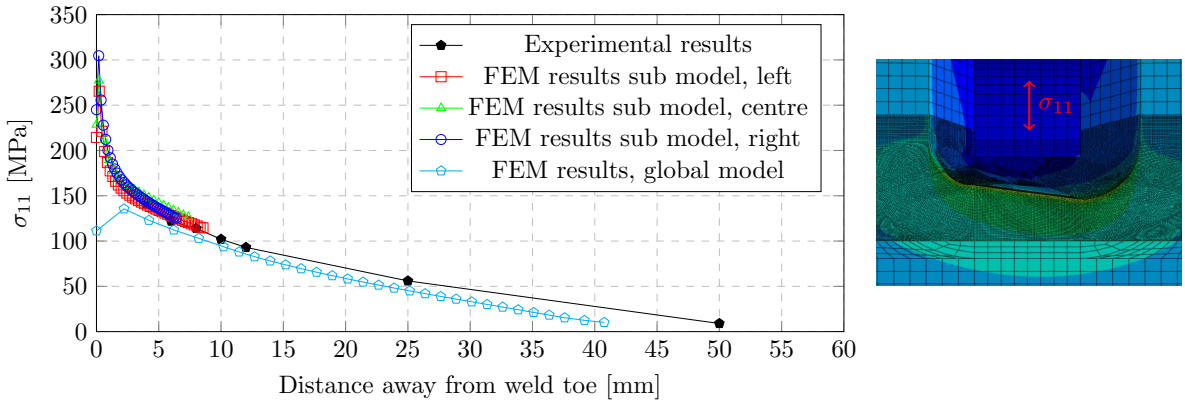


Figure 5.11: Validation of stresses in the vertical direction

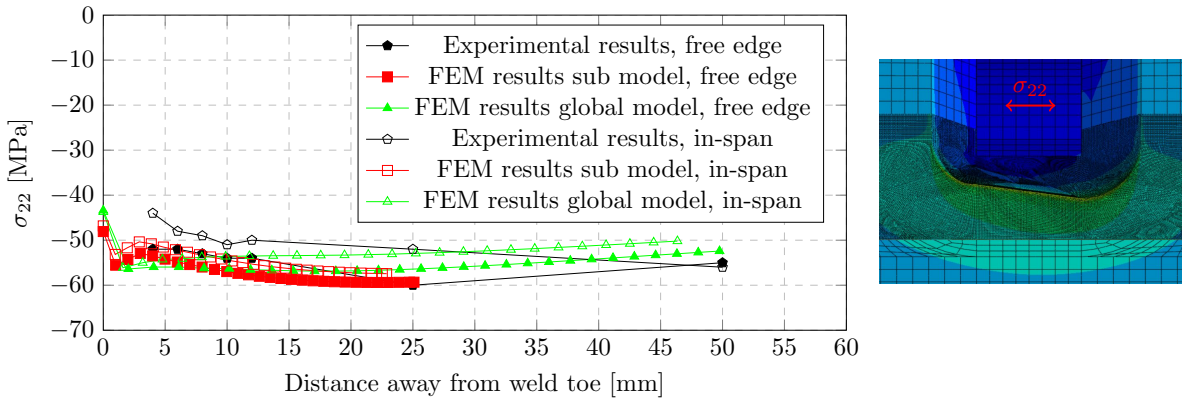


Figure 5.12: Validation of stresses in the horizontal direction

The stresses σ_{11} and σ_{22} from the FEM calculation correlate with those measured from the experiment, as shown in figures 5.11 and 5.12. Therefore, it is assumed that this model is suitable for crack propagation analysis. Prior to such analysis, the location of the crack and the initial crack parameters are discussed.

5.2.4. Crack position

In this section, the position of the crack is discussed. The coordinates of the crack with respect to the cross-beam are presented in table 5.1. An initial crack starting at mark 1'-1" is chosen for crack propagation analysis as these are the first marks at which data is observed (see figure 5.4 for more details). The assumed initial crack is shown in the following graph. The corresponding coordinates are listed in table 5.1.

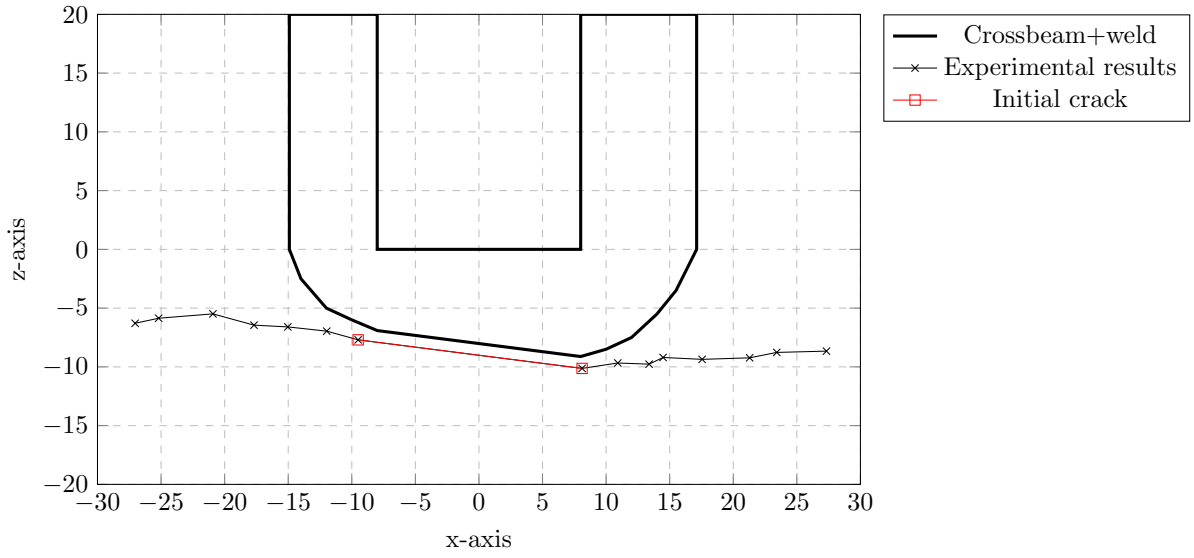


Figure 5.13: Assumed initial crack in model

5.2.5. Initial crack parameters

The crack profile on the surface of the rib was obtained from experimental data. These are X- and Z-coordinates of the crack from the local coordinate system as shown in the graph above. However, the depth of the crack and the shape of the crack inside the rib remains unknown. Therefore, assumptions are made to solve this problem.

It is assumed that the top view of the crack has a semi-elliptical shape (the XY-view), such that one parameter defines the assumed shape, namely the crack depth a. Moreover, initial crack depth of 1 mm is assumed. This yields a somewhat constant distribution of K_I along the crack front. Furthermore, as the crack tends to propagate in the upwards direction, as shown in appendix E.3.1, choosing a small initial depth allows the

crack to find its correct shape in the thickness direction of the rib. To determine the inner coordinates of the crack front the following formula are used:

$$\theta = \arccos(x/c) \tag{5.1}$$

$$y = a \cdot \sin\theta \tag{5.2}$$

where:
- a = Crack depth;
- c = Half crack length;
- x, y = Crack front coordinates in mm;
- θ = Position specifier in degrees (see figure 4.2).

Therefore, provided that parameters c and x are known and the value for a is assumed, y-values can be determined. Since X- and Y-coordinates of the crack front are determined, only Z-coordinates along the crack front are to be found. Regarding the initial crack, it is assumed that the Z-coordinates are equal to those of the crack on the surface of the rib. In the following graph, the top view of the modelled initial crack is presented.

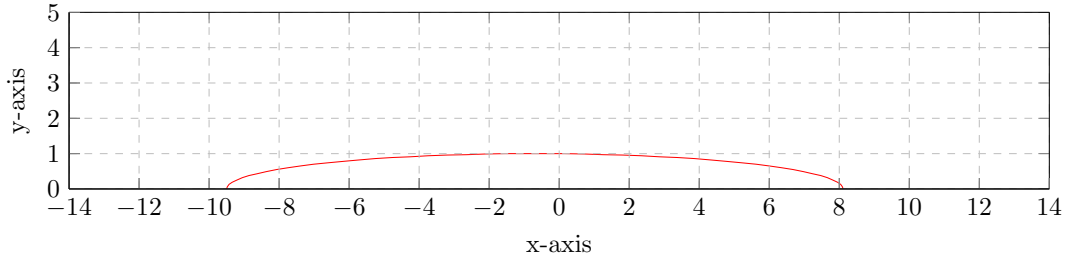


Figure 5.14: Top view of initial crack

In the table below, all X-, Y-, and Z-coordinates are listed along the initial crack front. This concludes all information needed for the initial crack in the model. Next, the methodology of crack propagation is discussed.

X-coordinate in mm	Y-coordinate in mm	Z-coordinate in mm
-20.92	0.00	-5.49
-20.46	0.62	-5.62
-19.10	1.22	-6.03
-17.70	1.59	-6.45
-15.04	2.08	-6.60
-11.98	2.45	-6.96
-9.51	2.67	-7.70
-6.34	2.85	-8.13
-3.12	2.96	-8.58
0.18	3.00	-9.04
4.57	2.93	-9.64
8.12	2.78	-10.14
10.93	2.58	-9.68
13.39	2.34	-9.78
14.49	2.21	-9.21
17.55	1.71	-9.36
19.47	1.22	-9.30
20.83	0.62	-9.25
21.29	0.00	-9.23

Table 5.4: XYZ coordinates along initial crack front

5.3. Crack propagation methodology

The crack propagation methodology differs slightly from that of chapter 4. The main difference between them is procedure for smoothing results. The method in chapter 4 uses smoothing on the front of the new crack only. The crack propagation methodology used in chapter 4 is presented in section 4.5 and in appendix D. In this chapter, smoothing is used on both the contour integral output as well as on the crack front. Manipulation of the contour integral output is needed to obtain smooth results as the output can be prone to oscillations. The following manipulations have been used:

- Excluding inaccurate contour integral output on the surface of the structure;
- Regression on the contour integral output as well as the crack front.

Contour integral output of XFEM may show strong oscillating behaviour as is discussed in chapter 4. The results can be improved by smoothing the calculated stress intensity factors. A 9^{th} order polynomial is used to fit the raw data obtained from the Abaqus output. The contour integral output may show inaccurate values at the boundaries of the structure. Therefore, these values should be disregarded. This can be manipulated by performing the regression polynomial not taking into account the values on the boundary of the structure and extrapolating the results. Extrapolating a 9^{th} order polynomial may also yield inaccurate values as the polynomial may have a high slope at that specific location. The results are compared with the raw data after smoothing to avoid the inaccuracy caused by the high order polynomial functions in such positions with large slopes. An example of data smoothing is shown in the graph below. Here, highly inaccurate values at the boundary of the structure are present. In this example, the two data on either side of the surface of the structure are disregarded and extrapolated.

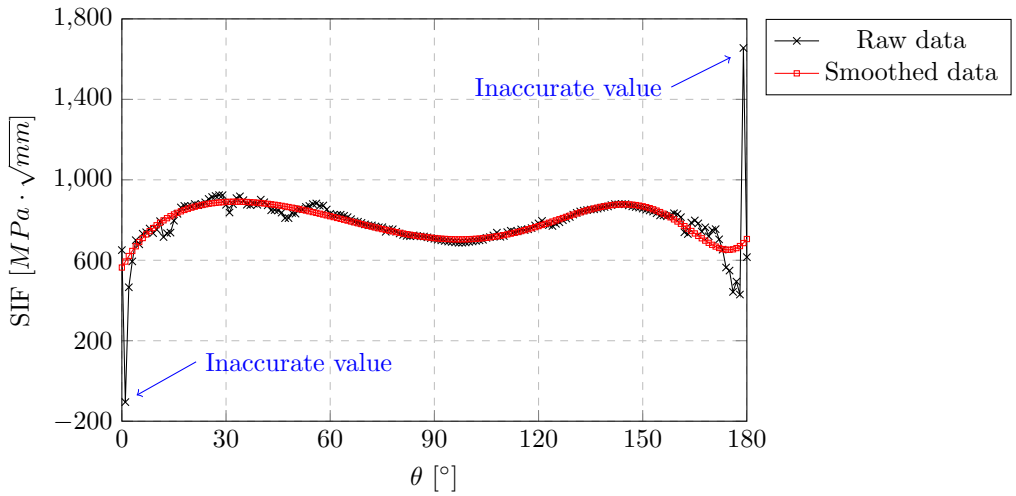


Figure 5.15: Comparison of smoothed data with raw data

5.4. Crack propagation results

In appendix E, crack propagation was examined starting at mark 6'-6" (see figure 5.4 for more details). These results showed good resemblance with the experimental results. Therefore, crack propagation was also investigated for an initial crack starting at mark 1'-1" (see section 5.1.2). In this chapter, the crack trajectory is presented prior to the stress intensity factors. Finally, the fatigue life assessment is discussed.

5.4.1. Crack trajectory

Provided that the experimental results of the crack propagation could only be measured at the surface of the rib, the results are compared at this surface. The front view of the initial crack shape is shown in the graph below. The results of the modelled crack propagation are also plotted in this graph.

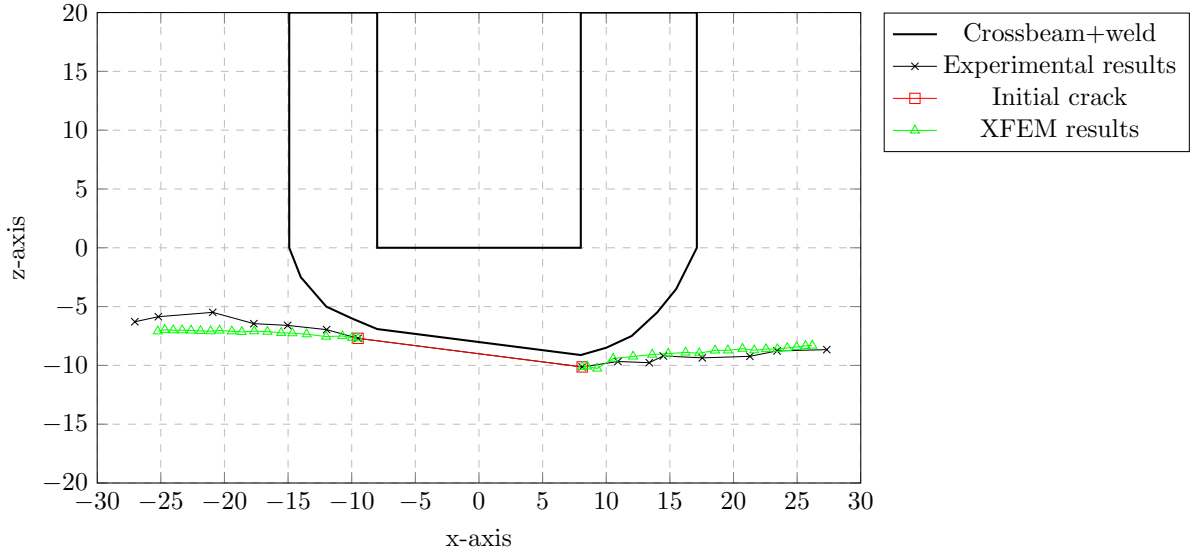


Figure 5.16: Crack propagation results starting at mark 1'-1"

From the graph above, containing the front view of the crack, the results from the modelled crack correlate to the measured crack from the experiment. Of the first five steps, some coordinates of the crack front are presented in table 5.4.1. Given the first nine steps of the propagation analysis, minor differences were observed, all of which are well below 1 mm. Afterwards, a difference of approximately 1 mm was observed at the free edge side; the results of the in-span side remained very close to the experimental results.

Table 5.5: Coordinates along crack front

Step 0			**Step 1**			**Step 2**		
X	Y	Z	X	Y	Z	X	Y	Z
-9.50	0.00	-7.70	-9.97	0.00	-7.64	-10.73	0.00	-7.50
-9.43	0.19	-7.72	-7.81	1.18	-7.83	-8.45	1.95	-7.65
-8.63	0.43	-7.82	-6.53	1.44	-7.99	-7.04	2.28	-7.96
-7.75	0.60	-7.94	-5.24	1.58	-8.16	-5.62	2.42	-7.96
-6.18	0.78	-8.16	-3.95	1.66	-8.34	-4.21	2.52	-8.15
-4.77	0.89	-8.35	-2.66	1.72	-8.51	-2.80	2.61	-8.36
-2.65	0.97	-8.64	-1.37	1.77	-8.69	-1.38	2.69	-8.56
0.15	1.00	-9.03	-0.09	1.79	-8.85	0.03	2.72	-8.76
1.26	0.97	-9.19	0.83	1.78	-8.97	1.04	2.71	-8.89
3.38	0.89	-9.48	2.12	1.74	-9.14	2.46	2.66	-9.05
4.79	0.78	-9.67	3.41	1.66	-9.31	3.87	2.57	-9.17
6.36	0.60	-9.89	4.70	1.51	-9.49	5.28	2.43	-9.28
7.24	0.43	-10.01	5.99	1.26	-9.68	6.70	2.12	-9.44
7.95	0.19	-10.11	7.27	0.80	-9.89	8.11	1.36	-9.76
8.11	0.00	-10.13	8.51	0.00	-10.09	9.29	0.00	-8.92

Step 3			**Step 4**			**Step 5**		
X	Y	Z	X	Y	Z	X	Y	Z
-12.01	0.00	-7.55	-13.54	0.00	-7.35	-14.66	0.00	-7.26
-9.41	2.31	-7.42	-10.05	2.63	-7.29	-12.23	2.41	-7.04
-7.84	2.71	-7.60	-8.28	3.01	-7.51	-10.35	3.11	-7.16
-6.28	2.89	-7.83	-6.51	3.22	-7.76	-8.47	3.36	-7.44
-4.71	3.03	-8.06	-4.73	3.38	-8.01	-6.59	3.52	-7.74
-3.15	3.15	-8.27	-2.96	3.50	-8.26	-4.71	3.69	-8.01
-1.58	3.23	-8.47	-1.44	3.56	-8.47	-2.57	3.86	-8.29

Continued on next page

Table 5.5 – *Continued from previous page*

X	Y	Z	X	Y	Z	X	Y	Z
-0.02	3.23	-8.67	0.08	3.57	-8.66	0.12	3.93	-8.62
1.32	3.18	-8.84	2.11	3.51	-8.89	1.73	3.86	-8.80
2.89	3.09	-9.02	3.88	3.41	-9.04	3.61	3.73	-8.97
4.45	3.02	-9.15	5.66	3.29	-9.13	5.49	3.59	-9.06
6.02	2.96	-9.22	7.43	3.07	-9.14	7.37	3.47	-9.05
7.58	2.73	-9.24	9.21	2.58	-9.12	9.25	3.25	-8.95
9.15	1.90	-9.27	10.98	1.36	-9.16	11.13	2.55	-8.90
10.54	0.00	-9.42	12.11	0.00	-9.24	13.61	0.00	-9.08

With the data given in table 5.4.1, the propagating crack can be plotted. These are shown in the graphs below. In these graphs, the smooth crack front can be observed after each step in the crack propagation analysis. This is a result of the data fitting polynomial as explained in section 5.3. As the depth of the crack of the experiment remains unknown, no validation on this part is performed. Nonetheless, presenting the modelled crack propagation in the thickness direction may prove instructive and is therefore presented.

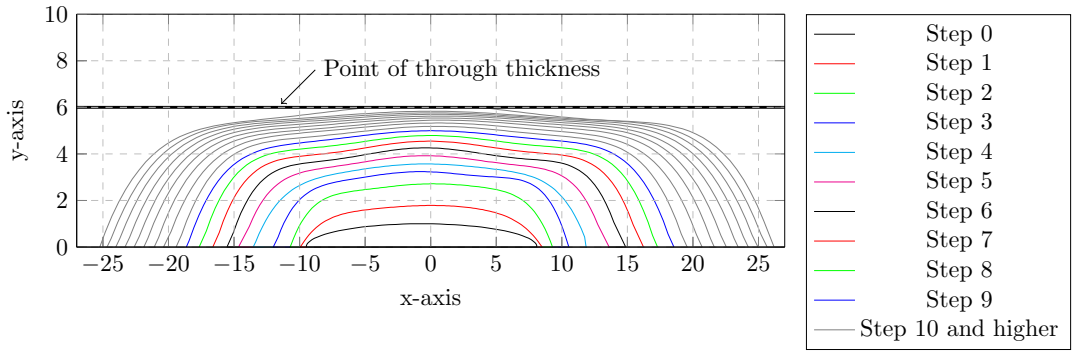


Figure 5.17: XFEM crack propagation results, top view

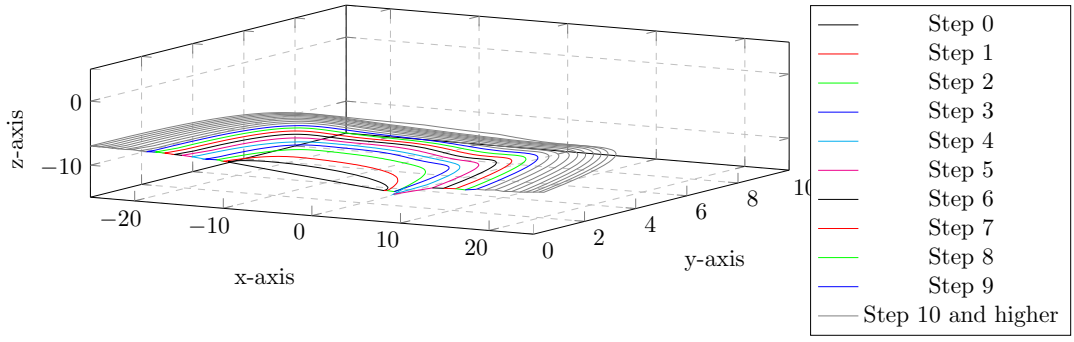


Figure 5.18: XFEM crack propagation results, isometric view

5.4.2. Stress intensity factors

The main data output of the calculations performed through Abaqus are stress intensity factors along the crack front for each mode. These are presented in the graphs below for each step in the analysis.

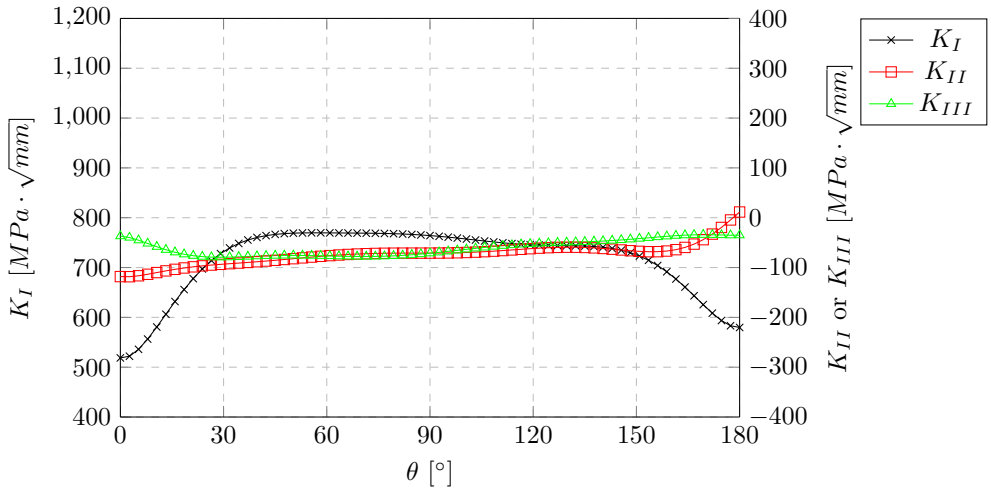


Figure 5.19: Stress intensity factor results step 0

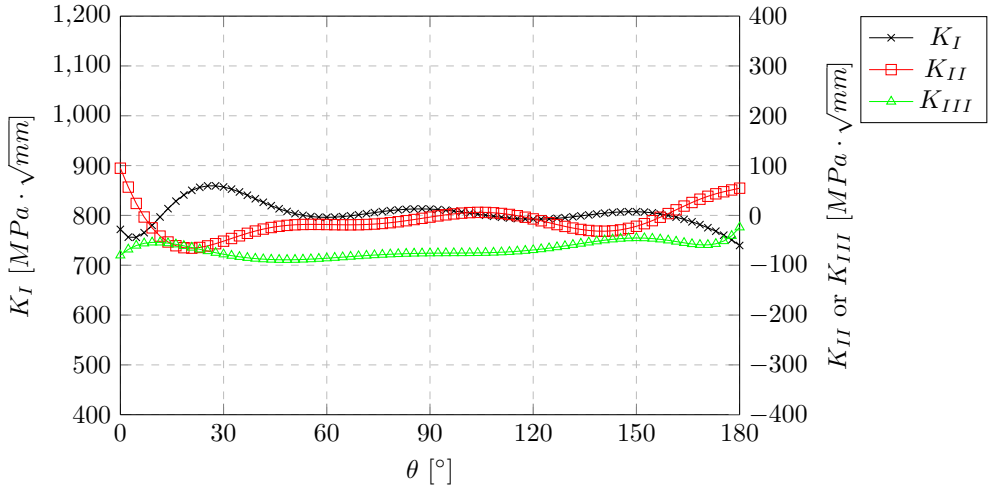


Figure 5.20: Stress intensity factor results step 1

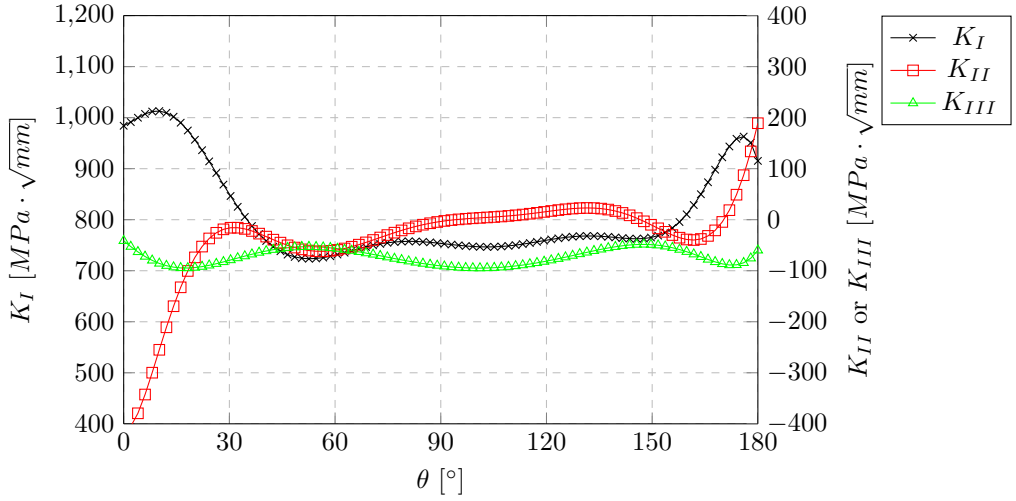


Figure 5.21: Stress intensity factor results step 2

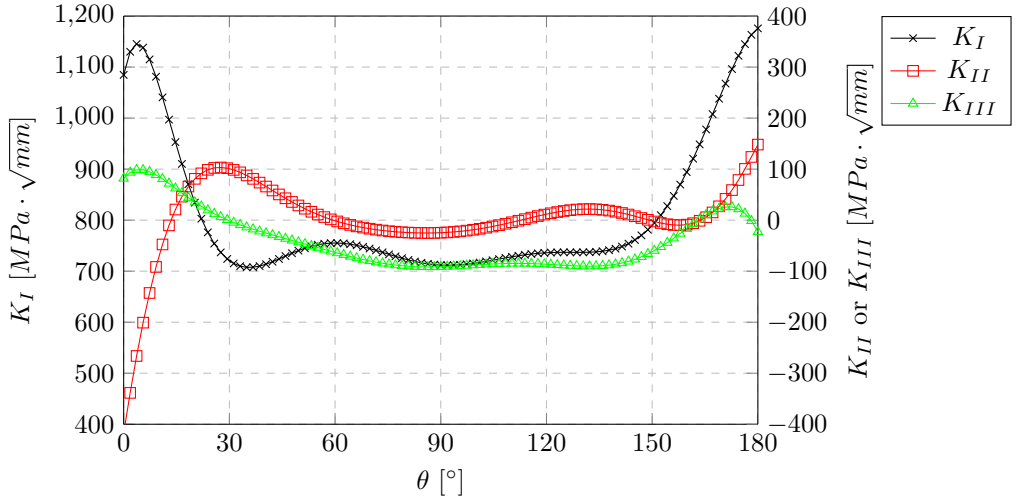


Figure 5.22: Stress intensity factor results step 3

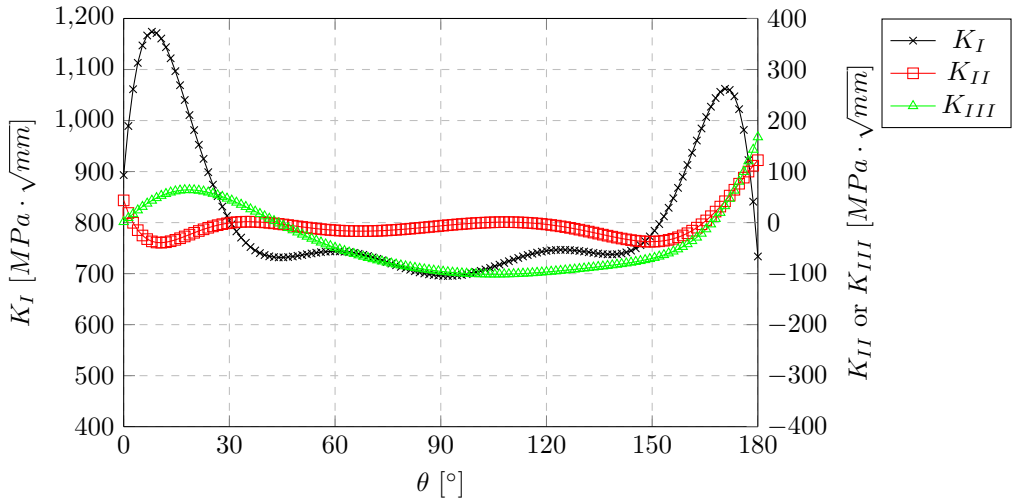


Figure 5.23: Stress intensity factor results step 4

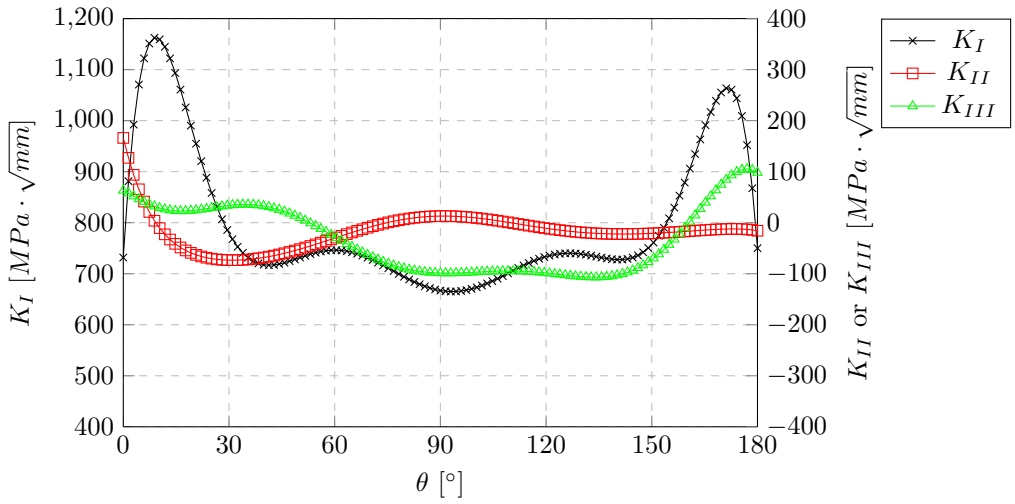


Figure 5.24: Stress intensity factor results step 5

As was the case for the results in appendix E.3.2, the values for K_I are significantly higher than those of K_{II} and K_{III} leading to a dominant mode I cracking behaviour. Also K_I, for step two onwards, is much higher for $0° < \theta < 30°$ and $150° < \theta < 180°$ than for $30° < \theta < 150°$. As a result. the crack tends to propagate faster at the surface of the rib than in the thickness direction.

5.4.3. Fatigue life assessment

Fatigue life assessment is discussed in this section. For each step, the number of load cycles are selected and the crack extension is calculated. Subsequently, the cumulative number of cycles are determined as well as the crack length. In the crack propagation analysis, the Paris law was implemented as crack propagation relation:

$$\Delta a = \Delta n \cdot C \cdot \Delta K_{eff}^{m} \tag{5.3}$$

$$\Delta K_{eff} = \sqrt[4]{(\Delta K_I)^4 + 8(\Delta K_{II})^4} \tag{5.4}$$

where:

Δa	= Crack extension in mm;
Δn	= Number of cycles responsible for Δa;
ΔK_I	= Stress intensity factor range corresponding to mode I;
ΔK_{II}	= Stress intensity factor range corresponding to mode II.

Parameters C and m have been selected as $3.98 \cdot 10^{-13}$ and 2.88 respectively, as recommended in BS7910 [13]. In the table below, the fatigue data is listed for each step in the crack propagation analysis. These data are plotted in figure 5.25. The experimental data can be found in table 5.2.

Step	Δa in mm	Δn in cycles	Crack length in mm	Cumulative number of cycles
0	-	-	17.78	88,900
1	0.88	10,000	18.66	98,900
2	2.18	10,000	20.84	108,900
3	2.63	7,100	23.47	116,000
4	3.12	5,000	26.59	121,000
5	2.63	5,000	29.22	126,000
6	2.14	5,000	31.36	131,000
7	2.45	5,000	33.81	136,000
8	2.11	5,000	35.92	141,000
9	2.25	5,000	38.17	146,000
10	1.79	5,000	39.96	151,000
11	2.11	5,000	42.07	156,000
12	1.65	5,000	43.72	161,000
13	1.72	5,000	45.44	166,000
14	1.64	5,000	47.08	171,000
15	1.48	5,000	48.56	176,000
16	1.45	5,000	50.01	181,000
17	1.38	5,000	51.39	186,000
18	1.10	5,000	52.48	191,000

Table 5.6: Fatigue life assessment data

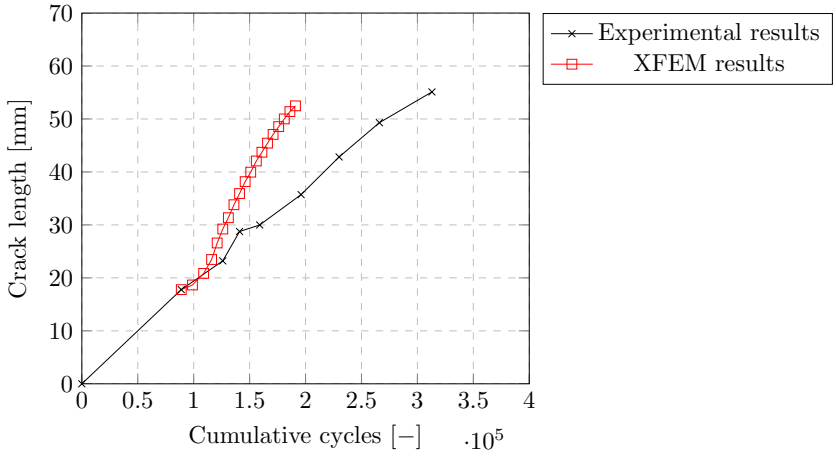


Figure 5.25: Fatigue life assessment

At first, the results show close resemblance with the experimental data. However, after step 5, more significant differences occur. Therefore, a parametric study is performed to investigate the sensitivity of the Paris law constant C. These results are presented in figure 5.26.

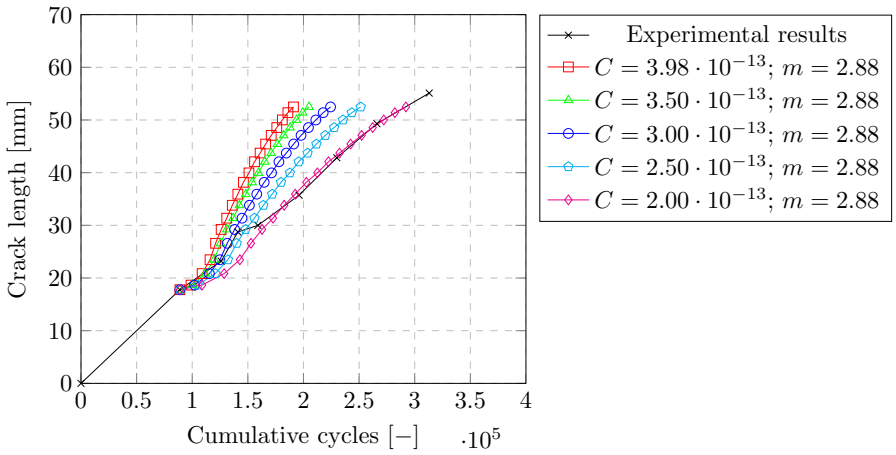


Figure 5.26: Parametric study for Paris law parameters

As presented in figure 5.26, selecting a different value for C may yield in closer results with respect to the experimental values. Herefrom, a value of $2.00 \cdot 10^{-13}$ proved to fit the experimental results best given a value of 2.88 for m.

5.4.4. Parametric study

As the depth of the crack could not be measured, an initial crack depth was assumed as treated in section 5.2.5. To investigate the sensitivity of this assumption, a parametric study was performed. Initial crack depths of 1-, 2- and 3 mm have been investigated of which the top view is presented in the following figure. A semi-elliptical crack shape was assumed in all cases.

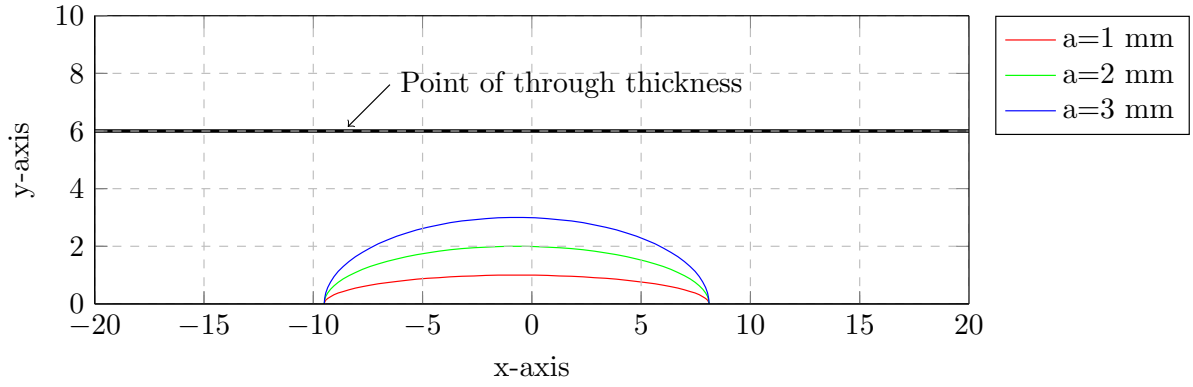


Figure 5.27: Parametric study for initial crack depths

With these three various crack depths, crack propagation was analysed. Due to the computational time, crack propagation was only executed for $a = 2$ mm and $a = 3$ mm for several steps. However, from these steps, the effect of the initial crack depth can be determined. Firstly, the crack propagation trajectories are presented in figure 5.28. Here, the differences between the three cracks remains small. For all three propagation analyses, the crack propagated in similar directions while the depths of the cracks are still different. However, the difference in depth becomes smaller as more steps are performed in the analysis. For example, step 7 in figure 5.29a, step 5 in figure 5.29b and step 3 in figure 5.29c, all show a depth of approximately 4.5 mm. For all cases, the crack length, on the surface of the rib was 32 mm at this point of crack depth. The top view of the various crack propagation results is presented in figure 5.29.

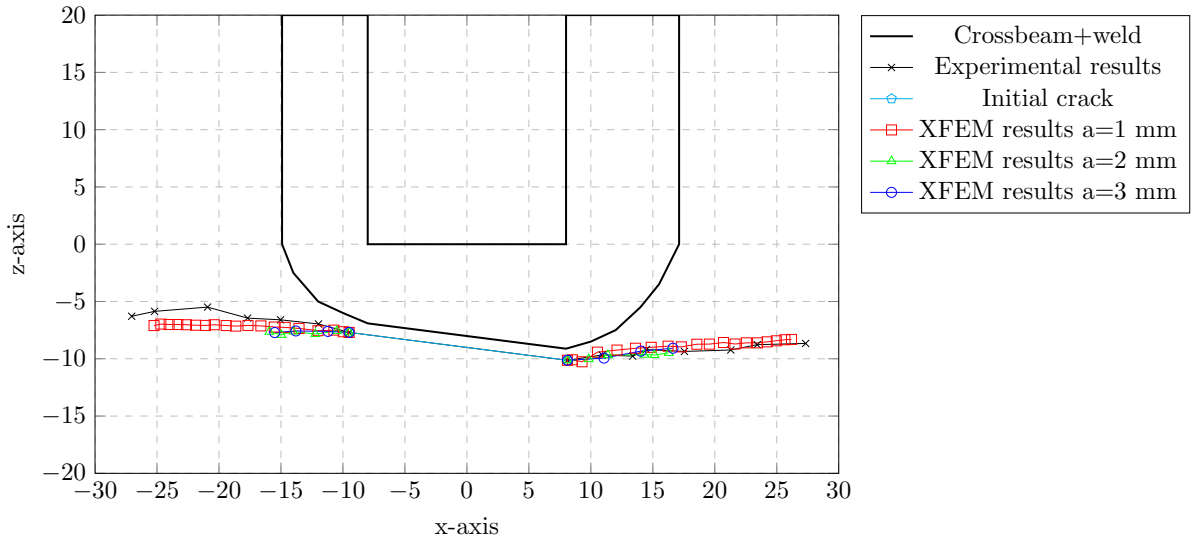
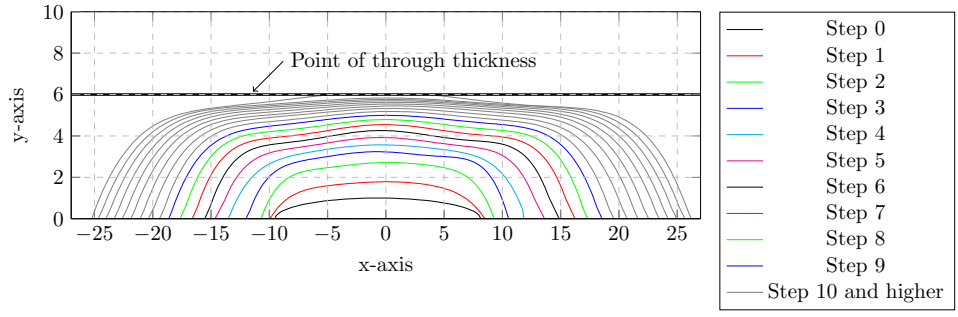
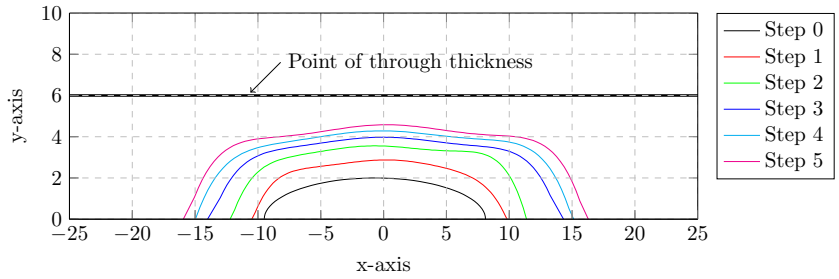


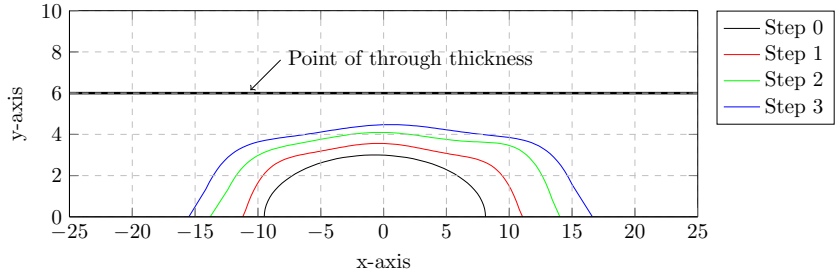
Figure 5.28: Parametric study for propagation results



(a) Results for initial crack depth of 1 mm



(b) Results for initial crack depth of 2 mm



(c) Results for initial crack depth of 3 mm

Figure 5.29: Top view of propagation results for various initial crack depths

Finally, the fatigue life assessment is analysed. The Paris law parameters were selected after the parametric study performed in section 5.4.3. This resulted in values of $2.00 \cdot 10^{-13}$ and 2.88 for C and m respectively. The

results are presented in figure 5.30.

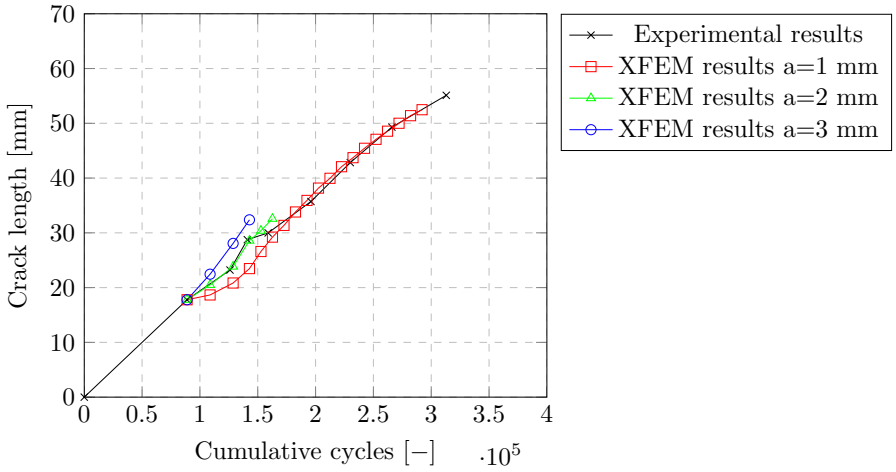


Figure 5.30: Parametric study for fatigue life assessment

5.5. Conclusion

In this chapter, 3D crack propagation was investigated in the weld toe crack of the rib-to-crossbeam joint in an orthotropic steel deck (OSD). The study is based on an analysis of the crack observed in the experiment carried out at TU Delft laboratory. In this experiment, the crack was initiating from the lower weld toe and was propagating in the rib. The length of the crack was carefully measured by marking the crack tips during the cyclic loading. However, the crack depth and shape inside the thickness of the rib cannot be measured, and assumptions were therefore made of a semi-elliptical shape for the initial crack insert. Using Abaqus, the extended finite element method (XFEM) was employed to obtain contour integral results along the crack front. Stress intensity factors for modes I, II and III as well as crack propagation angles are calculated using this method. These were used with a Python code to determine the new crack front after a predefined number of cycles. Using the Paris law, a fatigue life assessment was carried out and compared with the experimental results. A parametric study shows the effect of the parameter C in the Paris law. The following is concluded.

- The stress intensity factor values for mode I were significantly larger than those of mode II and III for all steps in the analysis along the entire crack front. Therefore, this crack is driven mainly by the opening mode I.
- The crack grows at a somewhat constant propagation rate, this is reflected in a linear trend in the crack length versus cumulative cycles plot (see figure 5.30 for more details). Furthermore, the crack propagation in the thickness direction of the rib was considerably slower than in the longitudinal direction. The crack length, as measured on the surface of the rib, is approximately 8 times larger than the depth of the crack.
- The crack shapes from the XFEM calculations closely correlate to the experimental results, as minor differences were found of approximately 1 mm. These could be perfected by using a smaller number of applied load cycles per step, choosing a more refined mesh, applying a more realistic weld shape and including residual stresses. Furthermore, errors can originate from the measurements of the experiments.
- The fatigue life assessment from the XFEM model shows conformity with the results from the experiments. A parametric study was performed on the Paris law parameter C for calibration of results. From this, the values of $2.00 \cdot 10^{-13}$ and 2.88 for C and m respectively, corresponds best with the experimental results.

In general terms, the modelled crack propagation and the experimentally determined crack results show significant resemblance. As a result, this methodology for crack propagation analysis is promising for further application.

6

Conclusion, Discussion and Recommendations

This final chapter of this thesis is focused on the conclusions drawn from this research. Firstly, the conclusions are stated, after which a discussion is presented. Finally, recommendations for further research are discussed.

6.1. Conclusion

This research comprises a study on both the literature and application study on fracture mechanics, primarily on orthotropic steel decks. Here, computational fracture mechanics, using the extended finite element method on Abaqus, was used to analyse three different structures. These contain an asymmetrical four point bending specimen, a semi-elliptical centre cracked plate and finally an orthotropic steel deck. Validation was used by experimental results, commercial fracture and fatigue software Franc3D, as well as research papers. A Python script was composed for data analysis from Abaqus output as well as to determine crack propagation. The computational results are compared with the validation data. The following is concluded.

- Despite the high computational effort, the results prove potential of this methodology in modelling of crack propagation in research application.
- Crack propagation with XFEM can be modelled using a constant mesh. Care should be taken on inaccurate contour integral values and oscillating behaviour. A polynomial fit can be used to smooth the results and thereby minimising the effect of the oscillation.
- To obtain accurate results for crack propagation and fatigue life assessment, a low amount of load cycles per step in the analysis should be used and a fine mesh should be applied in the cracked region.
- Paris law parameters should be chosen with care as they have a great influence on the fatigue life assessment. After calibration, $C = 2.00 \cdot 10^{-13}$ and $m = 2.88$ showed best results for fatigue life assessment for the specific case of the OSD study.

6.2. Discussion

In this thesis, research questions were constructed of which the main research question is as follows:

How can crack growth, in orthotropic steel decks, be modelled using fracture mechanics concepts?

During this thesis, a methodology for crack propagation was used for analysis of crack growth in steps. While XFEM in Abaqus holds a tool for automatic crack growth, the data used to compute the crack remains mainly unknown for the user. As the contour integral output may be sensitive and subjected to oscillating behaviour, managing the data for accurate crack modelling was considered highly important. Therefore, stationary crack analysis was used to obtain all data, after which this was manipulated. Here, unrealistic data was disregarded (mainly at the surface of the structure) and polynomial functions were used to fit such results. This was

necessary to obtain a new smooth crack front in 3D crack propagation. A script was composed to obtain and manipulate the data. Furthermore, the new crack was determined through this script. This methodology holds many benefits, for instance:

- Stress intensity factors (SIF's) of each mode are obtained along the entire crack front. As a result of the magnitude of the three SIF's, the dominating cracking mode(s) can be concluded.
- Any crack propagation relation and effective stress intensity range definition can be used considering stress intensity factors of the three modes.
- Unrealistic data output can be disregarded from the analysis as all data can be manipulated by the user. It should be addressed that the user should not manipulate data to obtain closer results to experimental results. Instead, objective judgement should be made whether data is unrealistic and should be disregarded.
- A clear crack shape can be obtained by using a polynomial fit. Here, for each step in the analysis the crack front and its coordinates are known and can be obtained.

This methodology was used for both through thickness cracks and 3D cracks. While in all cases accurate data was obtained, the results have shown to be sensitive. An extremely fine mesh should be used to obtain accurate contour integral output. Given that for such method a constant mesh along the entire crack propagation analysis was used, spiderweb-type meshes were not an option. If a spiderweb-type kind of mesh is demanded, conventional FEM analysis may prove to be sufficient as well. Enrichment functions in XFEM provide significant benefits in modelling as the crack shape may have a complex geometry. In this case, no further treatment is needed for the mesh other than refinement. On the other hand, the modelling of the larger part with a fine mesh results in a significantly increased computation time due to the increased number of elements.

In addition to the mesh, the fatigue life assessment has shown to be quite sensitive to the constants used in the Paris law. These should be chosen with care as the fatigue life assessment may be demanded with accuracy (i.e. for the determination of inspection intervals). As this may vary per application, no universal values are defined. Nonetheless, for the case study considering the orthotropic steel deck (OSD), the values of $2.00 \cdot 10^{-13}$ and 2.88 for C and m respectively have shown to be accurate. However, the results were based on the assumption of an initial crack depth of 1 mm. Moreover, it should be noted that this corresponds to a experimental set-up which is indoors where weather and corrosion does not influence the results. In outdoor applications, higher values are more likely to be the more accurate.

Given the commercial nature of this research, due to the collaboration with TU Delft and engineering firm Iv-infra, application in design practice is reviewed. This methodology of crack propagation has proved to accurately describe both the crack path as well as the fatigue life assessment. However, as fracture mechanics concepts are used, an initial crack must be present for the analysis. For a new design, this may be an issue for determining the total fatigue life of the structure. Therefore, this approach may yield more value in re-evaluation of structures and determination of inspection intervals. Moreover, the fatigue life assessment has shown to be sensitive to the chosen parameters in Paris' law. Especially parameter 'm' should be chosen with care due to the powered relationship with the crack propagation rate as can be seen in figure E.14. While validation with experiments can lead to calibration of these parameters, this is not intuitive in new designs or re-evaluations. This is primarily due to the lack of fatigue data together with crack measurements throughout the lifetime of the structure. Given that this data often remains unknown, choosing parameters of crack propagation relations can become an arduous task.

6.3. Recommendations

This final part of this thesis is dedicated to recommendations for future research. These are listed below:

- Using smaller step sizes (number of cycles per step) may improve the results. It could prove useful to obtain knowledge in the effect of the step size in the crack propagation analysis. As a result, the step size can be chosen with more confidence.
- Implementation of residual stresses may improve the results. It would be interesting to investigate the effect of residual stresses on both crack shape and fatigue life assessment.

- Implementation of more complex crack propagation relations may be performed to describe the fatigue life assessment with higher accuracy.
- A comparison study may be performed between the methodology performed in this thesis and the automated crack propagation procedure by Abaqus. From this, a recommendation may originate for which method is the more accurate, time inexpensive and user friendly.
- As this methodology has proven useful for propagating cracks in the rib-to-crossbeam joint, this method may be extrapolated to the analysis of different cracks in orthotropic steel decks.

Bibliography

[1] Abaqus/cae user's guide (6.13). *Simulia*, .

[2] Abaqus analysis user's guide (6.13). *Simulia*, .

[3] Abaqus scripting user's guide (6.13). *Simulia*, .

[4] Abaqus theory guide (6.13). *Simulia*, .

[5] A. Ahmed. Extended finite element method (xfem) - modeling arbitrary discontinuities and failure analysis. *Pavia University*, 2009.

[6] M. Alam, Z. Barsoum, P. Jonsén, H. Häggblad, A. Kaplan, et al. Fatigue behaviour study of laser hybrid welded eccentric fillet joints: Part ii: State-of-the-art of fracture mechanics and fatigue analysis of welded joints. In *Nordic Conference on Laser Processing of Materials: 24/08/2009-26/08/2009*. ATV-SEMAPP, 2009.

[7] T.L. Anderson. Fracture mechanics: Fundamentals and applications. *CRC press*, 2017.

[8] A.O. Ayhan. Mixed mode stress intensity factors for deflected and inclined surface cracks in finite-thickness plates. *Engineering fracture mechanics*, 71(7-8):1059–1079, 2004.

[9] A.O. Ayhan and U. Yücel. Stress intensity factor equations for mixed-mode surface and corner cracks in finite-thickness plates subjected to tension loads. *International Journal of Pressure Vessels and Piping*, 88(5-7):181–188, 2011.

[10] R. Baptista, J. Marques, and V. Infante. Algorithm for automatic fatigue crack growth simulation on welded high strength steels. *Frattura ed Integrità Strutturale*, 13(48):257–268, 2019.

[11] P. Beld. Improvements for the prediction of the fatigue life of the deck plate in orthotropic steel decks. *Delft University of Technology*, 2019. Msc thesis.

[12] D. Broekaart. Modelling crack propagation using xfem. *Simulia*, 2017.

[13] BSI. Guide to methods for assessing the acceptability of flaws in metallic structures. *The British Standards Institution*, 2015. BS 7910:2013+A1:2015.

[14] Fracture Analysis Consultants. Franc3d reference manual. 2019. Version 7.4.

[15] M. Creager and P.C. Paris. Elastic field equations for blunt cracks with reference to stress corrosion cracking. *International journal of fracture mechanics*, 3(4):247–252, 1967.

[16] A.M.P. de Jesus, R. Matos, B.F.C. Fontoura, C. Rebelo, L.S. da Silva, and M. Veljkovic. A comparison of the fatigue behavior between s355 and s690 steel grades. *Journal of constructional steel research*, 79:140–150, 2012.

[17] H. Dirik and T. Yalçinkaya. Fatigue crack growth under variable amplitude loading through xfem. *Procedia Structural Integrity*, 2:3073–3080, 2016.

[18] F. Erdogan and M. Ratwani. Fatigue and fracture of cylindrical shells containing a circumferential crack. *International Journal of Fracture Mechanics*, 6(4):379–392, 1970.

[19] ESTEC. Esacrack user's manual. *European Space Research and Technology Centre (ESTEC)*, 2000.

[20] M. Fleming, Y.A. Chu, B. Moran, and T. Belytschko. Enriched element-free galerkin methods for crack tip fields. *International journal for numerical methods in engineering*, 40(8):1483–1504, 1997.

[21] R.G. Forman. Study of fatigue crack initiation from flaws using fracture mechanics theory. *Engineering Fracture Mechanics*, 4(2):333–345, 1972.

[22] V.F. González-Albuixech, E. Giner, J.E. Tarancón, F.J. Fuenmayor, and A. Gravouil. Domain integral formulation for 3-d curved and non-planar cracks with the extended finite element method. *Computer methods in applied mechanics and engineering*, 264:129–144, 2013.

[23] P. Gupta, C.A. Duarte, and A. Dhankhar. Accuracy and robustness of stress intensity factor extraction methods for the generalized/extended finite element method. *Engineering Fracture Mechanics*, 179: 120–153, 2017.

[24] M.A. Hirt and J.P. Lebet. Steel bridges: conceptual and structural design of steel and steel-concrete composite bridges. *Epfl Press*, 2013.

[25] C. Huang, T. Chen, and S. Feng. Finite element analysis of fatigue crack growth in cfrp-repaired four-point bend specimens. *Engineering Structures*, 183:398–407, 2019.

[26] M. Janssen and J. Zuidema. Fracture mechanics. *VSSD*, 2006.

[27] F.B.P. de Jong. Overview fatigue phenomenon in orthotropic bridge decks in the netherlands. *Delft University of Technology*, 2004.

[28] J. Kim, H. Lee, Y. Kim, and Y. Kim. The mesh density effect on stress intensity factor calculation using abaqus xfem. *Journal of Mechanical Science and Technology*, 33(10):4909–4916, 2019.

[29] M.H. Kolstein. Fatigue classification of welded joints in orthotropic steel bridge decks. *Delft University of Technology*, 2007. Doctoral thesis.

[30] M. Kuna. Finite elements in fracture mechanics. *Springer*, 10(1007), 2013.

[31] M. Levén. Stationary 3d crack analysis with abaqus xfem for integrity assessment of subsea equipment. *Chalmers University of Technology*, 2012.

[32] J. Maljaars, F. van Dooren, and H. Kolstein. Fatigue assessment for deck plates in orthotropic bridge decks. *Steel Construction*, 5(2):93–100, 2012.

[33] E. Martínez-Pañeda, S. Natarajan, and S. Bordas. Gradient plasticity crack tip characterization by means of the extended finite element method. *Computational Mechanics*, 59(5):831–842, 2017.

[34] NEN-EN. Design of steel structures - material toughness and through thickness properties. *Nederlands Normalisatie-instituut*, 2011. NEN-EN 1993-1-10+C2.

[35] NEN-EN. Design of steel structures - stalen bruggen. *Nederlands Normalisatie-instituut*, 2011. NEN-EN 1993-2+C1/NB nl.

[36] NEN-EN. Design of steel structures - fatigue. *Nederlands Normalisatie-instituut*, 2012. NEN-EN 1993-1-9+C2.

[37] A. Nussbaumer, L. Borges, and L. Davaine. Fatigue design of steel and composite structures. *ECCS Eurocode Design Manuals*, 2011.

[38] C. Obbink-Huizer. Modelling a crack using abaqus. *Simulia*, 2017.

[39] P. Paris and F. Erdogan. A critical analysis of crack propagation laws. *Journal of basic engineering*, 85(4): 528–533, 1963.

[40] J.H. Park and G.P. Nikishkov. Growth simulation for 3d surface and through-thickness cracks using sgbem-fem alternating method. *Journal of mechanical science and technology*, 25(9):2335, 2011.

[41] G. Qian, V.F. González-Albuixech, M. Niffenegger, and E. Giner. Comparison of ki calculation methods. *Engineering Fracture Mechanics*, 156:52–67, 2016.

[42] D. Roylance. Introduction to fracture mechanics. *Massachusetts Institute of Technology, Cambridge*, 1, 2001.

[43] J. Schijve. Fatigue of structures and materials. *Springer Science & Business Media*, 2001.

[44] P.J.G. Schreurs. Fracture mechanics. *Eindhoven University of Technology*, 2011.

[45] J. Shi, D. Chopp, J. Lua, N. Sukumar, and T. Belytschko. Abaqus implementation of extended finite element method using a level set representation for three-dimensional fatigue crack growth and life predictions. *Engineering Fracture Mechanics*, 77(14):2840–2863, 2010.

[46] J.C. Sobotka and R.C. McClung. Automatic 3d crack placement using the python api in abaqus cae. *Southwest Research Institute*, 2002.

[47] K. Spyridoni. Prediction of residual stress of an orthotropic plate due to welding. *Delft University of Technology*, 2019. MSc thesis.

[48] C.T. Sun and Z.-H. Jin. Fracture mechanics. *Academic Press*, 2012.

[49] W. Wu. Out-of-plane fatigue loading tests for rib-to-crossbeam welded joint. *Delft University of Technology*, 2020.

[50] W. Wu, H. Kolstein, M. Veljkovic, R. Pijpers, and J. Vorstenbosch-Krabbe. Fatigue behaviour of the closed rib to deck and crossbeam joint in a newly designed orthotropic bridge deck. 2017.

[51] W. Wu, H. Kolstein, and M. Veljkovic. Stress intensity factors of the rib-to-deck welded joint at the crossbeam conjunction in osds. *Procedia Structural Integrity*, 13:2017–2023, 2018.

[52] W. Wu, H. Kolstein, and M. Veljkovic. Fatigue resistance of rib-to-deck welded joint in osds, analyzed by fracture mechanics. *Journal of Constructional Steel Research*, 162:105700, 2019.

[53] W. Wu, H. Kolstein, and M. Veljkovic. Fatigue resistance of rib-to-deck welded joint in osds, analyzed by fracture mechanics. *Journal of Constructional Steel Research*, 162, 2019.

[54] U. Zerbst, R.A. Ainsworth, H.Th. Beier, H. Pisarski, Z.L. Zhang, K. Nikbin, T. Nitschke-Pagel, S. Münstermann, P. Kucharczyk, and D. Klingbeil. Review on fracture and crack propagation in weldments–a fracture mechanics perspective. *Engineering fracture mechanics*, 132:200–276, 2014.

A

Asymmetrical Four Point Bending Specimen

This annex contains information about the asymmetrical four point bending specimen which has been treated in chapter 3. Firstly, the Python scripts that where used to model crack propagation are presented. Subsequently the illustrations, or projections made in Abaqus are presented. These may help to visualise the process made in the calculations.

A.1. Python script

A python script has been composed as has been explained in section 3.2.3. The script is divided in two separate subscripts, the main script and the output script. The main script contains the design for the crack, meshing the structure and assigning a job. The output script controls the subtraction of the desired results from the calculations. Furthermore, illustrations are made through the output script. At the end of the main script, a command is given to run the output script. Therefore, the scripts run automatically.

A.1.1. Main script

```
# -*- coding: mbcs -*-
# Do not delete the following import lines
from abaqus import *
from abaqusConstants import *
import __main__

import section
import regionToolset
import displayGroupMdbToolset as dgm
import part
import material
import assembly
import step
import interaction
import load
import mesh
import optimization
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior
import numpy as np


# ------------------------------------------------------------------------------------------------------
        # Input parameters
Number_models               = 11    # Can take maximum of 23 for the moment
```

```
Crack_increment_size        = 3
box_size                    = 1.5   # Should be smaller than two times the Crack_increment_size
elements_box_sides          = 17    # Choose uneven value for crack tip in middle of an element
Contour_numbers             = 7     # Should be smaller than elements_box_sides/2;
N_sol_percontour            = 8     # Choose 1 more than elements in thickness direction
# ------------------------------------------------------------------------------------------------------------


        # Define arrays
K1_array = np.zeros(Number_models, dtype=object)
K2_array = np.zeros(Number_models, dtype=object)
K3_array = np.zeros(Number_models, dtype=object);  Keff_array = np.zeros(Number_models, dtype=int)
N_array_paris = np.zeros(Number_models-1, dtype=int); N_tot_paris = 0
angle_array = np.zeros(Number_models + 1, dtype=object)         # angle with respect to initial crack direction
calculated_angle_array = np.zeros(Number_models, dtype=object)  # angle with respect to crack incerement direction
x_coordinates = np.zeros(Number_models+2, dtype = object); x_mid_loc = np.zeros(Number_models, dtype = object)
y_coordinates = np.zeros(Number_models+2, dtype = object); y_mid_loc = np.zeros(Number_models, dtype = object)
x_mid_glob = np.zeros(Number_models, dtype = object); y_mid_glob = np.zeros(Number_models, dtype = object)

        # Define initial crack coordinates
x_coordinates[0] = 0;   x_coordinates[1] = 0
y_coordinates[0] = 0;   y_coordinates[1] = 22

            # Setting coordinates instead of masks
session.journalOptions.setValues(replayGeometry=COORDINATE, recoverGeometry=COORDINATE)

            # Loop for all the models to be created
for i in range (Number_models):

            # Open cae file
    openMdb(
        pathName='C:/temp/Four_point_specimen/Different_mesh/Four_point_specimen_noinitialcrack.cae')
    session.viewports['Viewport: 1'].setValues(displayedObject=None)
    p = mdb.models['Model-1'].parts['Loading_block_1']
    session.viewports['Viewport: 1'].setValues(displayedObject=p)


        # ---------------------------------------------------------
        #               CREATING BOX FOR CRACK TIP REFINEMENT
        # ---------------------------------------------------------
            # Go to sketch plane view
    p = mdb.models['Model-1'].parts['Sheet']
    session.viewports['Viewport: 1'].setValues(displayedObject=p)
    p = mdb.models['Model-1'].parts['Sheet']
    f, e, d = p.faces, p.edges, p.datums
    t = p.MakeSketchTransform(sketchPlane=f.findAt(coordinates=(226.666667,
        33.333333, 8)), sketchUpEdge=e.findAt(coordinates=(250.0, 12.5, 8)),
        sketchPlaneSide=SIDE1, origin=(220.0, 25.0, 8))
    s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
        sheetSize=1007.17, gridSpacing=25.17, transform=t)
    g, v, d1, c = s.geometry, s.vertices, s.dimensions, s.constraints
    s.setPrimaryObject(option=SUPERIMPOSE)
    p = mdb.models['Model-1'].parts['Sheet']
    p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)

            # Create, move and rotate the square
    s.rectangle(point1=(-float(box_size)/2, -float(box_size)/2), point2=(float(box_size)/2, float(box_size)/2))
    x_cr_co = x_coordinates[i+1] + 10; y_cr_co = y_coordinates[i+1] - 29
    s.move(vector=(x_cr_co, y_cr_co), objectList=(g.findAt((-float(box_size)/2, 0)), g.findAt((0,
        float(box_size)/2)), g.findAt((float(box_size)/2, 0)), g.findAt((0, -float(box_size)/2))))
    Rotateangle = float(angle_array[i] * 180) / np.pi
    s.rotate(centerPoint=(x_cr_co, y_cr_co), angle=Rotateangle, objectList=(g.findAt((x_cr_co-float(box_size)/2,
        y_cr_co)), g.findAt((x_cr_co, y_cr_co+float(box_size)/2)), g.findAt((x_cr_co+float(box_size)/2, y_cr_co)),
        g.findAt((x_cr_co, y_cr_co-float(box_size)/2))))

    p = mdb.models['Model-1'].parts['Sheet']
    f = p.faces
    pickedFaces = f.findAt(((226.666667, 33.333333, 8), ))
    e1, d2 = p.edges, p.datums
    p.PartitionFaceBySketch(sketchUpEdge=e1.findAt(coordinates=(250.0, 12.5, 8)),
        faces=pickedFaces, sketch=s)
```

```
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']

        # Sweep rectangle through thickness
x_tip = x_coordinates[i+1] + 230
y_tip = y_coordinates[i+1] - 4
x1_left = (float(box_size)/2) * np.cos(angle_array[i]); x1_right = x1_left
y1_left = (float(box_size)/2) * np.sin(angle_array[i]); y1_right = y1_left
x2_top = y1_left; x2_bottom = x2_top
y2_top = x1_left; y2_bottom = y2_top
x_left = x_tip - x1_left; x_top = x_tip - x2_top; x_right = x_tip + x1_right; x_bottom = x_tip + x2_bottom
y_left = y_tip - y1_left; y_top = y_tip + y2_top; y_right = y_tip + y1_right; y_bottom = y_tip - y2_bottom


p = mdb.models['Model-1'].parts['Sheet']
c = p.cells
pickedCells = c.findAt(((226, 16, 0), ))
e = p.edges
pickedEdges =(e.findAt(coordinates=(x_left, y_left, 8)), e.findAt(
    coordinates=(x_top, y_top, 8)), e.findAt(coordinates=(x_right, y_right,
    8)), e.findAt(coordinates=(x_bottom, y_bottom, 8)))
p.PartitionCellBySweepEdge(sweepPath=e.findAt(coordinates=(500.0, 60.0, 2.0)),
    cells=pickedCells, edges=pickedEdges)


    # ------------------------------------------------------
    #               CREATING CRACK GEOMETRY
    # ------------------------------------------------------
        # Go to sketch plane
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=200.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)

        # Sketch geometry of crack
for tt in range(i+1):
    s.Line(point1=(x_coordinates[tt], y_coordinates[tt]), point2=(x_coordinates[tt+1], y_coordinates[tt+1]))

        # Create the crack as a new part
p = mdb.models['Model-1'].Part(name='Crack_geometry', dimensionality=THREE_D,
    type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['Crack_geometry']
p.BaseShellExtrude(sketch=s, depth=16.0)
s.unsetPrimaryObject()
p = mdb.models['Model-1'].parts['Crack_geometry']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
del mdb.models['Model-1'].sketches['__profile__']

        # Create instance in assambly module
a = mdb.models['Model-1'].rootAssembly
session.viewports['Viewport: 1'].setValues(displayedObject=a)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
a = mdb.models['Model-1'].rootAssembly
p = mdb.models['Model-1'].parts['Crack_geometry']
a.Instance(name='Crack_geometry-1', part=p, dependent=OFF)
p1 = a.instances['Crack_geometry-1']
p1.translate(vector=(501.6, 0.0, 0.0))
session.viewports['Viewport: 1'].view.fitView()

        # Define local and global coordinates for crack parts at x/y mid points
for tt in range (i+1):
    x_mid_loc[tt] = np.mean([x_coordinates[tt], x_coordinates[tt+1]])
    y_mid_loc[tt] = np.mean([y_coordinates[tt], y_coordinates[tt+1]])
x_mid_glob = x_mid_loc + 501.6
y_mid_glob = y_mid_loc

        # Create set for the crack geometry
a = mdb.models['Model-1'].rootAssembly
f1 = a.instances['Crack_geometry-1'].faces
```

```
    ##################################################################
    if i == 0: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ))
    if i == 1: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ))
    if i == 2: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 3: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 4: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 5: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 6: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 7: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 8: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 9: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 10:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 11:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 12:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 13:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 14:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 15:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 16:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 17:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 18:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 19:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 20:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 21:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    if i == 22:faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10), ), ((x_mid_glob[1], y_mid_glob[1], 10), ), ((x_mi
    ##################################################################

        # use these lines in online python to get all the if statements shown above
    #begin_text = 'if i == number: faces1 = f1.findAt(((x_mid_glob[0], y_mid_glob[0], 10)'
    #add_text = ''
    #end_text = ', ))'
    #tot_text = begin_text.replace('number', '0') + add_text + end_text
    #print(tot_text)
    #for tt in range (0, i):
    #    add_text = add_text + ', ), ((x_mid_glob['+str(tt+1)+'], y_mid_glob['+str(tt+1)+'], 10)'
    #    tot_text = begin_text.replace('number', str(tt+1)) + add_text + end_text
    #    print(tot_text)

    a.Set(faces=faces1, name='Crack_geometry_set')

            # Place crack geometry at right position
    a1 = mdb.models['Model-1'].rootAssembly
    a1.translate(instanceList=('Crack_geometry-1', ), vector=(-1.6, -4, -4))
    a1 = mdb.models['Model-1'].rootAssembly
    a1.translate(instanceList=('Crack_geometry-1', ), vector=(-270, 0, 0))



        # ----------------------------------------------------
        #               CREATING CRACK
        # ----------------------------------------------------
            # Select entire sheet as crack domain and Crack_geometry_set as crack location
    a = mdb.models['Model-1'].rootAssembly
    c1 = a.instances['Sheet-1'].cells
    cells1 = c1.findAt(((x_tip-0.1, y_tip+0.2, 0.0), ), ((210, 10.333333,
        0.0), ), ((166.666667, 60.0, 5.333333), ))
#    cells1 = c1.findAt(((210, 10.333333,
#         0.0), ), ((166.666667, 60.0, 5.333333), )) #For no mesh alignment at crack tip
    crackDomain = regionToolset.Region(cells=cells1)
    a = mdb.models['Model-1'].rootAssembly
    crackLocation = a.sets['Crack_geometry_set']
    a = mdb.models['Model-1'].rootAssembly
    a.engineeringFeatures.XFEMCrack(name='XFEM_crack', crackDomain=crackDomain,
        allowCrackGrowth=False, crackLocation=crackLocation)



        # ----------------------------------------------------
        #               CHANGE OUTPUT
        # ----------------------------------------------------
```

```
            # Change field output
    session.viewports['Viewport: 1'].assemblyDisplay.setValues(interactions=OFF,
        constraints=OFF, connectors=OFF, engineeringFeatures=OFF,
        adaptiveMeshConstraints=ON)
    mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(variables=(
        'S', 'U', 'RF', 'CF', 'CSTRESS', 'CDISP',
        'PHILSM', 'PSILSM', 'STATUSXFEM'))

            # Change history output
    mdb.models['Model-1'].historyOutputRequests['H-Output-1'].setValues(
        contourIntegral='XFEM_crack', sectionPoints=DEFAULT, rebar=EXCLUDE,
        numberOfContours=7)

            # unlock the following to change crack initiation criteria (MTS or KII0)
#    mdb.models['Model-1'].historyOutputRequests['H-Output-1'].setValues(
#        kFactorDirection=KII0)


        # ----------------------------------------------------
        #               ASSIGNING MESH AND JOB
        # ----------------------------------------------------
            # Go to mesh module
    session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=ON,
        adaptiveMeshConstraints=OFF)
    session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
        meshTechnique=ON)

            # Assign mesh controls to crack tip box
    a = mdb.models['Model-1'].rootAssembly
    c1 = a.instances['Sheet-1'].cells
    pickedRegions = c1.findAt(((x_tip + 0.1, y_tip + 0.2, 0), ))
    a.setMeshControls(regions=pickedRegions, technique=SWEEP,
        algorithm=ADVANCING_FRONT)

            # Local seed the crack tip box
    a = mdb.models['Model-1'].rootAssembly
    e1 = a.instances['Sheet-1'].edges
    pickedEdges = e1.findAt(((x_left, y_left, 0), ), ((x_left,
        y_left, 8), ), ((x_top, y_top, 0), ), ((x_top,
        y_top, 8), ), ((x_right, y_right, 0), ), ((x_right,
        y_right, 8), ), ((x_bottom, y_bottom, 0), ), ((x_bottom,
        y_bottom, 8), ))
    a.seedEdgeByNumber(edges=pickedEdges, number=elements_box_sides, constraint=FINER)

            # Mesh the structure
    a = mdb.models['Model-1'].rootAssembly
    partInstances =(a.instances['Sheet-1'], )
    a.generateMesh(regions=partInstances)

            # Create a job
    session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=OFF)
    session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
        meshTechnique=OFF)
    mdb.Job(name='Job-'+str(i+1), model='Model-1', description='', type=ANALYSIS,
        atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
        memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
        explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF,
        modelPrint=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine='',
        scratch='', resultsFormat=ODB, multiprocessingMode=DEFAULT, numCpus=1,
        numGPUs=0)

            # Submit the job
    mdb.jobs['Job-'+str(i+1)].submit(consistencyChecking=OFF)
    mdb.jobs['Job-'+str(i+1)].waitForCompletion()

            # Run the output script
    execfile('C:/temp/Four_point_specimen/Different_mesh/Tryout_General_Output_script1.py', __main__.__dict__)
```

```
# End of script
```

A.1.2. Output script

```
from abaqus import *
from abaqusConstants import *
session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=31.0,
    height=106.430557250977)
session.viewports['Viewport: 1'].makeCurrent()
session.viewports['Viewport: 1'].maximize()
from caeModules import *
from driverUtils import executeOnCaeStartup


        # Open odb file (output database)
o3 = session.openOdb(name='C:/temp/Job-'+str(i+1)+'.odb')
session.viewports['Viewport: 1'].setValues(displayedObject=o3)

        # Safe SIF to XYData
odb = session.odbs['C:/temp/Job-'+str(i+1)+'.odb']
for k in range (N_sol_percontour):
    for kk in range (3):
        for kkk in range (Contour_numbers):
            session.XYDataFromHistory(
            name='K'+str(kk+1)+' at XFEM_CRACK_XFEM_'+str(k+1)+'_Contour_'+str(kkk+1)+
                ' ELSET  ALL ELEMENTS-1', odb=odb,
            outputVariableName='Stress intensity factor K'+str(kk+1)+': K'+str(kk+1)+
                ' at XFEM_CRACK_XFEM_'+str(k+1)+'_Contour_'+str(kkk+1)+' in ELSET  ALL ELEMENTS',
            __linkedVpName__='Viewport: 1')

        # Defining the SIFs and angle of crack propagation
K1_array_percontour = np.zeros(Contour_numbers, dtype=object)
K2_array_percontour = np.zeros(Contour_numbers, dtype=object)
K3_array_percontour = np.zeros(Contour_numbers, dtype=object)

for u in range (Contour_numbers):
    aaa = np.zeros(N_sol_percontour, dtype=object); bbb = np.zeros(N_sol_percontour, dtype=object)
    ccc = np.zeros(N_sol_percontour, dtype=object)
    for uu in range (N_sol_percontour):
        aaa[uu] = ((session.xyDataObjects['K1 at XFEM_CRACK_XFEM_'+str(uu+1)+
            '_Contour_'+str(u+1)+' ELSET  ALL ELEMENTS-1'])[0])[1]
        bbb[uu] = ((session.xyDataObjects['K2 at XFEM_CRACK_XFEM_'+str(uu+1)+
            '_Contour_'+str(u+1)+' ELSET  ALL ELEMENTS-1'])[0])[1]
        ccc[uu] = ((session.xyDataObjects['K3 at XFEM_CRACK_XFEM_'+str(uu+1)+
            '_Contour_'+str(u+1)+' ELSET  ALL ELEMENTS-1'])[0])[1]
    K1_array_percontour[u] = np.mean(aaa)
    K2_array_percontour[u] = np.mean(bbb)
    K3_array_percontour[u] = np.mean(ccc)

K1 = np.mean(K1_array_percontour); K1_array[i] = K1
K2 = np.mean(K2_array_percontour); K2_array[i] = K2
K3 = np.mean(K3_array_percontour); K3_array[i] = K3

        # Unlock the following 10 lines for not taking into account the first contour integral
K1_array_percontour_nofirst = np.zeros(Contour_numbers-1, dtype=object)
K2_array_percontour_nofirst = np.zeros(Contour_numbers-1, dtype=object)
K3_array_percontour_nofirst = np.zeros(Contour_numbers-1, dtype=object)
for tt in range(Contour_numbers-1):
    K1_array_percontour_nofirst[tt] = K1_array_percontour[tt+1]
    K2_array_percontour_nofirst[tt] = K2_array_percontour[tt+1]
    K3_array_percontour_nofirst[tt] = K3_array_percontour[tt+1]
K1 = np.mean(K1_array_percontour_nofirst); K1_array[i] = K1
K2 = np.mean(K2_array_percontour_nofirst); K2_array[i] = K2
K3 = np.mean(K3_array_percontour_nofirst); K3_array[i] = K3

        # Calculate angles and number of cycles
angle = -2* np.arctan((K1 - np.sqrt(K1**2 + 8*K2**2))/(4*K2))
calculated_angle_array[i] = angle
Keff_array[i] = (K1**4 + 8 * (K2**4))**(0.25)
if i < Number_models-1:
    N_array_paris[i] = Crack_increment_size / (3.98*(10**(-13)) * (Keff_array[i])**(2.88))
```

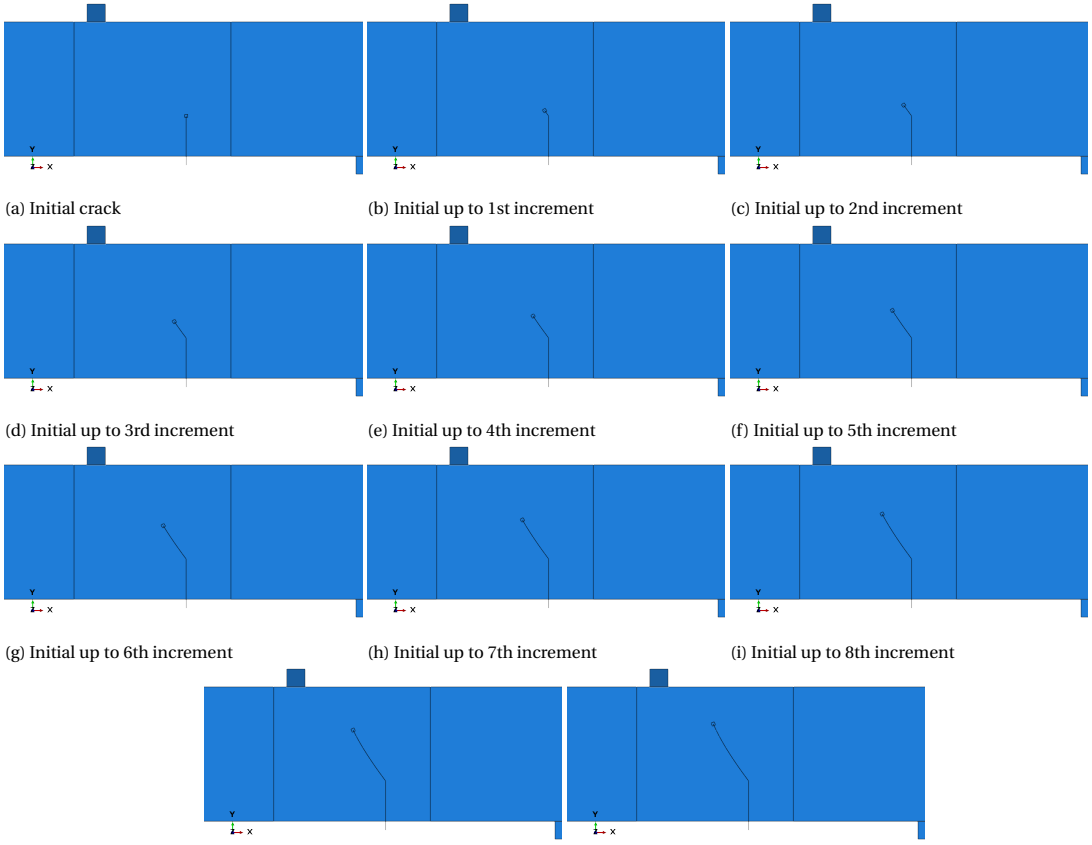
```
    N_tot_paris = N_tot_paris + N_array_paris[i]
if i == 0:
    angle_array[i+1] = angle
elif i > 0:
    angle_array[i+1] = angle + angle_array[i]

        # Calculating new x and y coordinates
x_change = -1 * Crack_increment_size * np.sin(angle_array[i+1])
y_change = Crack_increment_size * np.cos(angle_array[i+1])
x_coordinates[i+2] = x_coordinates[i+1] + x_change
y_coordinates[i+2] = y_coordinates[i+1] + y_change


        # ---------------------------------------------------------
        #               SAVE PICURES IN WORKING DIRECTORY
        # ---------------------------------------------------------
        # Set viewport to assembly module
a = mdb.models['Model-1'].rootAssembly
session.viewports['Viewport: 1'].setValues(displayedObject=a)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
session.viewports['Viewport: 1'].view.setValues(session.views['Front'])
session.viewports['Viewport: 1'].view.setProjection(projection=PARALLEL)
session.viewports['Viewport: 1'].view.setValues(nearPlane=978.912,
    farPlane=1044.31, width=142.219, height=81.6956, viewOffsetX=-21.1621,
    viewOffsetY=-1.53111)

        # Save pictures of assembly module front view and iso-view
session.printToFile(fileName='Image_'+ str(i+1), format=PNG, canvasObjects=(
    session.viewports['Viewport: 1'], ))
session.viewports['Viewport: 1'].view.setValues(session.views['Iso'])
session.viewports['Viewport: 1'].view.setValues(nearPlane=814.874,
    farPlane=1208.35, width=139.5, height=80.1337, cameraPosition=(816.429,
    621.768, 597.967), cameraTarget=(232.375, 37.7129, 13.9125))
session.printToFile(fileName='Image_'+ str(i+1) +'_iso', format=PNG, canvasObjects=(
    session.viewports['Viewport: 1'], ))

        # Set viewport to mesh module and zoom
session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=ON)
session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
    meshTechnique=ON)
session.viewports['Viewport: 1'].view.setValues(session.views['Front'])

if x_coordinates[i] > -20:
    session.viewports['Viewport: 1'].view.setValues(nearPlane=1038.44,
        farPlane=1057.65, width=26.6609, height=14.4271, cameraPosition=(221.182,
        24.225, 1052.05), cameraTarget=(221.182, 24.225, 4))
elif x_coordinates[i] > -60:
    session.viewports['Viewport: 1'].view.setValues(nearPlane=1038.44,
        farPlane=1057.65, width=26.6609, height=14.4271, cameraPosition=(221.182,
        24.225, 1052.05), cameraTarget=(221.182, 24.225, 4))
    session.viewports['Viewport: 1'].view.setValues(cameraPosition=(207.979,
        35.7619, 1052.05), cameraTarget=(207.979, 35.7619, 4))
else:
    session.viewports['Viewport: 1'].view.setValues(nearPlane=1038.44,
        farPlane=1057.65, width=26.6609, height=14.4271, cameraPosition=(221.182,
        24.225, 1052.05), cameraTarget=(221.182, 24.225, 4))
    session.viewports['Viewport: 1'].view.setValues(cameraPosition=(190.109,
        43.6945, 1052.05), cameraTarget=(190.109, 43.6945, 4))

        # Save picture of mesh at cracktip
session.printOptions.setValues(vpDecorations=OFF)
session.printToFile(fileName='Image_'+ str(i+1) +'_mesh_zoomattip', format=PNG,
    canvasObjects=(session.viewports['Viewport: 1'], ))

        # Set viewport to results Mises stresses
session.viewports['Viewport: 1'].setValues(
    displayedObject=session.odbs['C:/temp/Job-'+ str(i+1) +'.odb'])
odb = session.odbs['C:/temp/Job-'+ str(i+1) +'.odb']
session.viewports['Viewport: 1'].setValues(displayedObject=odb)
session.viewports['Viewport: 1'].odbDisplay.display.setValues(plotState=(
```

```
    CONTOURS_ON_DEF, ))
session.viewports['Viewport: 1'].view.setProjection(projection=PARALLEL)
session.viewports['Viewport: 1'].view.setValues(session.views['Front'])
session.viewports['Viewport: 1'].odbDisplay.contourOptions.setValues(
    contourStyle=CONTINUOUS, maxValue=7875.26, minValue=0.18924)
session.viewports['Viewport: 1'].view.setValues(nearPlane=1038.44,
    farPlane=1057.65, width=26.6609, height=14.4271, cameraPosition=(221.182,
    24.225, 1052.05), cameraTarget=(221.182, 24.225, 4))

        # Save zoomed picture of results Mises stresses
session.printToFile(fileName='Image_'+ str(i+1) +'_stresses', format=PNG, canvasObjects=(
    session.viewports['Viewport: 1'], ))


        # ---------------------------------------------------------
        #               SAVE DATA IN TEXT FILES
        # ---------------------------------------------------------
file = open("Data_"+str(i+1)+".txt","w")
file.write("K1 :"); file.write(str(K1)); file.write("      ")
file.write("K2 :"); file.write(str(K2)); file.write("      ")
file.write("K3 :"); file.write(str(K3)); file.write("      ")
file.write("Keff :"); file.write(str(Keff_array[i])); file.write("      ")
file.write("Calculated Angle: "); file.write(str(calculated_angle_array[i])); file.write("      ")
file.write("x_coordinate tip: "); file.write(str(x_coordinates[i+1]-20)); file.write("      ")
file.write("y_coordinate tip: "); file.write(str(y_coordinates[i+1]-4)); file.write("      ")
file.write("K1_percontour :"); file.write(str(K1_array_percontour)); file.write("      ")
file.write("K2_percontour :"); file.write(str(K2_array_percontour)); file.write("      ")
file.write("K1_lastcontour_perelement :"); file.write(str(aaa)); file.write("      ")
file.write("K2_lastcontour_perelement :"); file.write(str(bbb)); file.write("      ")
file.close()


# End of Script
```

A.2. Abaqus projections

Making projections in Abaqus was incorporated in the scripts. These include projections of the crack trajectory in XY- and isometric view and are shown below.

A.2.1. Crack trajectories XY projection



(a) Initial crack (b) Initial up to 1st increment (c) Initial up to 2nd increment

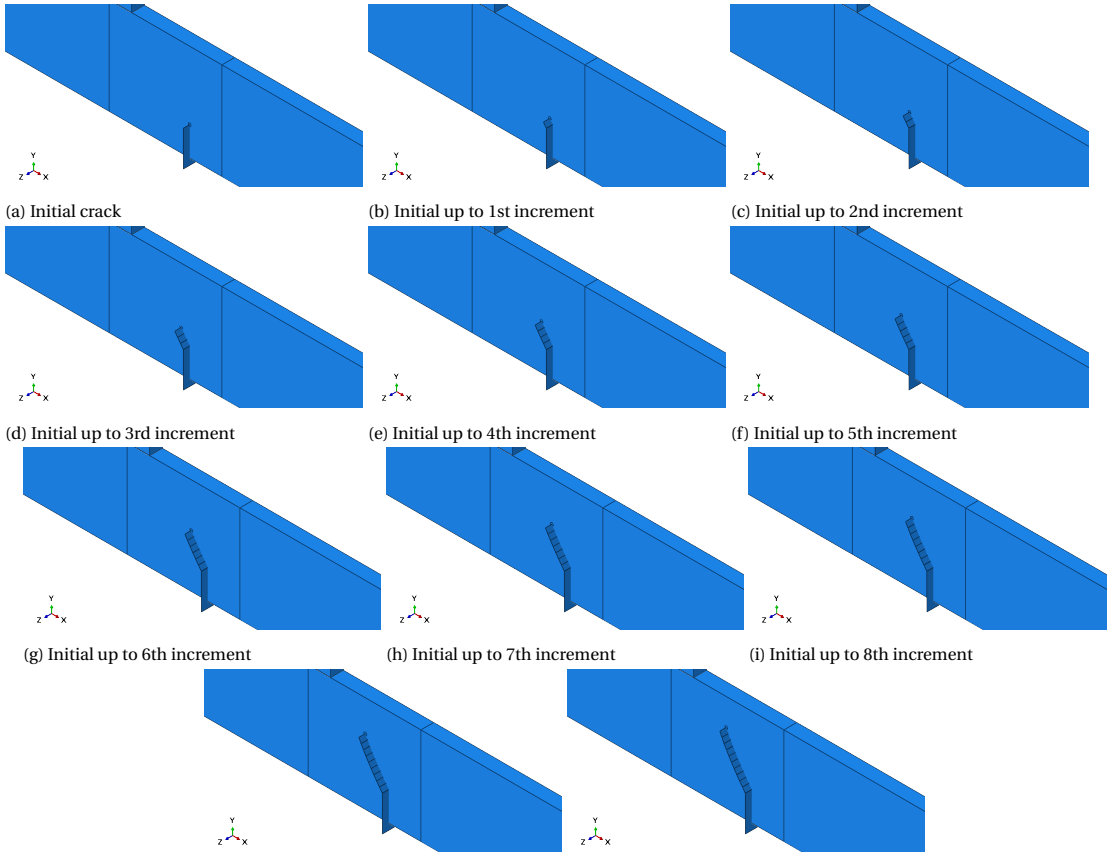
(d) Initial up to 3rd increment (e) Initial up to 4th increment (f) Initial up to 5th increment

(g) Initial up to 6th increment (h) Initial up to 7th increment (i) Initial up to 8th increment

(j) Initial up to 9th increment (k) Initial up to 10th increment

Figure A.1: XY projection of the crack trajectory up to 10th increment

A.2.2. Crack trajectories isometric projection



(a) Initial crack (b) Initial up to 1st increment (c) Initial up to 2nd increment

(d) Initial up to 3rd increment (e) Initial up to 4th increment (f) Initial up to 5th increment

(g) Initial up to 6th increment (h) Initial up to 7th increment (i) Initial up to 8th increment

(j) Initial up to 9th increment (k) Initial up to 10th increment

Figure A.2: Isometric projection of the crack trajectory up to 10th increment

B

Computational Fracture Mechanics using Abaqus

The goal of this study is to gain insight into what modelling techniques are available and what information can be obtained about a structure containing a crack. Abaqus 6.13 is used as a finite element program. Abaqus/Standard, offers two different methods to evaluate contour integrals [1]. These contour integrals are used to deduce the stress intensity factors which help to describe the fatigue life and crack propagation direction. The first method is based on the conventional finite element method (FEM) which shall first be addressed. The second method is based on the extended finite element method (XFEM). Reference for modelling cracks in Abaqus can be found in [1, 2, 4, 12, 38].

B.1. Contour integrals using FEM

In this chapter, contour integral evaluations of the conventional finite element method (FEM) are discussed.

B.1.1. Seams

A seam defines an edge (for 2D) or face (for 3D) where the nodes are duplicated and separated during the analysis. This creates a debonded structure at the location of the seam. A seam can be modelled for both 2D and 3D models. A seam cannot be used for the extended finite element method (XFEM). Independent meshing is needed since the seam will modify the mesh. From a seam in a structure, a stress state can be obtained from a cracked structure. In order to obtain accurate results for the stresses, local meshing is needed around the crack tip using a spider web mesh. If crack properties need to be obtained, a contour integral analysis can be used. Thus, if no crack properties such as stress intensity factor's, J-integral or crack extension direction are needed, only a seam will suffice. This will still give information about the cracked structure as stress and strain states [1].

B.1.2. Contour integrals

Abaqus can evaluate contour integrals for a given crack geometry. These can be computed for both 2D as 3D models. The following can be computed [2]:

- J-integral;
- Ct-integral;
- T-stress;
- Stress intensity factor.

In order to compute any one of these, an opening must be created. This can be modelled using an existing region or by defining a seam. The latter is the more simple approach of the two. Furthermore, the crack front, crack tip (or line) and the crack extension direction must be defined. The output results can be managed using the history output request [2].

B.1.3. Stress intensity factor's

Abaqus calculates the J-integral trough an energy based formula. This J-integral can be related to the stress intensity factors. For only mode I loading (i.e. opening mode), K_I can be deduced from the J-integral value. However, for mixed mode problems, all three stress intensity factors cannot be deduced from the J-Integral (3 unknowns per J) [2]. Abaqus therefore uses an interaction integral to compute the stress intensity factors for a crack under mixed-mode loading. This is only available for linear isotropic and anisotropic materials [4]. According to [4], the stress intensity factor for mode α is calculated through:

$$K_\alpha = 4\pi \mathbf{B}_{\alpha\alpha} J^\alpha_{int} \tag{B.1}$$

Where $\mathbf{B}$ is a diagonal matrix and J_{int} is the interaction integral which is calculated by:

$$J^\alpha_{int} = \lim_{\Gamma \to 0} \int \mathbf{n} \cdot \mathbf{M}^\alpha \cdot \mathbf{q} d\Gamma \tag{B.2}$$

$$M^\alpha = \boldsymbol{\sigma} : \boldsymbol{\epsilon}^\alpha_{aux} \mathbf{I} - \boldsymbol{\sigma} \cdot \frac{\delta u^\alpha}{\delta x_{aux}} - \boldsymbol{\sigma}^\alpha_{aux} \cdot \frac{\delta u}{\delta x} \tag{B.3}$$

It can be noticed from these equations that the stress intensity factor is calculated through integrating '$\sigma\epsilon$' terms, which represents the internal energy.

B.1.4. Crack propagation direction

Abaqus also calculates the crack propagation direction. This is determined from the stress intensity factors of mode I and mode II (K_I and K_{II}). Mode III is not taken into account. One of the following crack propagation criteria can be chosen [2].

- Maximum tangential stress criterion;
- Maximum energy release rate criterion;
- $K_{II} = 0$ criterion.

The maximum engergy release rate criterion chooses the direction that maximizes the energy release rate. Similarly, the $K_{II} = 0$ criterion chooses the direction that makes $K_{II} = 0$. The maximum tangential stress (MTS) criterion used the direction which is perpendicular to the maximum tensile stress. In the MTS-criterion Abaqus uses the following relation to determine the angle [2]:

$$\theta = \arccos\left(\frac{3K^2_{II} + \sqrt{K^4_I + 8K^2_I K^2_{II}}}{K^2_I + 9K^2_{II}} \right) \tag{B.4}$$

where:

$$\theta < 0 \quad \text{if} \quad K_{II} > 0 \qquad \text{and} \qquad \theta > 0 \quad \text{if} \quad K_{II} < 0 \tag{B.5}$$

Abaqus defines several vectors along the crack front. These are shown in the figure below. The vector $\mathbf{t}$ defines the direction along which Abaqus reads the crack front. This defines the order in which the crack front identities are listed. The vector $\mathbf{q}$ is orthogonal to the crack front and vector $\mathbf{n}$ is normal to the crack plane. As a result, vector $\mathbf{q}$ is the cross product of vectors $\mathbf{t}$ and $\mathbf{n}$. Abaqus reads the crack propagation direction around vector $\mathbf{t}$ in direction from $\mathbf{q}$ to $\mathbf{n}$. This should be taken into account when analysing the results as it determines the positive direction of crack propagation [2].

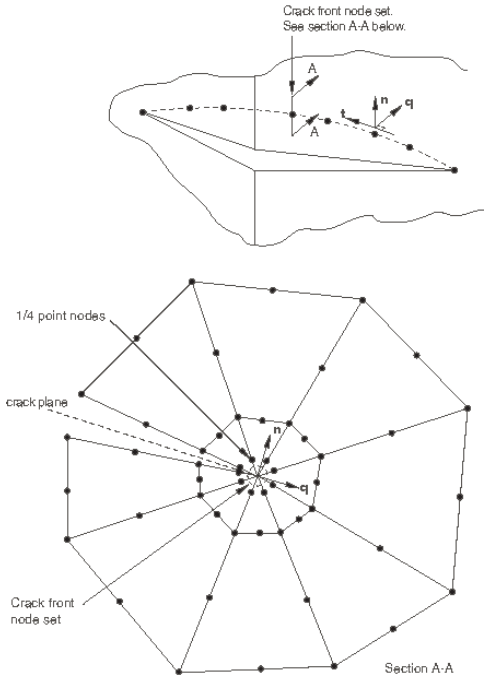


Figure B.1: Abaqus crack orientation [2]

B.1.5. Example - single edge notched plate

As an example, a single edge notched plate will be examined. The theoretical structure is presented in figure B.2. The input parameters are given in table B.1. The applied stress is loaded in the direction to give axial tension creating a mode I crack, as shown in figure B.2.

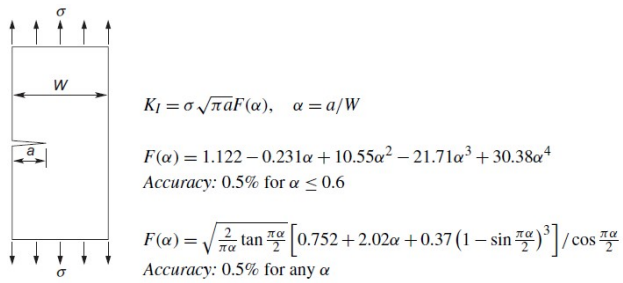


Figure B.2: Single edge notched plate SIF [48]

Parameter	Value	Unit
Width	200	$[mm]$
Height	500	$[mm]$
Thickness	20	$[mm]$
Crack length (a)	100	$[mm]$
Applied stress (σ)	100	$[MPa]$

Table B.1: Input parameters

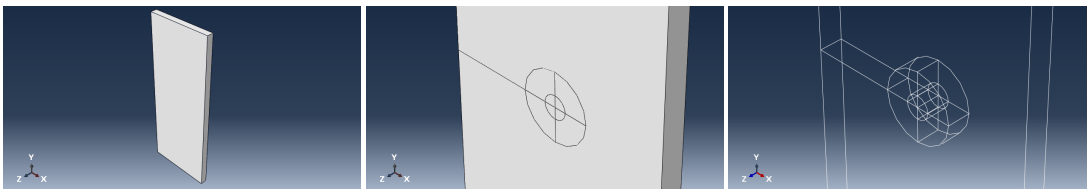
The analytical solution of the stress intensity factor K_I is thus:

$$\alpha = \frac{100}{200} = 0,5 \quad [-] \tag{B.6}$$

$$F = 1,122 - 0,231\alpha + 10,55\alpha^2 - 21,71\alpha^3 + 30,38\alpha^4 = 2,829 \quad [-] \tag{B.7}$$

$$K_I = 100 \cdot \sqrt{\pi \cdot 100} \cdot F = 5014 \quad MPa\sqrt{mm} \tag{B.8}$$

Modelling the geometry
A 3D solid extruded sheet is modelled with the parameters as given in the table above. Partitioning of the front surface is used to make a fine mesh region possible. The goal is to make a spiderweb kind of mesh around the crack tip. This partitioning is swept through the sheets thickness. This is shown in the figures below.



(a) Model of sheet (b) Partitioning of front face (c) Swept partitioned part

Figure B.3: Modelling of the geometry

Material properties and assembly
Elastic isotropic material is chosen with a Young's modulus of 210 GPa and a Poisson's ratio of 0.3. These parameters are chosen to represent steel. When assembling, independent mesh must be chosen so the mesh can be controlled per region.

Boundary conditions
The boundary conditions are modelled as presented in figure B.2. Therefore, a tensile load of 100 MPa is applied on the top and bottom surface of the sheet. This is shown in following figure.

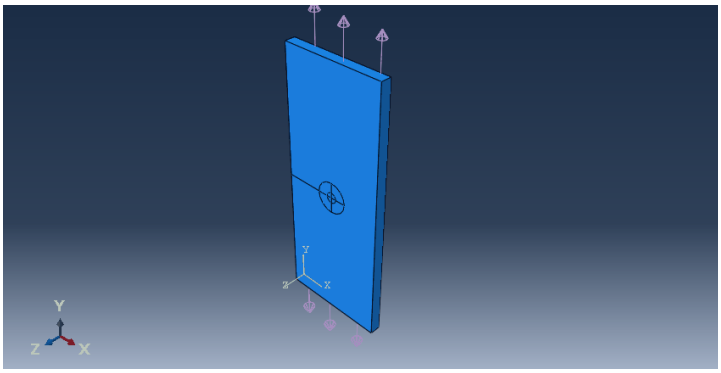
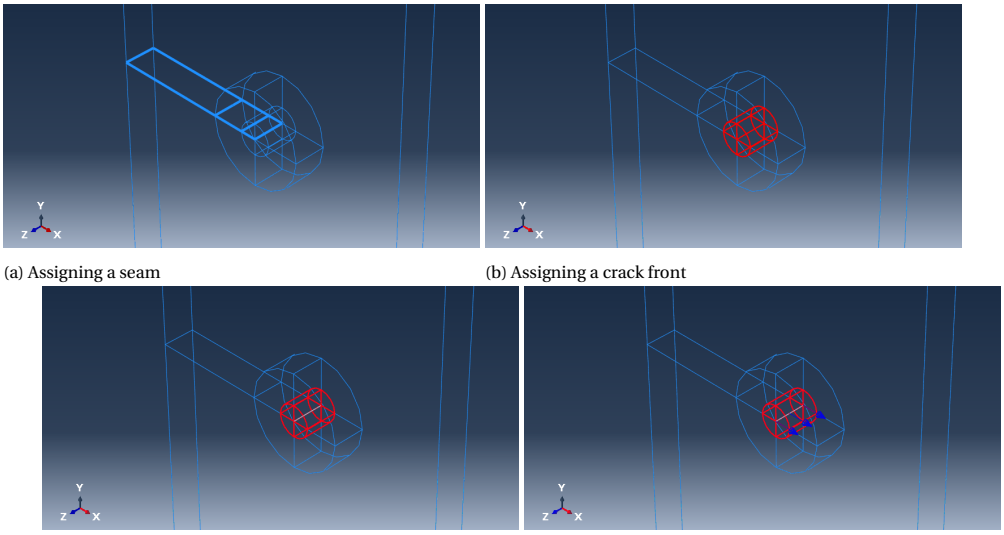


Figure B.4: Boundary conditions on sheet

Defining the crack
Defining a crack can be done in the interaction module of Abaqus. As explained in section B.1, first a seam needs to be created for a crack analysis using FEM. By following 'Special, Crack, Assign Seam' the seam can be modelled. By following 'Special, Crack, Create Crack' and choosing 'Contour integral' a crack can be defined. Then, as explained in B.1, the crack front, crack line and propagation direction must be chosen. These are chosen as showed in the following figures. Afterwards, the singularity can be changed. To have a $\frac{1}{\sqrt{r}}$ singularity as is the case in linear elastic fracture mechanics (see section 2.2.1), one must choose 0.25 for 'midside node parameter' and 'collapsed element side, single node' [2]. to define the propagation direction, q-vectors can be chosen on which 2 nodes must be picked. these nodes will represent the direction in which the crack propagates.



(a) Assigning a seam (b) Assigning a crack front

(c) Assigning crack line (d) Assigning q-vectors

Figure B.5: Modelling of the crack

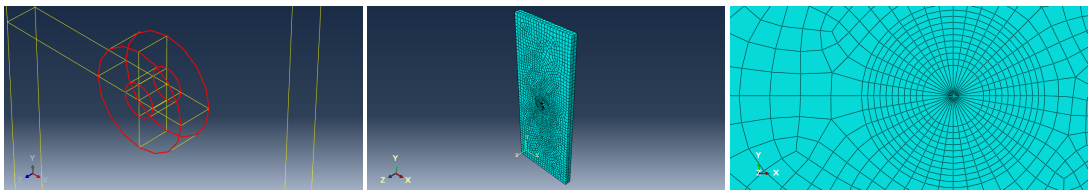
The crack front represents the first contour integral region. Thus from that region outwards the contours are calculated. If we want the contours to be calculated in between the two cylinders, we need to assign the small cylinder (the one around the crack tip) as the crack front.

Request contour integral output

To obtain contour integral output, we need to go to the 'Step' module. Here, one can request for example stress intensity factors as output through the 'History Output Manager'. Choosing a crack initiation criterion will define on which bases the propagation angle will be calculated. This is for now, not of interest since we analyse a stationary crack. 7 contours are requested of type 'Stress intensity factors'.

Meshing the structure

To create a spiderweb kind of mesh, wedge elements are chosen in the smallest cylinder and hexagonal elements in the largest cylinder. This leads to linear C3D6, C3D8R and C3D8R elements for the smallest cylinder, largest cylinder and remainder part respectively. Local seeding is used to control the number of elements around the crack tip. 10 elements are used for each quadrant of the circles. Together with 1 element in radius direction of the smallest circle will lead to 40 wedge elements around the crack tip. A global element size of 10 $[mm]$ is used for the remainder part. The resulting mesh is shown below.



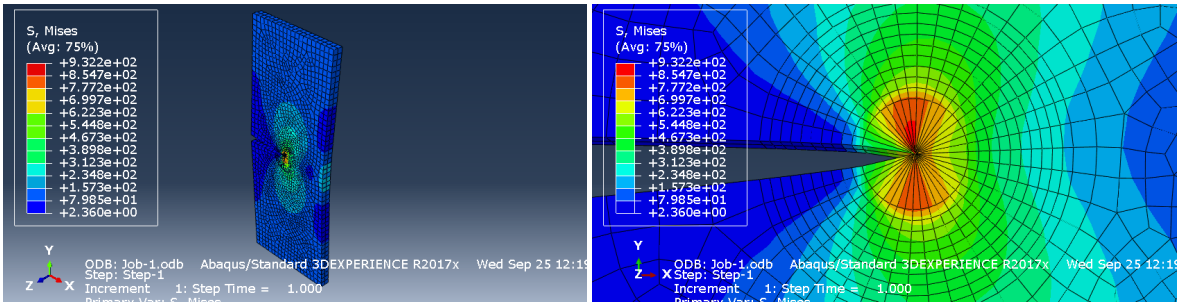
(a) Local seeding of the circles (b) Global view (c) Zoomed at spiderweb mesh

Figure B.6: Meshing the structure

Results

The resulting 'von Mises' stresses are presented below. A typical butterfly shape is seen from the contours of the stresses. The requested contour integral output can be found in in 'Monitor, Data file'. Here, the resulting stress intensity factors are presented for each contour integral. Furthermore, for 2 elements, 3 contour integrals are evaluated along the crack front [2].

The resulting stress intensity factors are presented in the table below. In the table '-5-, -6- and -7-'stand for



(a) Global view

(b) Zoomed at crack tip

Figure B.7: Von Mises stresses in cracked sheet

the contours that are evaluated along the crack front. Thus, -6- are the contour integrals in the middle of the sheet and -5- and -7- are the contour integrals evaluated at the outside of the sheet [2]. The total averaged value of K_I is 4928 $MPa\sqrt{mm}$. The analytical value is 5014 $MPa\sqrt{mm}$ as presented at the beginning of this section. The total error is thus 1,7 % .

Contour number	**K_I [MPa$\sqrt{\text{mm}}$]**			**Averaged**
	-5-	-6-	-7-	
1	4968	4938	4878	4928
2	4967	4939	4877	4928
3	4970	4937	4879	4929
4	4970	4937	4879	4929
5	4971	4937	4878	4929
6	4971	4937	4877	4928
7	4971	4936	4877	4928
				4928

Table B.2: Stress intensity factor output

B.2. Contour integrals using XFEM

XFEM Studies initiation and propagation of a crack along an arbitrary solution-dependent path. No seam needs to be created. XFEM is known to have significantly longer computation time compared to conventional FEM. However, the results are said to be more accurate. Moreover, the modelling of the propagation of a crack is not possible with conventional FEM. XFEM uses an arbitrary, solution-dependent path for the crack to propagate. Therefore, no crack path needs to be defined. This method can study a growing as well as a stationary crack. The, by the user defined, crack region will include any existing cracks and the region in which a crack might initiate and/or propagate [1]. Quadratic elements are not available for XFEM crack analysis [1, 12].

B.2.1. Types of analysis's

For studying initiation and propagation in Abaqus, the user can choose one out of two analyses for the XFEM calculation. These are [1]:

- Traction-separation cohesive behavior;
- Linear elastic fracture mechanics.

In the analysis for LEFM, Abaqus uses the Virtual Crack Closure Technique (VCCT) to calculate the strain energy release rate [1].

B.2.2. Crack initiation

One can choose to select a initial crack or to let Abaqus search for a region where a crack may initiate. One of the following criteria can be chosen [1].

- Maximum principle stress;
- Maximum principle strain.

Choosing the maximum principle stress criterion has the advantage that the crack plane can be perpendicular to the direction of the maximum principle stress. This makes the crack solution dependent [12].

B.2.3. Contour integral output

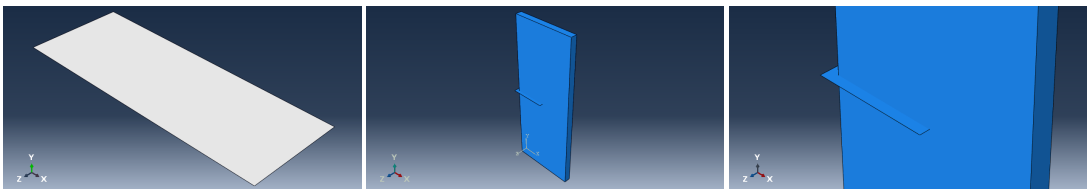
Contour integrals can be generated as output using XFEM. Therefore, SIF's and J-integral values can be obtained. However, this is only possible for a stationary crack [1]. A benefit, by using XFEM, can be that less effort is needed for meshing the structure. Furthermore, the results for SIF might be more accurate compared to the conventional FEM.

B.2.4. Example - single edge notched plate

Similar to the example presented in section B.1.5, a single edge notched plate will be modelled. Now, a XFEM crack will be used. all the parameters stated in table B.1 remain the same. The main difference in modelling is that with XFEM one models a separate part to represent the crack [1].

Creating the crack geometry

As mentioned above, a seperate part is modelled to represent the crack. Here, a 3D extruded shell is used. The length of the shell is chosen 110 mm and it is extruded 40 mm. This ensures that the crack sticks out of the sheet at all 3 sides 10 mm. When assembling the parts, the crack is located at the same position as in section B.1.5. This is shown in the figures below.

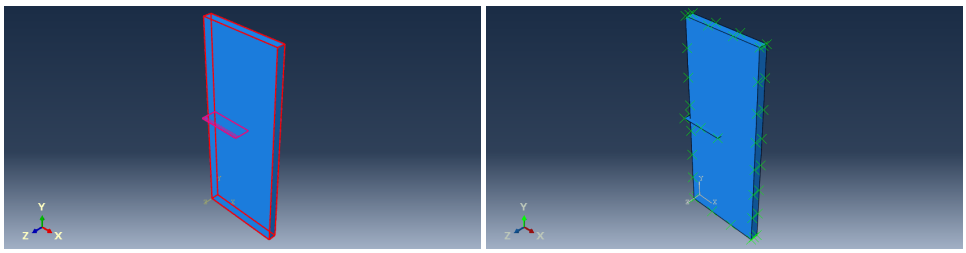


(a) Crack part (b) Assembled global ISO view (c) Assembled zoomed at crack

Figure B.8: Geometry of the crack

Creating the XFEM crack

The crack can be defined in the 'interaction' module. Here one can make the crack by following: 'Special, Crack, Create' and choose 'XFEM'. As mentioned in section B.2, no seam must be created when analysing a XFEM crack. After choosing 'XFEM' one must choose the crack domain. This must contain the region in which the crack exists and (if so) propagates. Here, the entire sheet is chosen. In the menu 'Edit Crack' allow crack growth is checked off and the crack location is chosen as the crack geometry part. The crack location and domain are shown in the figures below.



(a) Crack location (b) Crack domain

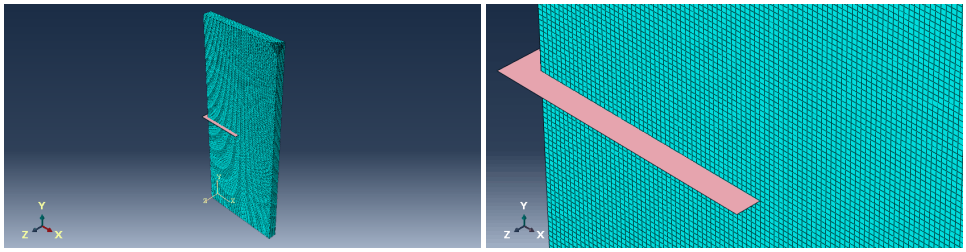
Figure B.9: Defining the crack

Request contour integral output

In the field output request manager, the output variables 'PHILSM, PSILSM and STATUSXFEM' must be checked on for XFEM analysis. Similar as for the conventional FEM analysis, the contour integral output can be requested in the 'History output request manager'. Here 7 contour are chosen of type stress intensity factor (similar to in section B.1).

Meshing the structure

The sheet is meshed with structured hexagonal elements. Element size of 2 mm is chosen to ensure accurate results. As mentioned in section B.2, no quadratic elements can be used for XFEM crack analysis. Therefore, linear elements are used. The elements are of type C3D8R (8 node linear brick elements). The crack part does need to be meshed. In the figures below, the meshed structure is presented. Here one can see that the crack part is not meshed.

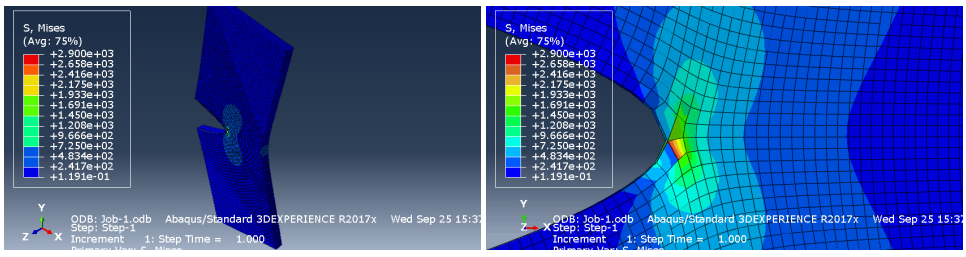


(a) Global ISO view (b) Zoomed at crack

Figure B.10: Meshed model

Results

The resulting 'von Mises' stresses are presented below. Similar as for conventional FEM analysis, the contour integral values can be viewed by following 'Monitor, Data file'. The resulting stress intensity factors are presented for each contour integral. As there are 10 elements in thickness direction, there are 11 contour outputs along the crack front.



(a) Global ISO view (b) Zoomed at crack

Figure B.11: Von Mises stresses

In the table presented below, all the results of the contour integrals are shown. The XFEM number gives the contour integral along the crack front. The average is taken from contours 2-7 since the first contour value deviates significantly from the trend. The total average was 5316 $MPa\sqrt{mm}$ while the analytical value was 5014 $MPa\sqrt{mm}$. This leads to an error of 5,7%. This error is higher than that of the conventional FEM analysis. However, no spiderweb mesh was used here. Moreover, if a more fine mesh is used, more accurate results may be obtained.

XFEM number	Contour							av.
	1	2	3	4	5	6	7	
1	7246	4991	5232	5200	5177	5196	5168	5161
2	7317	4988	5260	5251	5235	5263	5250	5208
3	7456	5153	5411	5396	5388	5419	5403	5362
4	7510	5180	5444	5423	5413	5444	5429	5389
5	7535	5206	5466	5441	5436	5466	5449	5411
6	7542	5206	5469	5444	5436	5466	5450	5412
7	7535	5206	5466	5441	5436	5466	5449	5411
8	7510	5180	5444	5423	5413	5444	5429	5389
9	7456	5153	5411	5396	5388	5419	5403	5362
10	7317	4988	5260	5251	5235	5263	5250	5208
11	7246	4991	5232	5200	5177	5196	5168	5161
								5316

Table B.3: Stress intensity factor output

C

Parameterize Cracks in Abaqus

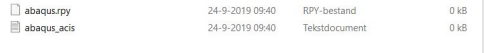
As explained in appendix B, only stationary cracks can give contour integral output in Abaqus. Therefore, if stress intensity factors (a contour integral output) are requested for various crack sizes, various models are needed. A helpful tool for this is the scripting interface of Abaqus. Abaqus uses the python coding language. In this appendix, the methodology of scripting with Abaqus is explained where the focus lies on stationary crack analysis for various crack geometries. More information about scripting in Abaqus can be found in the Abaqus scripting user's guide [3]. At the end of this appendix, an example is given of a script of a single edge notched plate for an XFEM crack analysis.

C.1. Scripting in Abaqus

During this thesis, two different scripting techniques have been found for scripting in Abaqus. One which uses the abaqus.rpy file and one which uses the 'Macro manager' option in Abaqus. Both are briefly explained below. Afterwards, some usefull expressions when scripting with in Abaqus are given.

C.1.1. Scripting techniques

When opening Abaqus, two files are created in the working directory, abaqus.rpy and abaqus_acis. As stated above, Abaqus uses python language for it to run on. This code, while building a model, is saved in the RPY-file (abaqus.rpy). This file can be opened and viewed at any time while building the model. When new commands are given in Abaqus, this RPY-file is updated. In this way, a code for a certain command can easily be extracted to use in an other code. In the figures below, the created files and opening code of the abaqus.rpy file are presented.



(a) Created files after opening Abaqus

```
# -*- coding: mbcs -*-
#
# Abaqus/CAE Release 2017 replay file
# Internal Version: 2016_09_27-23.54.59 126836
# Run by David on Tue Sep 24 09:40:43 2019
#

# from driverUtils import executeOnCaeGraphicsStartup
# executeOnCaeGraphicsStartup()
#: Executing "onCaeGraphicsStartup()" in the site directory ...
from abaqus import *
from abaqusConstants import *
session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=31.0,
    height=100.592597961426)
session.viewports['Viewport: 1'].makeCurrent()
session.viewports['Viewport: 1'].maximize()
from caeModules import *
from driverUtils import executeOnCaeStartup
executeOnCaeStartup()
session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
    referenceRepresentation=ON)
Mdb()
#: A new model database has been created.
#: The model "Model-1" has been created.
session.viewports['Viewport: 1'].setValues(displayedObject=None)
```

(b) Starting code of abaqus.rpy file

Figure C.1: abaqus.rpy file

An other way, is to use 'Macro manager' in Abaqus. This is an option in Abaqus to make a separate RPY-file in which all the commands, in code language, is saved. So, after opening macro manager, the model needs to be build. When this is done, macro manager is closed and the code can be found in the predefined location. This

will be the code for that specific model. The macro manager can be called upon by following: 'File, Macro Manager' in the toolbar menu. In figure C.2 the macro manager option, created file and opening code are presented. In the opening code one can see that a certain list of packages is imported. These are needed to create the file. Thus, when making an own code, these lines can be copied.

(a) Macromanager option

(b) Created file

(c) Opening code

Figure C.2: Macro manager in Abaqus

When the code is finished, Abaqus can be called upon to run this code. This can be done by following: 'File, Run Script' in the toolbar menu.

C.1.2. Useful expressions

In this section, some expressions are presented that may prove useful when scripting in Abaqus. One can try these commands in the command line in Abaqus to see the results. Therefore, not the entire scripts needs to run.

Opening a CEA-file

Opening an existing CEA-file may be useful. The unchanging parameters can be modelled in that file. The script can open this file and, for example, ad a crack to this file. Therefore, the code will be shorter and give a much better overview. The following expression can be used to open an CEA-file:

openMdb(pathName='pathname/filename.cae')

Setting coordinates as parameters

Abaqus uses masks to call upon lines, surfaces and cells [3]. This is usefull within Abaqud itself. However, when making a script for Abaqus this is not always the case. For example, when in a script a partitioning is made on a surface, all the masks numbers change. It might be easier to call upon lines/surfaces/cells with global coordinates instead of masks in the code. this can be done by entering the following command.

session.journalOptions.setValues(replayGeometry=COORDINATE, recoverGeometry=COORDINATE)

Creating and submitting a job

After the model is built by the script, the user may want to automatically create and start a job. To create a job, the following code may be used.

mdb.Job(name='jobname', model='modelname', description='', type=ANALYSIS, memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True, explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine='', scratch='', resultsFormat=ODB, multiprocessingMode=DEFAULT, numCpus=1, numGPUs=0)

After the job is created, you can call upon the job to run. This is done by the following command.

mdb.jobs['jobname'].submit(consistencyChecking=OFF)

Waiting until job is finished

Abaqus automatically keeps running the script. Therefore, if you coded that a job should be submitted, Abaqus submits the job and continues with reading the next lines of the code. If you want Abaqus to wait until the calculation is finished, the following code can be used.

mdb.jobs['jobname'].waitForCompletion()

Reading an other code
If there are multple python files which need to run after each other. The follwing command may be used. This may prove usefull to separate the code in which the model is changed with the one that extracts and analises data from the results.

execfile('pathname/filename.py')

Opening an ODB-file
An ODB- file (output database) is a file that Abaqus creates when a job is completed. Here the results from the calculation are written. This can be opened by use of a python code so that information can be extracted from it.

session.openOdb(name='filename.odb')

C.2. Example - single edge notched plate

Below, a script is presented for a single edge notched plate. A XFEM crack analysis in Abaqus is performed. This script can be runned directly by Abaqus. All input parameters as stated in table B.1 are used for the script.

```
# -*- coding: mbcs -*-
# Do not delete the following import lines
from abaqus import *
from abaqusConstants import *
import __main__

import section
import regionToolset
import displayGroupMdbToolset as dgm
import part
import material
import assembly
import step
import interaction
import load
import mesh
import optimization
import job
import sketch
import visualization
import xyPlot
import displayGroupOdbToolset as dgo
import connectorBehavior


#------------------------------------------------------------------
#            INPUT PARAMETERS
#------------------------------------------------------------------
Width               = 200       #[mm]
Height              = 500       #[mm]
Thickness           = 20        #[mm]
Crack_length        = 100       #[mm]
Applied_stress      = 100       #[MPa]
Youngs_modulus      = 210000    #[MPa]
Poisson_ratio       = 0.3       #[-]


#------------------------------------------------------------------
#            CREATING PARTS
#------------------------------------------------------------------
#           Create sheet
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=200.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
s.rectangle(point1=(0.0, 0.0), point2=(Width, Height))
p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=THREE_D,
    type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['Part-1']
```

```
p.BaseSolidExtrude(sketch=s, depth=Thickness)
s.unsetPrimaryObject()
p = mdb.models['Model-1'].parts['Part-1']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
del mdb.models['Model-1'].sketches['__profile__']

#           Partition front face
p = mdb.models['Model-1'].parts['Part-1']
f, e, d = p.faces, p.edges, p.datums
t = p.MakeSketchTransform(sketchPlane=f[4], sketchUpEdge=e[7],
    sketchPlaneSide=SIDE1, origin=(Width/2, Height/2, Thickness))
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=1077.77, gridSpacing=26.94, transform=t)
g, v, d1, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=SUPERIMPOSE)
p = mdb.models['Model-1'].parts['Part-1']
p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
s.Line(point1=(-1*Width/2 , Height/10), point2=(Width/2, Height/10))
s.HorizontalConstraint(entity=g[6], addUndoState=False)
s.Line(point1=(-1*Width/2, -1*Height/10), point2=(Width/2, -1*Height/10))
s.HorizontalConstraint(entity=g[7], addUndoState=False)
p = mdb.models['Model-1'].parts['Part-1']
f = p.faces
pickedFaces = f.getSequenceFromMask(mask=('[#10 ]', ), )
e1, d2 = p.edges, p.datums
p.PartitionFaceBySketch(sketchUpEdge=e1[7], faces=pickedFaces, sketch=s)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']

#           Sweep partition through sheets thickness
p = mdb.models['Model-1'].parts['Part-1']
c = p.cells
pickedCells = c.getSequenceFromMask(mask=('[#1 ]', ), )
e = p.edges
pickedEdges =(e[4], )
p.PartitionCellBySweepEdge(sweepPath=e[12], cells=pickedCells,
    edges=pickedEdges)
p = mdb.models['Model-1'].parts['Part-1']
c = p.cells
pickedCells = c.getSequenceFromMask(mask=('[#2 ]', ), )
e1 = p.edges
pickedEdges =(e1[13], )
p.PartitionCellBySweepEdge(sweepPath=e1[20], cells=pickedCells,
    edges=pickedEdges)

#           Creating crack part
s1 = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=200.0)
g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.constraints
s1.setPrimaryObject(option=STANDALONE)
s1.Line(point1=(0.0, 0.0), point2=(Crack_length+10, 0.0))
s1.HorizontalConstraint(entity=g[2], addUndoState=False)
p = mdb.models['Model-1'].Part(name='Crack', dimensionality=THREE_D,
    type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['Crack']
p.BaseShellExtrude(sketch=s1, depth=2*Thickness)
s1.unsetPrimaryObject()
p = mdb.models['Model-1'].parts['Crack']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
del mdb.models['Model-1'].sketches['__profile__']


#-------------------------------------------------------------------
#            ASSIGNING MATERIAL AND ASSEMBLY
#-------------------------------------------------------------------
#           Create material
session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=ON,
    engineeringFeatures=ON)
session.viewports['Viewport: 1'].partDisplay.geometryOptions.setValues(
    referenceRepresentation=OFF)
p = mdb.models['Model-1'].parts['Part-1']
```

```
session.viewports['Viewport: 1'].setValues(displayedObject=p)
mdb.models['Model-1'].Material(name='Steel')
mdb.models['Model-1'].materials['Steel'].Elastic(table=((Youngs_modulus, Poisson_ratio), ))

#           Create section
mdb.models['Model-1'].HomogeneousSolidSection(name='Section-1',
    material='Steel', thickness=None)

#           Assign section to sheet
p = mdb.models['Model-1'].parts['Part-1']
c = p.cells
cells = c.getSequenceFromMask(mask=('[#7 ]', ), )
region = regionToolset.Region(cells=cells)
p = mdb.models['Model-1'].parts['Part-1']
p.SectionAssignment(region=region, sectionName='Section-1', offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='',
    thicknessAssignment=FROM_SECTION)

#           Create instance - sheet part
a = mdb.models['Model-1'].rootAssembly
session.viewports['Viewport: 1'].setValues(displayedObject=a)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    optimizationTasks=OFF, geometricRestrictions=OFF, stopConditions=OFF)
a = mdb.models['Model-1'].rootAssembly
a.DatumCsysByDefault(CARTESIAN)
p = mdb.models['Model-1'].parts['Part-1']
a.Instance(name='Part-1-1', part=p, dependent=OFF)

#           Create instance - crack part
a = mdb.models['Model-1'].rootAssembly
p = mdb.models['Model-1'].parts['Crack']
a.Instance(name='Crack-1', part=p, dependent=OFF)
p1 = a.instances['Crack-1']
p1.translate(vector=(210.0, 0.0, 0.0))
session.viewports['Viewport: 1'].view.fitView()

#           Move crack part to desired location
a = mdb.models['Model-1'].rootAssembly
a.translate(instanceList=('Crack-1', ), vector=(-220.0, Height/2, -10.0))

#           Create a step
mdb.models['Model-1'].StaticStep(name='Step-1', previous='Initial')
session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Step-1')


#-------------------------------------------------------------------
#            Create a XFEM crack and output
#-------------------------------------------------------------------
session.viewports['Viewport: 1'].assemblyDisplay.setValues(interactions=ON,
    constraints=ON, connectors=ON, engineeringFeatures=ON,
    adaptiveMeshConstraints=OFF)
a = mdb.models['Model-1'].rootAssembly
c1 = a.instances['Part-1-1'].cells
cells1 = c1.getSequenceFromMask(mask=('[#7 ]', ), )
crackDomain = regionToolset.Region(cells=cells1)
a = mdb.models['Model-1'].rootAssembly
f1 = a.instances['Crack-1'].faces
faces1 = f1.getSequenceFromMask(mask=('[#1 ]', ), )
crackLocation = regionToolset.Region(faces=faces1)
a = mdb.models['Model-1'].rootAssembly
a.engineeringFeatures.XFEMCrack(name='XFEM_crack', crackDomain=crackDomain,
    allowCrackGrowth=False, crackLocation=crackLocation)

#           Change field output
session.viewports['Viewport: 1'].assemblyDisplay.setValues(interactions=OFF,
    constraints=OFF, connectors=OFF, engineeringFeatures=OFF,
    adaptiveMeshConstraints=ON)
mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].setValues(variables=(
    'S', 'PE', 'PEEQ', 'PEMAG', 'LE', 'U', 'RF', 'CF', 'CSTRESS', 'CDISP',
    'PHILSM', 'PSILSM', 'STATUS', 'STATUSXFEM'))
```

```
#           Change history output
mdb.models['Model-1'].historyOutputRequests['H-Output-1'].setValues(
    contourIntegral='XFEM_crack', sectionPoints=DEFAULT, rebar=EXCLUDE,
    numberOfContours=7, contourType=K_FACTORS)


#--------------------------------------------------------------------
#            Assign boundary conditions and mesh
#--------------------------------------------------------------------
#           Define load
session.viewports['Viewport: 1'].assemblyDisplay.setValues(loads=ON, bcs=ON,
    predefinedFields=ON, connectors=ON, adaptiveMeshConstraints=OFF)
a = mdb.models['Model-1'].rootAssembly
s1 = a.instances['Part-1-1'].faces
side1Faces1 = s1.getSequenceFromMask(mask=('[#2800 ]', ), )
region = regionToolset.Region(side1Faces=side1Faces1)
mdb.models['Model-1'].Pressure(name='Load', createStepName='Step-1',
    region=region, distributionType=UNIFORM, field='', magnitude=-1*Applied_stress,
    amplitude=UNSET)

#           Assign elements
session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=ON, loads=OFF,
    bcs=OFF, predefinedFields=OFF, connectors=OFF)
session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
    meshTechnique=ON)
elemType1 = mesh.ElemType(elemCode=C3D8R, elemLibrary=STANDARD,
    kinematicSplit=AVERAGE_STRAIN, secondOrderAccuracy=OFF,
    hourglassControl=DEFAULT, distortionControl=DEFAULT)
elemType2 = mesh.ElemType(elemCode=C3D6, elemLibrary=STANDARD)
elemType3 = mesh.ElemType(elemCode=C3D4, elemLibrary=STANDARD)
a = mdb.models['Model-1'].rootAssembly
c1 = a.instances['Part-1-1'].cells
cells1 = c1.getSequenceFromMask(mask=('[#7 ]', ), )
pickedRegions =(cells1, )
a.setElementType(regions=pickedRegions, elemTypes=(elemType1, elemType2,
    elemType3))

#           Local seed the fine mesh region (arround the crack)
a = mdb.models['Model-1'].rootAssembly
e1 = a.instances['Part-1-1'].edges
pickedEdges = e1.getSequenceFromMask(mask=('[#3005 ]', ), )
a.seedEdgeByNumber(edges=pickedEdges, number=Width, constraint=FINER)
a = mdb.models['Model-1'].rootAssembly
e1 = a.instances['Part-1-1'].edges
pickedEdges = e1.getSequenceFromMask(mask=('[#c0280 ]', ), )
a.seedEdgeByNumber(edges=pickedEdges, number=Height/5, constraint=FINER)

#           Assign global seeds
a = mdb.models['Model-1'].rootAssembly
partInstances =(a.instances['Part-1-1'], )
a.seedPartInstance(regions=partInstances, size=5.0, deviationFactor=0.1,
    minSizeFactor=0.1)

#           Mesh the structure
a = mdb.models['Model-1'].rootAssembly
partInstances =(a.instances['Part-1-1'], )
a.generateMesh(regions=partInstances)


#--------------------------------------------------------------------
#            Create/submit a job
#--------------------------------------------------------------------
#           Create a job
session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=OFF)
session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
    meshTechnique=OFF)
mdb.Job(name='Job-1', model='Model-1', description='', type=ANALYSIS,
    atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=90,
    memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
    explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF,
    modelPrint=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine='',
```

```
    scratch='', resultsFormat=ODB, multiprocessingMode=DEFAULT, numCpus=1,
    numGPUs=0)

#           Submit the job
mdb.jobs['Job-1'].submit(consistencyChecking=OFF)


# End of script
```

D

3D Crack Propagation

In chapter 4 and 5 of this thesis, a 3D crack is treated. Here, a surface crack with an initial semi-elliptical shape is examined. In the case of a through thickness crack (chapter 3), the stress intensity factors and crack propagation directions were approximately equal along the crack front. In the case of a more complex crack geometry, this is not likely to be the case as is shown in chapter 4 and 5. As a result, the crack may propagate in various directions along the crack front. This makes it more complex to determine the next crack shape. However, 3D cracks may often be more realistic than through thickness cracks and are therefore treated in this thesis as well. A method is introduced in this appendix to model propagating 3D cracks. Due to the complex nature of this analysis, this is not a full automatic procedure but also requires manual steps.

D.1. Methodology of analysis

The flow chard of the analysis is presented in figure D.1. Firstly, the initial crack will be defined prior to inserting it in an existing model of an uncracked structure. Afterwards, the calculation will be performed in which the stress intensity factors and crack propagation directions along the crack front will be calculated. These will be obtained from the contour integral output in the DAT-file after completion of the calculation in Abaqus. With these parameters, the new crack front will be determined. This new crack geometry will then be inserted in the uncracked structure and the procedure starts again.

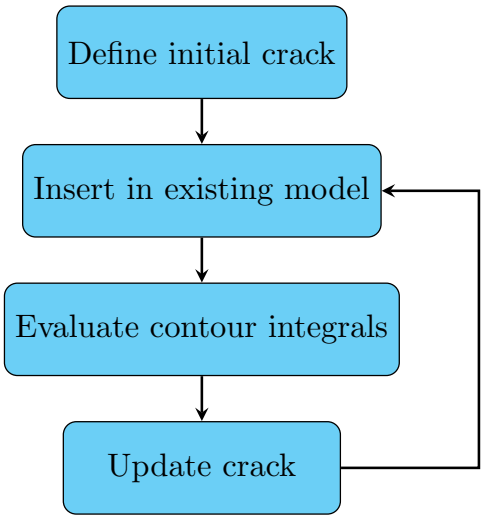


Figure D.1: Flow chard of analysis

D.1.1. Defining initial crack

The initial crack has a semi-elliptical shape. This could be inserted directly in Abaqus as a part. However, Autocad will be used to sketch this geometry after which this shall be imported into Abaqus. This will be useful when the crack shape becomes more complex since Autocad is a useful tool for sketching geometries.

Defining crack coordinates in Excel

The starting geometry is a semi-ellipse with parameters a and c. A full ellipse will be modelled of which half will represent the crack. The remainder sticks out of the model (see appendix B.2). To obtain the coordinates of an ellipse around the origin in XY-plane, the following equations apply [46]:

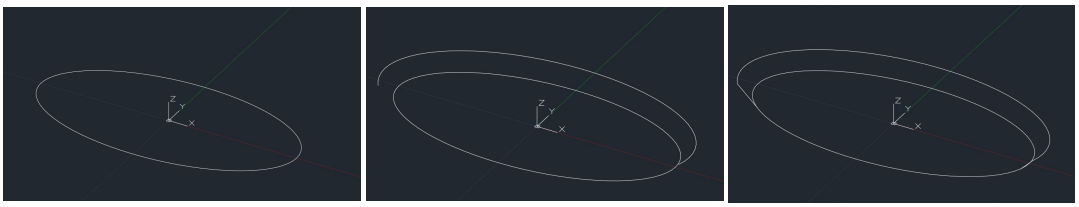
$$x = c\cos\phi \tag{D.1}$$

$$y = a\sin\phi \tag{D.2}$$

In these equations, c and a represent half the crack length and the crack depth respectively (figure 4.1). With this, the contour of an initial crack with a semi-elliptical shape can be determined. The coordinates along the ellipse are determined and saved in Excel.

Sketching crack geometry in Autocad

After the coordinates of the geometry of the crack are determined, the crack can be sketched in Autocad. This can be done by either the '3D Polyline' or the 'SP-line' command. This will generate an ellipse going through all the defined coordinates. All the coordinates can be copied from Excel to Autocad in the command window and the contour of the crack will be sketched. After copying the initial crack, the first increment can be sketched and the second, etc. Finally, the edges are closed to complete the sketch. This is shown in the figures below.



(a) Initial crack (b) First increment (c) Closing edges

Figure D.2: Sketching crack in Autocad

Creating crack part in Abaqus

In order to import the crack geometry in Abaqus, we need to export the sketch from Autocad to a certain file type. This can either be 'sat', 'iges' or 'igs'. Only a solid, region or shapemanager body can be exported to a sat-file from Autocad. In igs and iges format, also wireframes can be exported. After exporting the sketch to a certain file format, this can be imported in Abaqus by following: 'File, Import, Part' in the 'Part' menu. After importing the crack geometry, faces need to be created. This is obtained by following 'Tools, Geometry Edit, Face, Cover Edges' in the 'Part' module.

An igs- or iges file is recommended since no region needs to be created in Autocad. If several increments are present, we obtain separate faces, these can be combined in the 'Mesh' module by following: 'Tools, Virtual Topology, Combine Faces' (this is not actually needed for the analysis).

D.1.2. Updating the crack geometry

After the calculation is performed in Abaqus with the initial crack, the crack can be updated. This is done by following the flow chard as presented as followed:

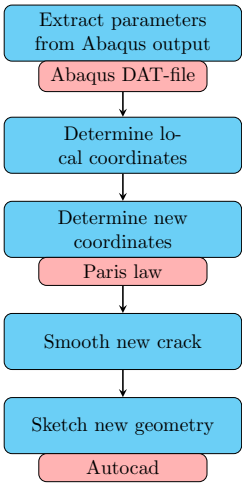


Figure D.3: Flow chard for updating crack

Obtaining SIF's and crack propagation angles
The greatest challenge lies in updating the crack geometry. For this, the calculated SIF and crack propagation angles θ through Abaqus are needed along the entire crack front. Since the SIF values vary along the crack front, many data points need to be obtained for reliable results. The amount of data obtained will increase when choosing a smaller mesh size around the crack front. In the DAT-file of Abaqus, the SIF and crack propagation angles per contour integral are listed. Furthermore, the global coordinates corresponding to these values are listed. In this way, we know what SIF's and crack propagation angles correspond to which data point on the crack front.

Determine local coordinates of crack front
The coordinates of the crack front, obtained through Abaqus, are in the global coordinate system. These coordinates are transferred to a local coordinate system. Here, the semi-elliptical crack lies in the XY-plane of the local coordinate system as can be seen from figure D.2. In order to transfer from global to local, the inverse rotation matrices and translation vector are used:

$$\textit{Translation vector:} \quad \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{D.3}$$

$$\textit{Inverse rotation matrix around x:} \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{pmatrix} \tag{D.4}$$

$$\textit{Inverse rotation matrix around y:} \quad \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{pmatrix} \tag{D.5}$$

$$\textit{Inverse rotation matrix around z:} \quad \begin{pmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{D.6}$$

Determining coordinates of new crack front
Before the new coordinates can be determined, the increment length needs to be determined for each data

point. This is done using the Paris law. The following relations have been used ($C = 3{,}98 \cdot 10^{-13}$ and $m = 2{,}88$):

$$\Delta a = \Delta n \cdot C \cdot \Delta K_{eff}^{m} \tag{D.7}$$

$$\Delta K_{eff} = \sqrt[4]{(\Delta K_I)^4 + 8(\Delta K_{II})^4} \tag{D.8}$$

In figure D.4 is illustrated how the new coordinates are determined. At a certain data point, the crack will increase with Δa orthogonal to the crack front and under an angle θ. In this figure, ϕ and θ are angles with respect to the local coordinate system as presented in figure D.2. The coordinates defined by Abaqus are called here data points. Although the number and position of these data points depend on the mesh size, they lie approximately on the crack front and are therefore fit to use.

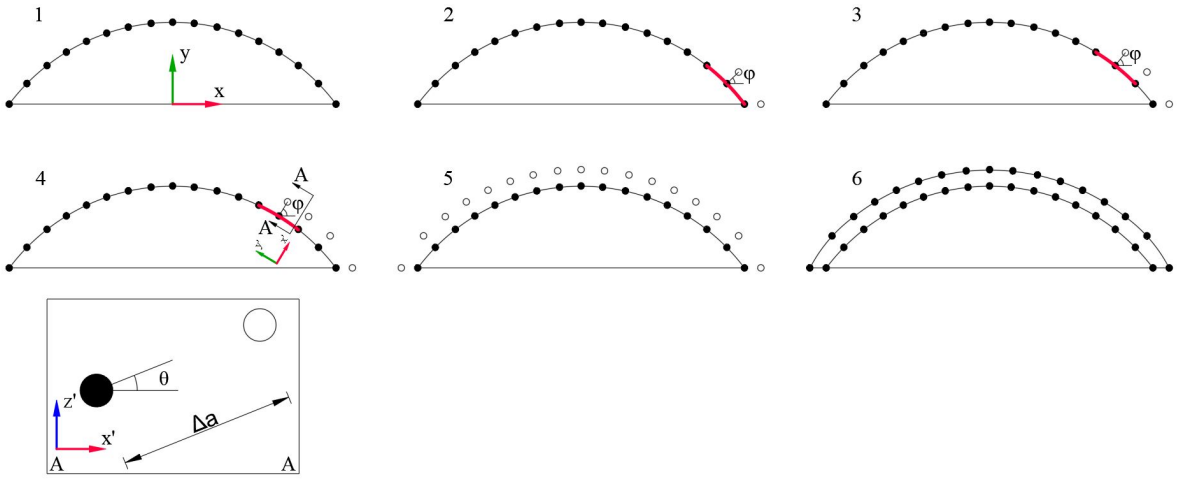


Figure D.4: Determining new coordinates

The angle θ is determined by the summation of the θ from the previous calculation and the crack propagation angle calculated by Abaqus for the given data point. If $n-1$ represents the old point and n the new point, the angle θ is determined by:

$$\theta_n = \theta_{n-1} + \theta_{Abaqus} \tag{D.9}$$

In Abaqus DAT-file, the crack direction vectors are listed for each data point. These can be used to obtain θ_{n-1}. Angle ϕ is orthogonal to the crack front. This angle is determined by first determining a regression line of second degree. For this, 3 datum points are needed. The regression line has the following form:

$$y = a_r \cdot x^2 + b_r \cdot x + c_r \tag{D.10}$$

Here, a_r, b_r and c_r are constants determined by using the coordinates of the 3 datum points as shown in figure D.4. The angle ϕ is orthogonal to the regression front and is then calculated through:

$$\phi = \tan\left(\frac{-1}{\frac{dy}{dx}}\right) = \tan\left(\frac{-1}{2a_r \cdot x + b_r}\right) \tag{D.11}$$

The new (global) coordinates are then determined by following the relations given below. It should be noted that the coordinates are with respect to the coordinate system as introduced with the initial crack.

$$\Delta X = \Delta a \cdot \cos\theta_n \cdot \cos\phi \tag{D.12}$$

$$\Delta Y = \Delta a \cdot \cos\theta_n \cdot \sin\phi \tag{D.13}$$

$$\Delta Z = \Delta a \cdot \sin\theta_n \tag{D.14}$$

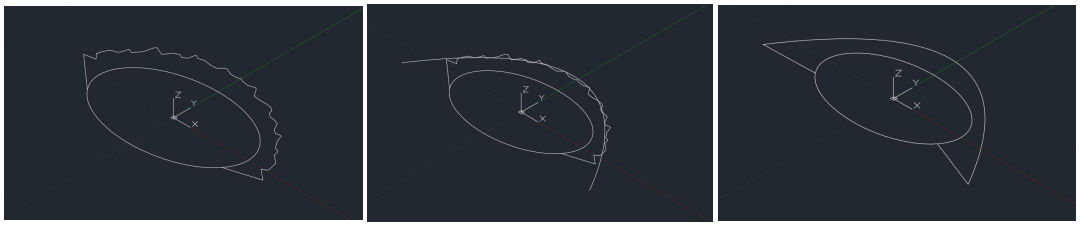
$$X_{new} = X_{old} + \Delta X \tag{D.15}$$

$$Y_{new} = Y_{old} + \Delta Y \tag{D.16}$$

$$Z_{new} = Z_{old} + \Delta Z \tag{D.17}$$

The procedure is performed for each data point to obtain the new coordinates.

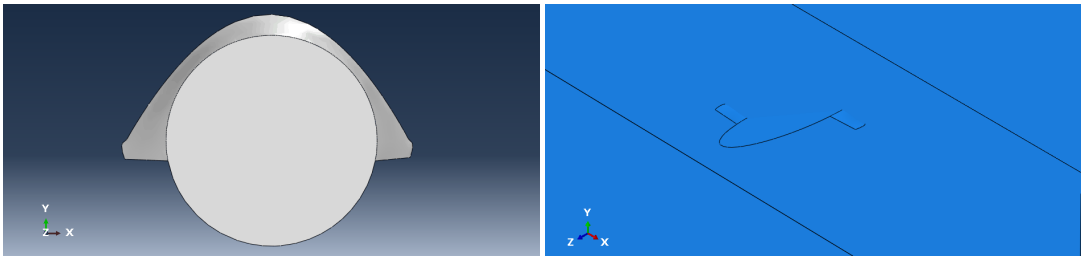
Smoothing the new crack front
Applying the steps described above does not lead to a smooth new crack front. Therefore regression formulas are applied on the new data points. In the XY-plane a regression of second degree while in the XZ plane of first degree. Since the initial crack has a semi-elliptical shape, a second degree fits this shape best. As for the XZ- plane, the new data points do follow a certain straight line and therefor the first degree fits best here. The application of the regression is shown in the figures below.



(a) Without regression (b) With and without regression (c) With regression

Figure D.5: Sketching first increment in Autocad

Sketching new geometry
After obtaining all the new coordinates, the new crack front can be sketched in Autocad. This is added to the previous crack sketch. The procedure is the same as was explained in section D.1.1. After importing this in Abaqus and combining the faces created (see section D.1.1, the new crack part is presented in Abaqus as shown in the following figure.



(a) First increment part (b) First increment assembled

Figure D.6: First increment presented in Abaqus

D.2. Python scripts

Below, the python scripts are presented used to execute the methodology as presented in the previous section. In total 4 scripts have been written which are linked to each other. The first script is used to read the DAT-file from the Abaqus output. Here all data will be saved to an excel file (here named data_input). Also regressions are applied in form of 9^{th} order polynomials. To obtain the raw data (unmanipulated) the lines for regression can also be made inactive. The second script used the translation vector and rotation matrices to obtain the local coordinates of the crack front. Additionally, the direction vectors of the crack on the entire front is also determined here. This is necessary to determine the new crack propagation direction with accuracy. The third script determines the new crack front by using the Paris law. Other crack propagation relations can also be used, simply change the formula and parameters. The fourth and final code is used to smooth the new crack front. Again polynomials are used to fit the data of the crack front in both xz- as in xy-plane.

D.2.1. Saving data to Excel

```
#   In this script, a DAT-file will be read and the crack front coordinates, SIFs and crack propagation angles will
#   be saved in an excel file. For this, an existing excel file needs to be present in the location stated below.

from openpyxl import load_workbook
import pandas as pd
import runpy2
import numpy as np

#   INPUT PARAMETERS
Number_cycles           = 10000    # Applied number of load cycles
Number_excluding        = 0         # Number of data points per boundary that will be extrapolated
Translation_vector      = [1882.98, 56.71, 7700.21]      # Translation of coordinate system (x, y, z) (global coor of m
Rotation_vector         = [0, 90, 77]      # Rotations of coordinate system (in deg) (x, y, z)
Order_rotation          = ['z', 'y', 'x']   # i.e. first rotation around x-axis then y, choose x,y,z all once


    # Specify which file to open
Path_datfile    = 'C:/temp/Semi_elliptical_centre_cracked_plate/'           # Location of DAT-file
Name_datfile    = 'Inrement_5_66_019mm_trythisone.dat'                        # Name of DAT-file
file_to_open2   = Path_datfile + Name_datfile

#   Defining the excel files to read and save
path            = "C:/temp/Semi_elliptical_centre_cracked_plate/"           # Location of existing Excel file
file_name1      = "data.xlsx"                                               # Name of existing Excel file
file_name2      = "data_input.xlsx"
file_to_open    = path + file_name1
file_to_save    = path + file_name2

#   Define the file and sheet to edit
wb      = load_workbook(file_to_open)
sheets  = wb.sheetnames
sheet1  = wb[sheets[0]]

    # Read the file
Names           = np.arange(0, 9)
df              = pd.read_csv(file_to_open2, delim_whitespace=True, names=Names)
Number_rows     = len(df.index)

#   Determine the number of countours and data points
Number_contours = int(df.iloc[48][4].strip('contours='))
Number_XFEM     = int(df.iloc[-20][0].strip('XFEM_'))

#   Reducing data size to read
condition = 'CALCULATION'; stop = 0
for i in range (Number_rows):
    for ii in range (len(Names)):
        if df.iloc[i][ii] == condition:
            stop = i
            break
    else:
        continue
    break
df              = pd.read_csv(file_to_open2, delim_whitespace=True, names=Names)[stop:]
Number_rows     = len(df.index)
```

```
#   Create empty arrays
XFEM_coordinates    = np.zeros((Number_XFEM+1, 4))
XFEM_virtual_direc  = np.zeros((Number_XFEM+1, 4))
XFEM_SIF1           = np.zeros((Number_XFEM+1, Number_contours+2))
XFEM_SIF2           = np.zeros((Number_XFEM+1, Number_contours+2))
XFEM_SIF3           = np.zeros((Number_XFEM+1, Number_contours+2))
XFEM_prop           = np.zeros((Number_XFEM+1, Number_contours+2))
for i in range(Number_XFEM+1):
    XFEM_coordinates[i][0] = i
    XFEM_virtual_direc[i][0] = i
    XFEM_SIF1[i][0] = i
    XFEM_SIF2[i][0] = i
    XFEM_SIF3[i][0] = i
    XFEM_prop[i][0] = i

#   Loop through file and fill in arrays
for i in range (Number_XFEM):
    condition = 'XFEM_'+str(i+1)
    count = 0
    row_number = [0, 0]; column_number = [0, 0]
    for ii in range (Number_rows):
        for iii in range (len(Names)):
            if df.iloc[ii][iii] == condition:
                count += 1
                row_number[count-1] = ii
                column_number[count-1] = iii
    for ii in range (3):
        XFEM_coordinates[i+1][ii+1] = df.iloc[row_number[0]][ii+3]
        XFEM_virtual_direc[i+1][ii+1] = df.iloc[row_number[0]+1][ii+6]
    for ii in range (5):
        XFEM_SIF1[i+1][ii+1] = df.iloc[row_number[1]][ii+2]
        XFEM_SIF2[i+1][ii+1] = df.iloc[row_number[1]+1][ii+1]
        XFEM_SIF3[i+1][ii+1] = df.iloc[row_number[1]+2][ii+1]
        XFEM_prop[i+1][ii+1] = df.iloc[row_number[1]+3][ii+3]
    for ii in range (Number_contours-5):
        XFEM_SIF1[i+1][ii+6] = df.iloc[row_number[1]+5][ii+1]
        XFEM_SIF2[i+1][ii+6] = df.iloc[row_number[1]+6][ii+1]
        XFEM_SIF3[i+1][ii+6] = df.iloc[row_number[1]+7][ii+1]
        XFEM_prop[i+1][ii+6] = df.iloc[row_number[1]+8][ii+3]

#   Calculating the average SIF
for i in range(Number_XFEM):
    XFEM_SIF1[i+1][Number_contours+1] = np.mean(XFEM_SIF1[i+1][6:Number_contours+1])
    XFEM_SIF2[i+1][Number_contours+1] = np.mean(XFEM_SIF2[i+1][6:Number_contours+1])
    XFEM_SIF3[i+1][Number_contours+1] = np.mean(XFEM_SIF3[i+1][6:Number_contours+1])
    XFEM_prop[i+1][Number_contours+1] = np.mean(XFEM_prop[i+1][6:Number_contours+1])

# Reducing the effect of outliers
for i in range  (Number_XFEM-1):
    if abs(XFEM_prop[i+2][11] - XFEM_prop[i+1][11]) > 5:
        XFEM_prop[i+2][11] = np.mean([XFEM_prop[i+2][11], XFEM_prop[i+1][11]])
    if abs(XFEM_SIF1[i+2][11] - XFEM_SIF1[i+1][11]) > 5:
        XFEM_SIF1[i+2][11] = np.mean([XFEM_SIF1[i+2][11], XFEM_SIF1[i+1][11]])
    if abs(XFEM_SIF2[i+2][11] - XFEM_SIF2[i+1][11]) > 5:
        XFEM_SIF2[i+2][11] = np.mean([XFEM_SIF2[i+2][11], XFEM_SIF2[i+1][11]])
    if abs(XFEM_SIF3[i+2][11] - XFEM_SIF3[i+1][11]) > 5:
        XFEM_SIF3[i+2][11] = np.mean([XFEM_SIF3[i+2][11], XFEM_SIF3[i+1][11]])

#   Excluding inaccurate surface values
for i in range (Number_excluding):
    XFEM_SIF1[Number_excluding-i][11] = np.mean([XFEM_SIF1[Number_excluding-i+1][11], XFEM_SIF1[Number_excluding-i+2][11]])
    XFEM_SIF1[Number_XFEM-Number_excluding+i+1][11] = np.mean([XFEM_SIF1[Number_XFEM-Number_excluding+i][11], XFEM_SIF1[Number_
    XFEM_SIF2[Number_excluding-i][11] = np.mean([XFEM_SIF2[Number_excluding-i+1][11], XFEM_SIF2[Number_excluding-i+2][11]])
    XFEM_SIF2[Number_XFEM-Number_excluding+i+1][11] = np.mean([XFEM_SIF2[Number_XFEM-Number_excluding+i][11], XFEM_SIF2[Number_
    XFEM_SIF3[Number_excluding-i][11] = np.mean([XFEM_SIF3[Number_excluding-i+1][11], XFEM_SIF3[Number_excluding-i+2][11]])
    XFEM_SIF3[Number_XFEM-Number_excluding+i+1][11] = np.mean([XFEM_SIF3[Number_XFEM-Number_excluding+i][11], XFEM_SIF3[Number_
    XFEM_prop[Number_excluding-i][11] = np.mean([XFEM_prop[Number_excluding-i+1][11], XFEM_prop[Number_excluding-i+2][11]])
    XFEM_prop[Number_XFEM-Number_excluding+i+1][11] = np.mean([XFEM_prop[Number_XFEM-Number_excluding+i][11], XFEM_prop[Number_

#   Filling in the excel sheet
```

```
for i in range(Number_XFEM):
    sheet1.cell(i+5, 4).value = XFEM_coordinates[i+1][1]        # X_global
    sheet1.cell(i+5, 5).value = XFEM_coordinates[i+1][2]        # Y_global
    sheet1.cell(i+5, 6).value = XFEM_coordinates[i+1][3]        # Z_global
    sheet1.cell(i+5, 36).value = XFEM_virtual_direc[i+1][1]
    sheet1.cell(i+5, 37).value = XFEM_virtual_direc[i+1][2]
    sheet1.cell(i+5, 38).value = XFEM_virtual_direc[i+1][3]
    sheet1.cell(i+5, 7).value = XFEM_SIF1[i+1][11]              # K1
    sheet1.cell(i+5, 8).value = XFEM_SIF2[i+1][11]              # K2
    sheet1.cell(i+5, 9).value = XFEM_SIF3[i+1][11]              # K3
    if sheet1.cell(5, 6).value < Translation_vector[2]:         # Note this part, check value (Abaqus definition differs
        sheet1.cell(i+5, 10).value = XFEM_prop[i+1][11]
    else:
        XFEM_prop[i+1][11] = -1*XFEM_prop[i+1][11]
        sheet1.cell(i+5, 10).value = XFEM_prop[i+1][11]

# ##############################################################################################
#   Unlock this part for regression on SIFs and propagation angles
Number_excluding = 0
List_numbers    = np.zeros(Number_XFEM)
XFEM_prop_regr  = np.zeros(Number_XFEM)
XFEM_SIF1_regr  = np.zeros(Number_XFEM)
XFEM_SIF2_regr  = np.zeros(Number_XFEM)
XFEM_SIF3_regr  = np.zeros(Number_XFEM)
for i in range(Number_XFEM):
    List_numbers[i]     = i
    XFEM_prop_regr[i]   = XFEM_prop[i+1][11]
    XFEM_SIF1_regr[i]   = XFEM_SIF1[i+1][11]
    XFEM_SIF2_regr[i]   = XFEM_SIF2[i+1][11]
    XFEM_SIF3_regr[i]   = XFEM_SIF3[i+1][11]
gg0             = np.polyfit(List_numbers, XFEM_prop_regr, 9)
gg1             = np.polyfit(List_numbers, XFEM_SIF1_regr, 9)
gg2             = np.polyfit(List_numbers, XFEM_SIF2_regr, 9)
gg3             = np.polyfit(List_numbers, XFEM_SIF3_regr, 9)
def XFEM_prop_poly(x):
    return gg0[0]*x**9+gg0[1]*x**8+gg0[2]*x**7+gg0[3]*x**6+gg0[4]*x**5+gg0[5]*x**4+gg0[6]*x**3+gg0[7]*x**2+gg0[8]*x+gg0
def XFEM_SIF1_poly(x):
    return gg1[0]*x**9+gg1[1]*x**8+gg1[2]*x**7+gg1[3]*x**6+gg1[4]*x**5+gg1[5]*x**4+gg1[6]*x**3+gg1[7]*x**2+gg1[8]*x+gg1
def XFEM_SIF2_poly(x):
    return gg2[0]*x**9+gg2[1]*x**8+gg2[2]*x**7+gg2[3]*x**6+gg2[4]*x**5+gg2[5]*x**4+gg2[6]*x**3+gg2[7]*x**2+gg2[8]*x+gg2
def XFEM_SIF3_poly(x):
    return gg3[0]*x**9+gg3[1]*x**8+gg3[2]*x**7+gg3[3]*x**6+gg3[4]*x**5+gg3[5]*x**4+gg3[6]*x**3+gg3[7]*x**2+gg3[8]*x+gg3
for i in range(Number_XFEM):
    sheet1.cell(i+5, 10).value = XFEM_prop_poly(List_numbers[i])
    sheet1.cell(i+5, 7).value = XFEM_SIF1_poly(List_numbers[i])
    sheet1.cell(i+5, 8).value = XFEM_SIF2_poly(List_numbers[i])
    sheet1.cell(i+5, 9).value = XFEM_SIF3_poly(List_numbers[i])
# ##############################################################################################

#   Save into a new excel file
wb.save(file_to_save)

#   Run 'Creating_Excel_output' script
runpy2.run_path("C:/Users/David/PycharmProjects/Geometry_crack/Creating_Excel_output.py")
```

D.2.2. Determining local coordinates of crack front

```
#   This script is runned automaticly by 'Creating_Excel_input.py'.
#   In this script, the local coordinates are determined of the crack front.

from openpyxl import load_workbook
import numpy as np
import runpy2
import Creating_Excel_input

#   INPUT PARAMETERS
Number_XFEM             = Creating_Excel_input.Number_XFEM
Translation_vector      = Creating_Excel_input.Translation_vector
Rotation_vector         = Creating_Excel_input.Rotation_vector
Order_rotation          = Creating_Excel_input.Order_rotation
```

```
#   Defining the excel files to read and save
path                    = Creating_Excel_input.path
file_name1              = Creating_Excel_input.file_name2
file_to_open    = path + file_name1

#   Define the file and sheet to edit
wb      = load_workbook(file_to_open)
sheets  = wb.sheetnames
sheet1  = wb[sheets[0]]

#   Defining local coordinates, rotation angles in radians
translation_x   = Translation_vector[0]         ; rotation_x      = Rotation_vector[0] *np.pi/180
translation_y   = Translation_vector[1]         ; rotation_y      = Rotation_vector[1] *np.pi/180
translation_z   = Translation_vector[2]         ; rotation_z      = Rotation_vector[2] *np.pi/180
rotation_matrix_z = np.array([[np.cos(rotation_z), np.sin(rotation_z), 0], [-1 * np.sin(rotation_z), np.cos(rotation_z), 0], [0
rotation_matrix_x = np.array([[1, 0, 0], [0, np.cos(rotation_x), np.sin(rotation_x)], [0, -1 * np.sin(rotation_x), np.cos(rotat
rotation_matrix_y = np.array([[np.cos(rotation_y), 0, -1 * np.sin(rotation_y)], [0, 1, 0], [np.sin(rotation_y), 0, np.cos(rotat
for i in range(Number_XFEM):
    x_global        = sheet1.cell(5+i, 4).value
    y_global        = sheet1.cell(5+i, 5).value
    z_global        = sheet1.cell(5+i, 6).value
    x_global_vector = np.array([x_global, y_global, z_global])
    x_local_vector = np.array([0, 0, 0])
    x_local_norot = x_global_vector[0] - translation_x
    y_local_norot = x_global_vector[1] - translation_y
    z_local_norot = x_global_vector[2] - translation_z
    x_local_vector_norot = np.array([x_local_norot, y_local_norot, z_local_norot])
    rotation_matrix_z = np.array([[np.cos(rotation_z), np.sin(rotation_z), 0], [-1*np.sin(rotation_z), np.cos(rotation_z), 0],
    rotation_matrix_x = np.array([[1, 0, 0], [0, np.cos(rotation_x), np.sin(rotation_x)], [0, -1*np.sin(rotation_x), np.cos(rot
    rotation_matrix_y = np.array([[np.cos(rotation_y), 0, -1*np.sin(rotation_y)], [0, 1, 0], [np.sin(rotation_y), 0, np.cos(rot
    if Order_rotation[0] == 'x':
        if Order_rotation[1] == 'y':
            x_local_vector = np.dot(rotation_matrix_z, np.dot(rotation_matrix_y, np.dot(rotation_matrix_x, x_local_vector_norot
        else:
            x_local_vector = np.dot(rotation_matrix_y, np.dot(rotation_matrix_z, np.dot(rotation_matrix_x, x_local_vector_norot
    if Order_rotation[0] == 'y':
        if Order_rotation[1] == 'x':
            x_local_vector = np.dot(rotation_matrix_z, np.dot(rotation_matrix_x, np.dot(rotation_matrix_y, x_local_vector_norot
        else:
            x_local_vector = np.dot(rotation_matrix_x, np.dot(rotation_matrix_z, np.dot(rotation_matrix_y, x_local_vector_norot
    if Order_rotation[0] == 'z':
        if Order_rotation[1] == 'x':
            x_local_vector = np.dot(rotation_matrix_y, np.dot(rotation_matrix_x, np.dot(rotation_matrix_z, x_local_vector_norot
        else:
            x_local_vector = np.dot(rotation_matrix_x, np.dot(rotation_matrix_y, np.dot(rotation_matrix_z, x_local_vector_norot
    sheet1.cell(5+i, 13).value = x_local_vector[0]
    sheet1.cell(5+i, 14).value = x_local_vector[1]
    sheet1.cell(5+i, 15).value = x_local_vector[2]



#   Determining the virtual extension direction in local coordinate system
Global_angles = np.zeros(Number_XFEM)
Numbers = np.zeros(Number_XFEM)
for i in range(Number_XFEM):
    Numbers[i] = i
    Vector_glob     = np.zeros(3)
    V_loc           = np.zeros(3)
    for ii in range (len(Vector_glob)):
        Vector_glob[ii] = sheet1.cell(5+i, 36+ii).value
    if Order_rotation[0] == 'x':
        if Order_rotation[1] == 'y':
            V_loc = np.dot(rotation_matrix_z, np.dot(rotation_matrix_y, np.dot(rotation_matrix_x, Vector_glob)))
        else:
            V_loc = np.dot(rotation_matrix_y, np.dot(rotation_matrix_z, np.dot(rotation_matrix_x, Vector_glob)))
    if Order_rotation[0] == 'y':
        if Order_rotation[1] == 'x':
            V_loc = np.dot(rotation_matrix_z, np.dot(rotation_matrix_x, np.dot(rotation_matrix_y, Vector_glob)))
        else:
            V_loc = np.dot(rotation_matrix_x, np.dot(rotation_matrix_z, np.dot(rotation_matrix_y, Vector_glob)))
```

```
    if Order_rotation[0] == 'z':
        if Order_rotation[1] == 'x':
            V_loc = np.dot(rotation_matrix_y, np.dot(rotation_matrix_x, np.dot(rotation_matrix_z, Vector_glob)))
        else:
            V_loc = np.dot(rotation_matrix_x, np.dot(rotation_matrix_y, np.dot(rotation_matrix_z, Vector_glob)))
    for ii in range (3):
        sheet1.cell(5+i, 39+ii).value = V_loc[ii]
    Local_angle = np.arccos((V_loc[0]**2 + V_loc[1]**2) / (np.sqrt(V_loc[0]**2 + V_loc[1]**2 + V_loc[2]**2)*np.sqrt(V_lo
    if V_loc[2] > 0:
        sheet1.cell(5+i, 42).value = np.round(Local_angle *180/np.pi, 2)
        Global_angles[i] = np.round(Local_angle *180/np.pi, 2)
    else:
        sheet1.cell(5+i, 42).value = -1*np.round(Local_angle*180/np.pi, 2)
        Global_angles[i] = -1*np.round(Local_angle *180/np.pi, 2)
Pga = np.polyfit(Numbers, Global_angles, 9)
def func_glob_ang(x):
    return Pga[0]*x**9+Pga[1]*x**8+Pga[2]*x**7+Pga[3]*x**6+Pga[4]*x**5+Pga[5]*x**4+Pga[6]*x**3+Pga[7]*x**2+Pga[8]*x+Pga[
for i in range(Number_XFEM):
    sheet1.cell(5+i, 43).value = func_glob_ang(Numbers[i])

# Save
wb.save(file_to_open)

# #   Run 'Creating Excel_output2' script
runpy2.run_path("C:/Users/David/PycharmProjects/Geometry_crack/Creating_Excel_output2.py")
```

D.2.3. Determining coordinates of new crack front

```
#   This script is runned automaticly by 'Creating_Excel_output.py'
#   In this script, the new coordinates are determined and saved in an excel file. Paris's law is implemented.

from openpyxl import load_workbook
import numpy as np
import runpy2
import Creating_Excel_input

#   INPUT PARAMETERS
Number_XFEM             = Creating_Excel_input.Number_XFEM
Number_cycles           = Creating_Excel_input.Number_cycles


#   Defining the excel files to read and save
path                    = Creating_Excel_input.path
file_name1              = Creating_Excel_input.file_name2
file_to_open    = path + file_name1
wb      = load_workbook(file_to_open)
sheets  = wb.sheetnames
sheet1  = wb[sheets[0]]

path2               = path
file_name2          = "data_global_angles.xlsx"
file_to_open2       = path2 + file_name2
wb2                 = load_workbook(file_to_open2)
sheets2             = wb2.sheetnames
sheet12             = wb2[sheets2[0]]

#   Defining arrays
global_angles       = np.zeros(Number_XFEM)
increment_lengths   = np.zeros(Number_XFEM)
delta_K             = np.zeros(Number_XFEM)
for i in range(Number_XFEM):
    global_angles[i] = sheet1.cell(i+5, 43).value + sheet1.cell(i+5, 10).value

#########################################################################################################################
######################################## APPLY SUITABLE PROPAGATION RELATION ############################################
#   Apply a suitable propagation relation including the definition of Keff (Current is Paris law)
C = 3.98*10**-13;   m = 2.88
for i in range(Number_XFEM):
    delta_K[i] = (sheet1.cell(i+5, 7).value**4 + 8*sheet1.cell(i+5, 8).value**4)**0.25  # Keff
    increment_lengths[i] = Number_cycles * C * (delta_K[i])**m                          # Paris law
#########################################################################################################################
```

```
#########################################################################################################################

#   Define some parameters for calculation
for i in range(Number_XFEM-2):
    x_coor_old      = np.array([sheet1.cell(i+5, 13).value, sheet1.cell(i+6, 13).value, sheet1.cell(i+7, 13).value])
    y_coor_old      = np.array([sheet1.cell(i+5, 14).value, sheet1.cell(i+6, 14).value, sheet1.cell(i+7, 14).value])
    z_coor_old      = np.array([sheet1.cell(i+5, 15).value, sheet1.cell(i+6, 15).value, sheet1.cell(i+7, 15).value])
    increment_length    = increment_lengths[i+1]
    increment_angle     = global_angles[i+1] * np.pi / 180

#   Regression formula 2nd order for 3 data points
    g = np.polyfit(x_coor_old, y_coor_old, 2)
    def polyfunc(x):
        return g[0]*x**2 + g[1]*x + g[2]
    def h(x):
        return -1/(2*g[0]*x + g[1])

#   Determine new parameters
    phi = np.arctan(h(x_coor_old[1]))
    delta_x             = increment_length * np.cos(increment_angle) * np.cos(phi)
    delta_y             = increment_length * np.cos(increment_angle) * np.sin(phi)
    delta_z             = increment_length*np.sin(increment_angle)
    if h(x_coor_old[1]) > 0:
        x_coor_new = x_coor_old[1] + delta_x
        y_coor_new = y_coor_old[1] + delta_y
    else:
        x_coor_new = x_coor_old[1] - delta_x
        y_coor_new = y_coor_old[1] - delta_y
    z_coor_new = z_coor_old[1] + delta_z

#   Save new coordinates in Excel
    sheet1.cell(i+6, 16).value = x_coor_new
    sheet1.cell(i+6, 17).value = y_coor_new
    sheet1.cell(i+6, 18).value = z_coor_new

#   Determine new coordinates for first and last crack front point
delta_x1 = increment_lengths[0] * np.cos(global_angles[0]*np.pi/180)
delta_y1 = 0
delta_z1 = increment_lengths[0] * np.sin(global_angles[0]*np.pi/180)

if sheet1.cell(5, 13).value > 0:
    sheet1.cell(5, 16).value = sheet1.cell(5, 13).value + delta_x1
    sheet1.cell(5, 17).value = sheet1.cell(5, 14).value + delta_y1
else:
    sheet1.cell(5, 16).value = sheet1.cell(5, 13).value - delta_x1
    sheet1.cell(5, 17).value = sheet1.cell(5, 14).value - delta_y1
sheet1.cell(5, 18).value = sheet1.cell(5, 15).value + delta_z1
delta_x2 = increment_lengths[Number_XFEM-1] * np.cos(global_angles[Number_XFEM-1]*np.pi/180)
delta_y2 = 0
delta_z2 = increment_lengths[Number_XFEM-1] * np.sin(global_angles[Number_XFEM-1]*np.pi/180)

if sheet1.cell(Number_XFEM+4, 13).value > 0:
    sheet1.cell(Number_XFEM+4, 16).value = sheet1.cell(Number_XFEM+4, 13).value + delta_x2
    sheet1.cell(Number_XFEM+4, 17).value = sheet1.cell(Number_XFEM+4, 14).value + delta_y2
else:
    sheet1.cell(Number_XFEM+4, 16).value = sheet1.cell(Number_XFEM+4, 13).value - delta_x2
    sheet1.cell(Number_XFEM+4, 17).value = sheet1.cell(Number_XFEM+4, 14).value - delta_y2
sheet1.cell(Number_XFEM+4, 18).value = sheet1.cell(Number_XFEM+4, 15).value + delta_z2

#   Save as a new file
wb.save(file_to_open)

#   Run 'Creating Excel_output2' script
runpy2.run_path("C:/Users/David/PycharmProjects/Geometry_crack/Creating_Excel_output3.py")
```

D.2.4. Smoothing new crack front

```
#   This script is runned automaticly by 'Creating_Excel_output2.py'
#   In this script, a regression is used on the crack front.

from openpyxl import load_workbook
```

```
import numpy as np
import Creating_Excel_input

#   INPUT PARAMETERS
Number_XFEM     = Creating_Excel_input.Number_XFEM


# Defining the excel files to read and save
path            = Creating_Excel_input.path
file_name1      = Creating_Excel_input.file_name2
file_to_open    = path + file_name1

# Define the file and sheet to edit
wb      = load_workbook(file_to_open)
sheets  = wb.sheetnames
sheet1  = wb[sheets[0]]

#   Define arrays
x_values = np.zeros(Number_XFEM); y_values = np.zeros(Number_XFEM); z_values = np.zeros(Number_XFEM)
for i in range  (Number_XFEM):
    x_values[i] = sheet1.cell(i+5, 16).value
    y_values[i] = sheet1.cell(i+5, 17).value
    z_values[i] = sheet1.cell(i+5, 18).value

#   Define polynomials
g = np.polyfit(x_values, y_values, 6)
gg = np.polyfit(x_values, z_values, 6)
def polyfunc(x):
    return g[0]*x**6 + g[1]*x**5 + g[2]*x**4 + g[3]*x**3 + g[4]*x**2 + g[5]*x + g[6]
def polyfunc2(x):
    return gg[0]*x**6 + gg[1]*x**5 + gg[2]*x**4 + gg[3]*x**3 + gg[4]*x**2 + gg[5]*x + gg[6]

#   Fill in sheet for regression on crack geometry
x_list = np.linspace(sheet1.cell(5, 16).value, sheet1.cell(Number_XFEM+4, 16).value, 100)
for i in range (len(x_list)):
    sheet1.cell(6+i, 25).value = x_list[i]
    sheet1.cell(6+i, 26).value = polyfunc(x_list[i])
    sheet1.cell(6+i, 27).value = polyfunc2(x_list[i])

#   linear extrapolating the part that sticks out the model
sheet1.cell(5, 26).value = -1; sheet1.cell(len(x_list)+6, 26).value = -1
sheet1.cell(5, 27).value = sheet1.cell(6, 27).value
sheet1.cell(len(x_list)+6, 27).value = sheet1.cell(len(x_list)+5, 27).value
x11 = sheet1.cell(6, 25).value; y11 = sheet1.cell(6, 26).value
x21 = sheet1.cell(10, 25).value; y21 = sheet1.cell(10, 26).value
a1 = (y21-y11)/(x21-x11); b1 = y11 - a1*x11
sheet1.cell(5, 25).value = (-1-b1)/a1
x12 = sheet1.cell(len(x_list)+5, 25).value; y12 = sheet1.cell(len(x_list)+5, 26).value
x22 = sheet1.cell(len(x_list)+1, 25).value; y22 = sheet1.cell(len(x_list)+1, 26).value
a2 = (y22-y12)/(x22-x12); b2 = y12 - a2*x12
sheet1.cell(len(x_list)+6, 25).value = (-1-b2)/a2

# Save as a new file
wb.save(file_to_open)
```

E

Propagation Results from Mark 6'-6"

In chapter 5, crack propagation was investigated for a crack at the rib-to-crossbeam joint in an orthotropic steel deck. This annex is the result of a preliminary study to the one presented in chapter 5.

E.1. Crack position

The coordinates of the crack with respect to the crossbeam are presented in table 5.1. Furthermore, it was observed that the crack follows the weld toe shape. In section 5.2.2 the weld shape was determined for the model accounting for this and the measurements. Given the difficulty of crack propagation modelling, a specific trajectory of the crack will be examined. The crack shape may be influenced by both the weld shape as the residual stresses originating from the weld. Therefore, the outermost trajectory of the crack is chosen to examine, as this is farthest away from the weld. Consequently trajectory 6'-6" is proposed as initial crack (see section 5.1.2). The assumed initial crack is shown in the following graph. The corresponding coordinates are listed in table 5.1.

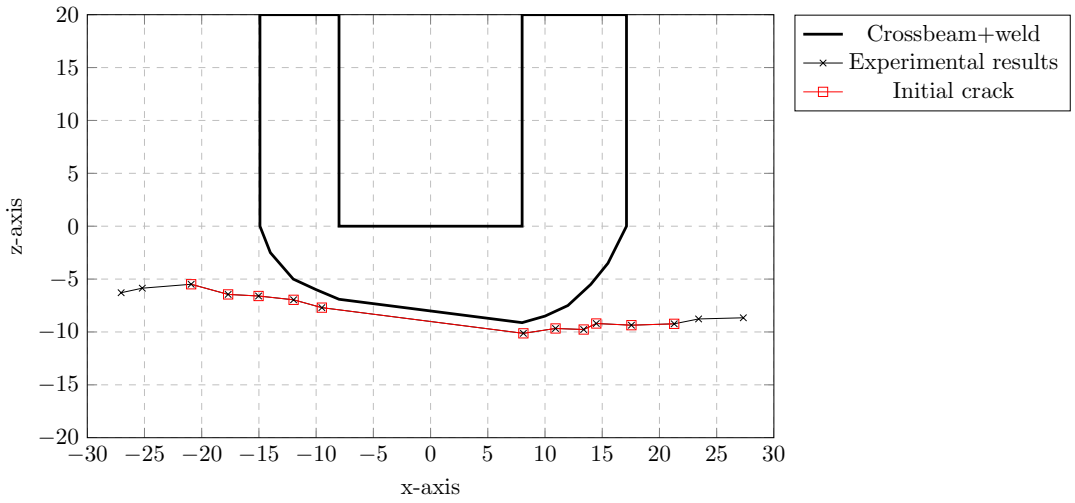


Figure E.1: Assumed initial crack in model

E.2. Initial crack parameters

The crack profile on the surface of the rib was obtained from experiment data. These are X- and Z-coordinates of the crack from the local coordinate system as shown in the graph above. However, the depth of the crack and the shape of the crack inside the rib remains unknown. Therefore, assumptions are made to solve this problem.

It is assumed that the top view of the crack will have a semi-elliptical crack (the XY-view), such that one

parameter defines the assumed shape, namely the crack depth a. A relatively small initial crack depth of $a = 3$ mm is chosen given that the crack shape in the depth direction may change during the propagation analysis. To determine the inner coordinates of the crack front the following formula are used:

$$\theta = \arccos(x/c) \tag{E.1}$$

$$y = a \cdot \sin\theta \tag{E.2}$$

where:

a	= Crack depth;
c	= Half crack length;
x, y	= Crack front coordinates in mm;
θ	= Position specifier in degrees (see figure 4.2).

Therefore, provided that parameters c and x are known and the value for a is assumed, y-values can be determined. Since X- and Y-coordinates of the crack front are determined, only Z-coordinates along the crack front are to be found. Regarding the initial crack, it is assumed that the Z-coordinates are equal to those of the crack on the surface of the rib. In the following two graphs, the top view and front view of the modelled initial crack are presented.

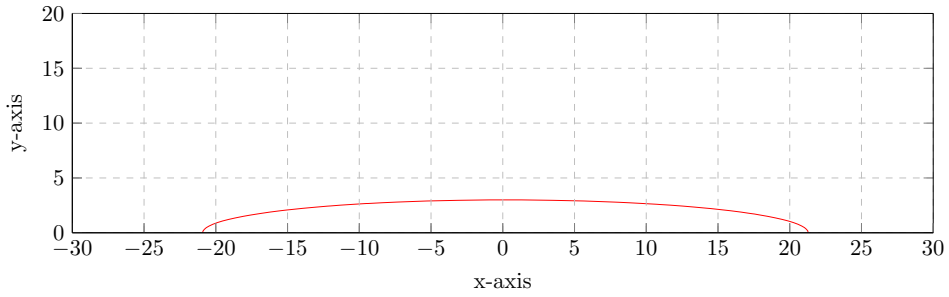


Figure E.2: Top view of initial crack

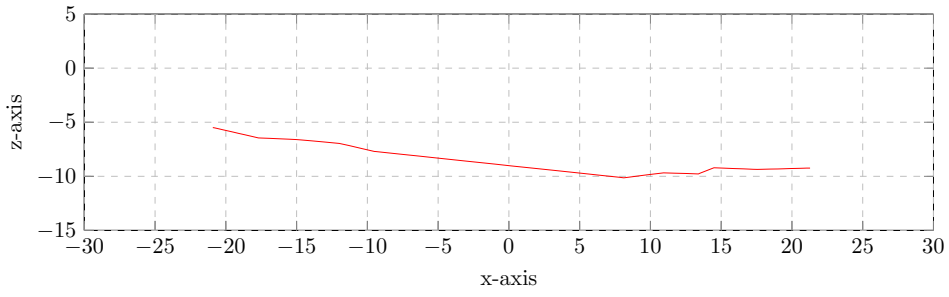


Figure E.3: Front view of initial crack

In the table below, all X-, Y-, and Z-coordinates are listed along the initial crack front. This concludes all information needed for the initial crack in the model. Next, the methodology of crack propagation will be discussed.

X-coordinate in mm	Y-coordinate in mm	Z-coordinate in mm
-20.92	0.00	-5.49
-20.46	0.62	-5.62
-19.10	1.22	-6.03
-17.70	1.59	-6.45
-15.04	2.08	-6.60
-11.98	2.45	-6.96
-9.51	2.67	-7.70
-6.34	2.85	-8.13
-3.12	2.96	-8.58
0.18	3.00	-9.04
4.57	2.93	-9.64
8.12	2.78	-10.14
10.93	2.58	-9.68
13.39	2.34	-9.78
14.49	2.21	-9.21
17.55	1.71	-9.36
19.47	1.22	-9.30
20.83	0.62	-9.25
21.29	0.00	-9.23

Table E.1: XYZ coordinates along initial crack front

E.3. Results starting at mark 6'-6"

All results from the calculations will be discussed in this section. These will include crack trajectories, stress intensity factors (SIF's) and fatigue life assessment and are presented in this order.

E.3.1. Crack trajectory

Firstly, the crack trajectory results are presented. The obtained crack trajectory can be compared to the measurements from the experiments (see section 5.1.2). Below, the XZ-view of the propagating crack is depicted. In table E.3.1 some coordinates along the crack front for all steps are listed. Here, step 0 represents the initial crack.

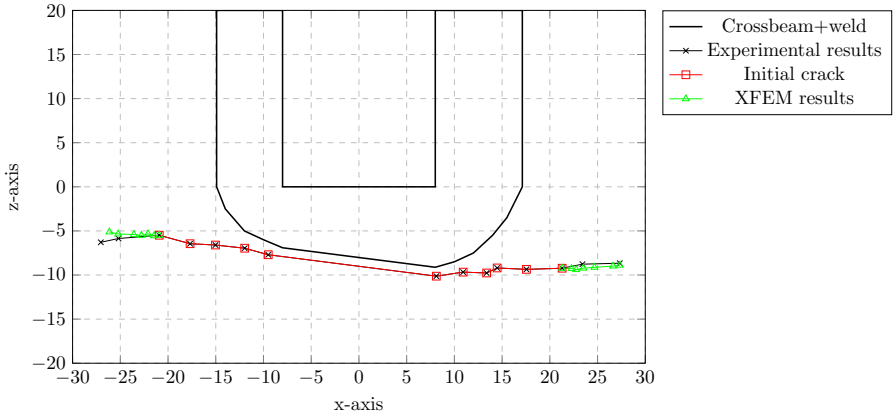


Figure E.4: Results XFEM crack propagation

In the graph above, crack propagation results closely follow the trend of the experimental results. The main validation of the crack trajectory is performed by comparison between the results of the crack on the exterior surface of the rib. Given that these results comply with the measured data, the crack trajectory is assumed to be accurately determined by the analysis.

Table E.2: Coordinates along crack front

Step 0			**Step 1**			**Step 2**		
X	Y	Z	X	Y	Z	X	Y	Z
-20.92	0.00	-5.49	-21.47	0.00	-5.56	-22.09	0.00	-5.34
-20.20	0.78	-5.70	-18.05	1.98	-6.20	-19.64	2.00	-5.91
-17.70	1.59	-6.45	-15.04	2.93	-6.64	-16.47	3.55	-6.36
-15.04	2.08	-6.60	-12.03	3.50	-7.04	-13.29	4.38	-6.73
-11.98	2.45	-6.96	-9.02	3.89	-7.47	-10.12	4.78	-7.17
-9.51	2.67	-7.70	-6.01	4.16	-7.95	-6.95	4.95	-7.70
-4.20	2.93	-8.43	-3.00	4.32	-8.45	-3.77	4.99	-8.27
0.18	3.00	-9.04	0.01	4.36	-8.94	-0.14	4.97	-8.85
8.12	2.78	-10.14	3.02	4.29	-9.34	3.03	4.95	-9.22
10.93	2.58	-9.68	6.03	4.11	-9.60	6.20	4.92	-9.40
13.39	2.34	-9.78	9.04	3.86	-9.68	9.38	4.87	-9.38
14.49	2.21	-9.21	12.05	3.53	-9.58	12.55	4.69	-9.23
17.55	1.71	-9.36	15.06	3.06	-9.37	15.72	4.19	-9.06
20.83	0.62	-9.25	18.07	2.28	-9.17	18.90	3.04	-9.01
21.29	0.00	-9.23	22.27	0.00	-9.27	22.76	0.00	-9.39

Step 3			**Step 4**			**Step 5**		
X	Y	Z	X	Y	Z	X	Y	Z
-22.79	0.00	-5.51	-23.57	0.00	-5.41	-25.23	0.00	-5.35
-19.66	3.27	-5.87	-20.16	3.53	-5.78	-21.38	3.72	-5.62
-16.43	4.55	-6.24	-16.81	4.81	-6.15	-17.82	4.98	-5.96
-13.20	4.93	-6.67	-13.47	5.16	-6.58	-14.27	5.32	-6.42
-9.97	5.10	-7.16	-10.12	5.27	-7.09	-10.72	5.44	-6.97
-6.74	5.27	-7.70	-6.78	5.40	-7.66	-7.17	5.58	-7.56
-3.52	5.43	-8.24	-3.43	5.54	-8.21	-3.62	5.74	-8.12
0.17	5.51	-8.76	-0.09	5.64	-8.67	-0.07	5.83	-8.57
3.40	5.45	-9.07	3.25	5.62	-8.98	3.99	5.81	-8.88
6.63	5.32	-9.21	6.60	5.52	-9.10	7.54	5.69	-8.96
9.86	5.20	-9.18	9.94	5.39	-9.06	11.59	5.54	-8.86
13.09	5.13	-9.02	13.29	5.27	-8.90	15.14	5.42	-8.68
16.32	4.94	-8.84	16.63	5.04	-8.74	18.69	5.12	-8.55
19.55	4.02	-8.89	19.98	4.22	-8.71	22.25	4.04	-8.60
23.51	0.00	-9.22	24.68	0.00	-9.13	26.62	0.00	-8.99

In the figures below, the front, top and isometric views of the propagating crack are visualised. It should be noted that the rib has a thickness of 6 mm. As can clearly be seen from the top view, the crack hardly propagates in the through thickness direction from step 3. The reason for this is that the crack is already almost through the thickness of the rib. As a result, low stresses occur at that location compared to the stresses at the crack front close to $y = 0$. As will be seen in the next section, the stress intensity factors are also comparably low at the middle of the crack, which corresponds to small crack extension as we see on the top view plot. From the front view, we see that the crack tends to propagate in the upwards direction along the entire crack front. As no experimental data was available for this, such results cannot be validated.

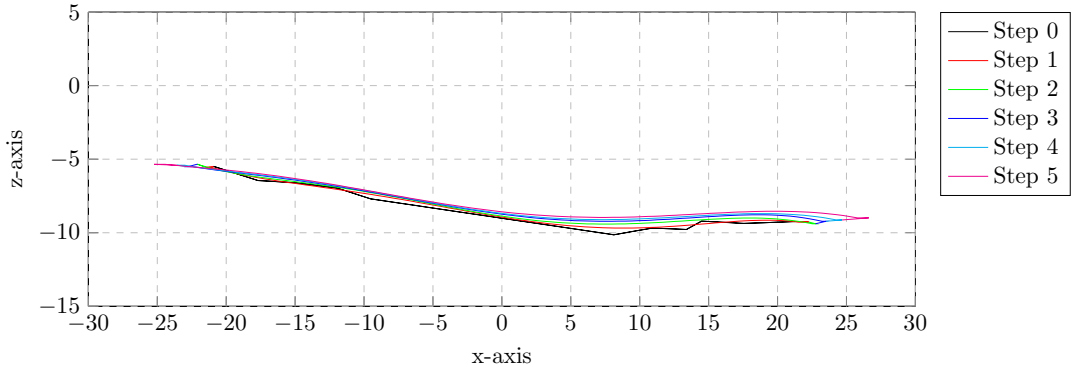


Figure E.5: Front view of propagating crack

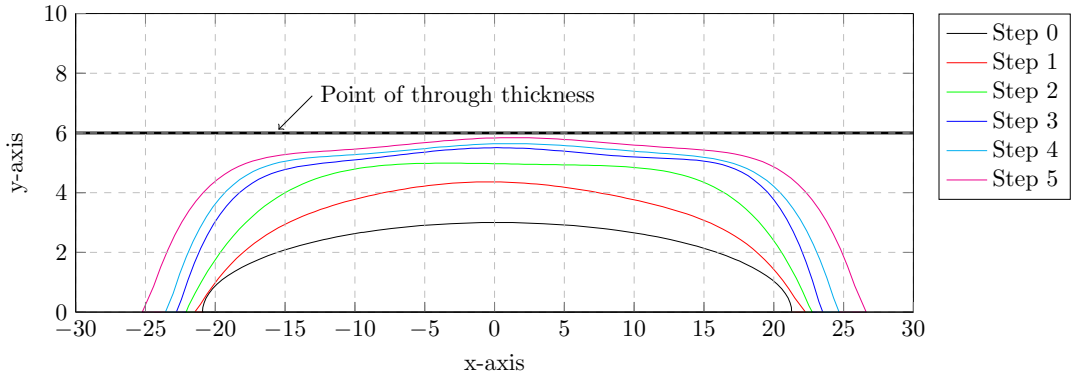


Figure E.6: Top view of propagating crack

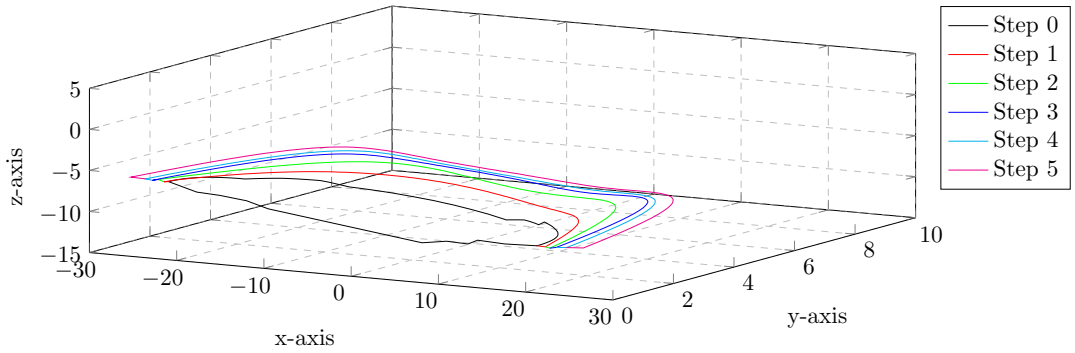


Figure E.7: Isometric view of propagating crack

E.3.2. Stress intensity factors

In the graphs below, the stress intensity factors are plotted for each step in the crack propagation analysis. In the data presented, smoothing is used as explained in section 5.3. Here, a 9^{th} order polynomial is used to fit the data and remove the oscillation in the results along the crack front as well as disregarding inaccurate surface values.

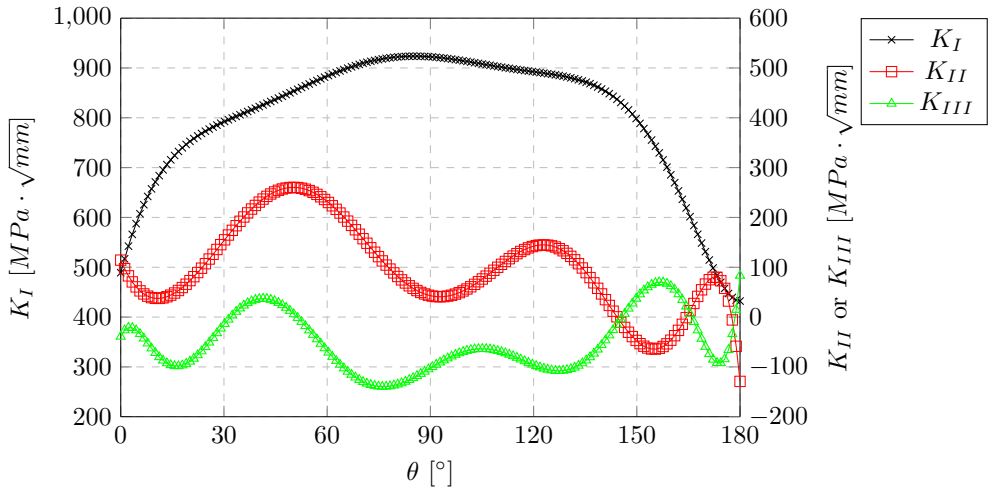


Figure E.8: Stress intensity factor results step 0 - initial crack

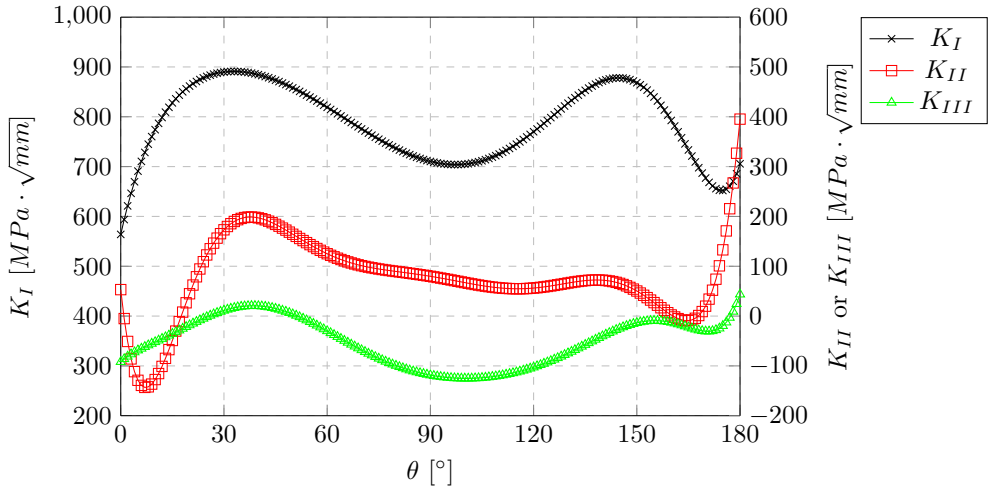


Figure E.9: Stress intensity factor results step 1

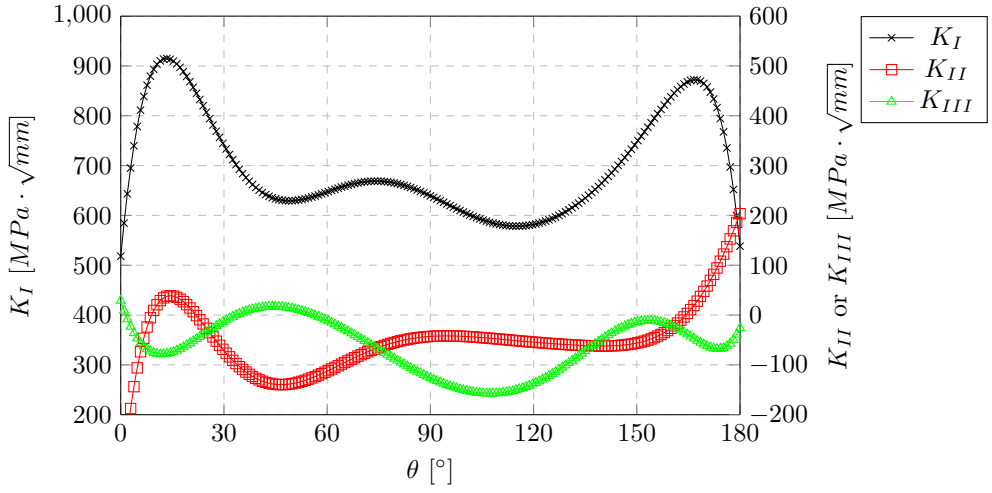


Figure E.10: Stress intensity factor results step 2

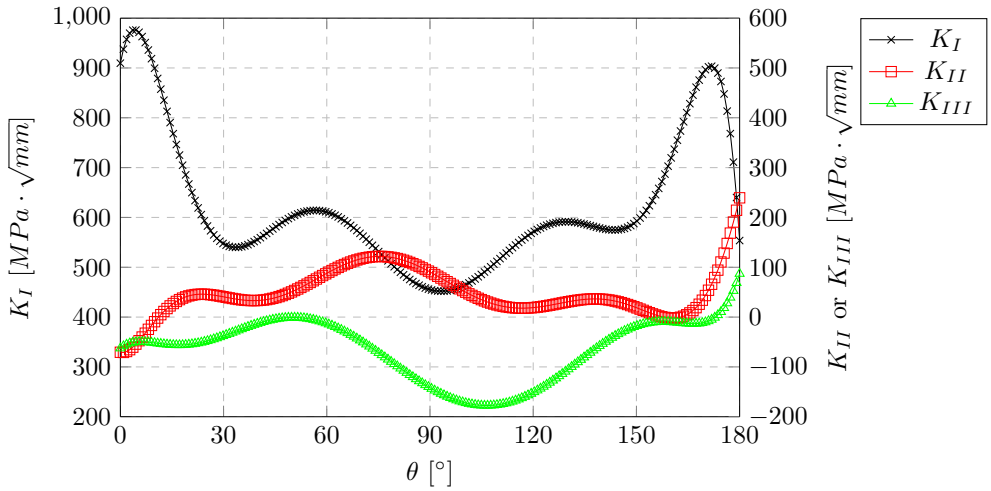


Figure E.11: Stress intensity factor results step 3

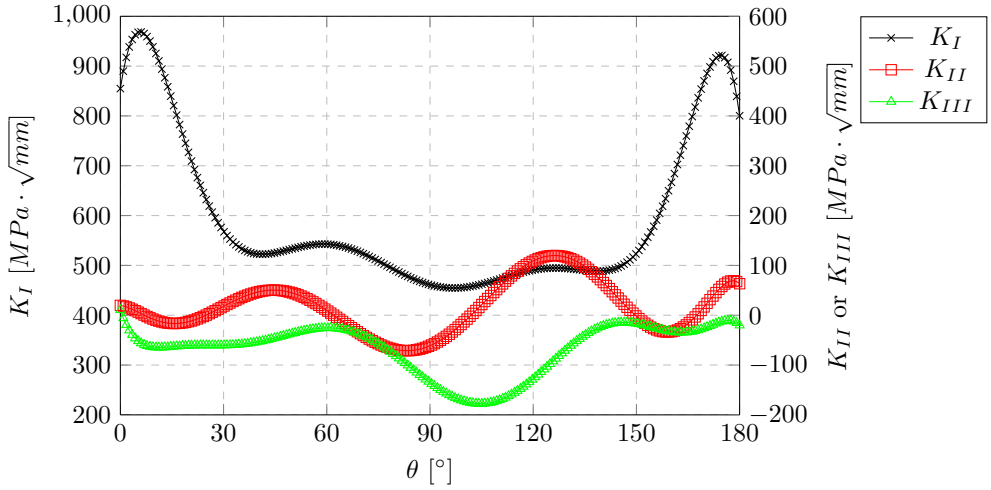


Figure E.12: Stress intensity factor results step 4

As can be seen from the graphs above, the values of K_I are significantly higher than those of K_{II} and K_{III}, leading to a more dominant mode I cracking. Furthermore, for K_I between $\theta = 30°$ and $\theta = 150°$, the values drop significantly while those at the boundaries are still much higher. As a result, the crack tends to propagate (starting from step 2) much more in the longitudinal direction of the rib than in thickness direction. This was also discussed in the results of crack trajectories (see section E.3.1).

E.3.3. Fatigue life assessment

In this section the fatigue life assessment will be discussed. For each step, the number of load cycles are selected and the crack extension is calculated. Subsequently, the cumulative number of cycles is determined as well as the crack length. In the crack propagation analysis, the Paris law was implemented as crack propagation relation:

$$\Delta a = \Delta n \cdot C \cdot \Delta K_{eff}^{m} \tag{E.3}$$

$$\Delta K_{eff} = \sqrt[4]{(\Delta K_I)^4 + 8(\Delta K_{II})^4} \tag{E.4}$$

where:

Δa = Crack extension in mm;
Δn = Number of cycles responsible for Δa;
ΔK_I = Stress intensity factor range corresponding to mode I;
ΔK_{II} = Stress intensity factor range corresponding to mode II.

Parameters C and m have been chosen as $3.98 \cdot 10^{-13}$ and 2.88 respectively, as recommended in BS7910 [13]. In the table below, the fatigue data is listed for each step in the crack propagation analysis.

Step	Δa in mm	Δn in cycles	Crack length in mm	Cumulative number of cycles
0	-	-	42.83	2.30E+05
1	1.54	10,000	44.37	2.40E+05
2	1.16	10,000	45.53	2.50E+05
3	1.49	10,000	47.02	2.60E+05
4	1.96	6,000	48.98	2.66E+05
5	3.61	10,000	52.59	2.76E+05
6	1.68	10,000	54.27	2.86E+05

Table E.3: Fatigue life assessment data

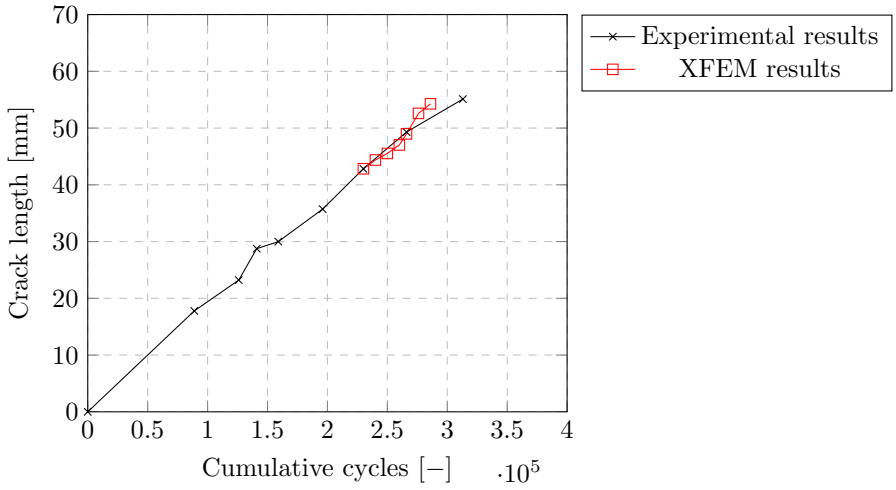


Figure E.13: XFEM results fatigue life assessment

To investigate the effect of the constants C and m in the Paris law, a parametric study is performed. The crack propagation rate is linearly dependent on parameter C and exponentially dependent on m. Given such linear dependence on C it is assumed that the crack shape is retained in the parametric study (as C is changes, Δn is changes accordingly). However, this does not hold for parameter m, as the crack shape may not be retained. Therefore, figure E.14 is used to illustrate the effect of parameter m and should not be used as actual results for fatigue assessment. Such results are shown in the following two graphs. As can be seen, the values chosen initially for C and m, namely $3.98 \cdot 10^{-13}$ and 2.88 respectively, resemble the experimental values quite closely.

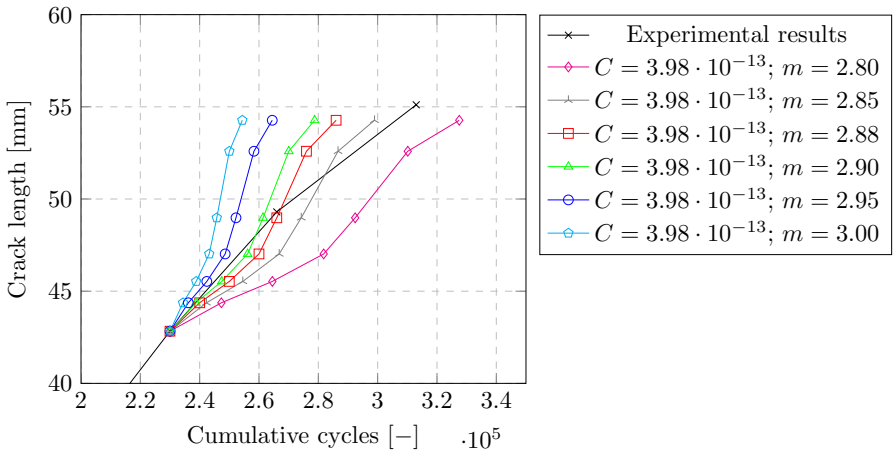


Figure E.14: Sensitivity study to Paris constant m

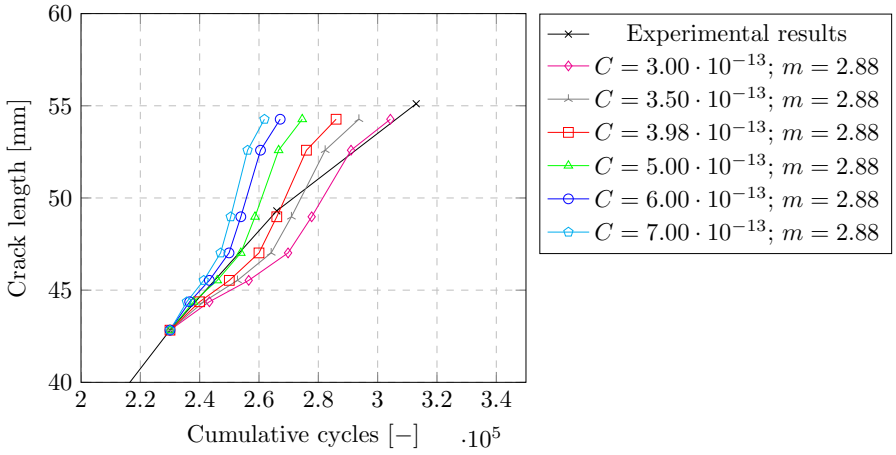


Figure E.15: Parametric study to Paris constant C

The parametric study of C shows that values of $3.50 \cdot 10^{-13}$ and 2.88 for C and m respectively show highest correlation with the experimental results. Therefore, these results are recommended for this specific application. It should be noted however that these parameters are influenced by the material used and can thus vary per application. In addition, the depth of the initial crack was assumed as well as the shape.