Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Fredrik Hiorth

Design and failure assessment of nature-inspired interlocking joints

Master's thesis in Mechanical Engineering
Supervisor: Seyed Mohammed Javad Razavi
June 2021

NTNU
Norwegian University of
Science and Technology

Fredrik Hiorth

Design and failure assessment of nature-inspired interlocking joints

Master’s thesis in Mechanical Engineering
Supervisor: Seyed Mohammed Javad Razavi
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

Nature use geometry in ingenious ways to exploit otherwise unsuited materials to fit special use cases. Mechanical design can be improved by learning the uses and limitations of these exploits. One such feature is sutures. Sutures are used in nature to create strong connections. The static properties of a simple interlocking suture are tested before studying fatigue properties of such a suture. The suture becomes increasingly stronger with higher interlocking angles. At small angles, the tab will pull out of the suture with relatively little force. At higher angles, the load is distributed over a smaller cross-section so the stress throughout the neck is higher. The high angle sutures can handle larger loads but is subjected to more critical modes of failure. The research done in this thesis suggest lower angle sutures are better suited to withstand fatigue and higher angle sutures are better suited for high load applications without cyclic loading. This should be researched further to get a definitive answer.

Future work should focus on different geometries and properties including hierarchical structures, clearance, friction, different material properties, mixing materials, and using rows of sutures to construct a metamaterial.

Sammendrag

Naturen bruker geometri på smarte måter for å bruke ellers uegnete materialer til å passe spesifikke bruksområder. Design kan forbedres ved å lære om bruken og begrensningene til disse naturlige fenomenene. En av disse er sutur. Suturer brukes i naturen til å lage sterke sammenføyninger. De statiske egenskapene til en enkel sammenlåsende sutur blir testet først, og så testes utmattingsegenskapene. Suturen blir sterkere og sterkere jo høyere sammenlåsingsvinkelen er. Ved små vinkler vil fliken på suturen kunne trekkes ut av suturen med relativt lite kraft. Ved større vinkler vil kraften fordeles over et mindre tverrsnitt så spenningen i nakken på suturen vil være høyere. Ved høyere vinkler kan altså suturene ta imot større krefter, men de er mer utsatt for kritiske feilmoduser. Forskningen gjort i denne oppgaven peker på at suturer med lavere vinkler er bedre egnet til å motstå utmatting og suturer med høyere vinkler er bedre egnet til å motstå høye laster uten syklisk belastning. For å være sikker kreves det mer forskning på området.

Fremover burde dette forskningsområdet sette søkelys på forskjellige geometrier og egenskaper, blant annet hierarkiske strukturer, klaring, friksjon, forskjellige material egenskaper, miksing av materialer og bruk av suturer for å lage et metamateriale.

List of figures

List of plots

List of tables

Nomenclature

FEA – Finite Element Analysis
UTM – Universal Testing Machine
DIC – Digital Image Correlation
EDM – Electric discharge machining
PMMA – Polymethyl Methacrylate
Male suture part – The part of the suture structure with a protruding segment
Female suture part – The part of the suture structure that has a recess
Suture tab – The protruding segment of male suture part
Suture neck – The area with the smallest cross section in the tab
Suture line – The line that defines the border between the male and female parts

Table of Contents

1 Introduction

1.1 Background and motivation

Aerospace and marine structures use a wide variety of different lightweight materials for applications that demand high strength-to-weight ratio. Designs can require the use of different materials in the same structure because various parts of the structure need distinct properties. Many conventional methods of joining, such as welding, utilize shared properties between the parts to be joined. A challenge with using dissimilar materials is creating a strong and lightweight bond between the parts, that will last the structures lifetime.

Materials and structures in nature include joining mechanisms that by today have not been fully recovered and used in real life mechanical design methods. One such method is suture joints. These joints use hard and brittle materials with sutures. The sutures make the brittle materials less prone to failing due to small local deformations. If the mechanical properties of sutures are better understood they can be used in engineering design to make connections stronger, more durable, and less prone to fatigue.

1.2 Problem description

The aim of this project is to explore interlocking joints found in nature and use this information to propose a design and subsequentially produce, test and assess the design. To the best of the author's knowledge, the mechanical response of this type of structures under fatigue loading has not been studied in literature. This thesis will present how fatigue affects suture geometries and give insight into how to design suture connections against fatigue.

1.3 Report structure

The thesis has an introduction to familiarize the reader with the background and motivation for choosing this topic, as well as an introduction to the problem which the thesis aims to solve. Next, a literature review of existing literature on nature inspired structures and interlocking joints is presented. Then, the method for how the FEA and experimental testing is conducted is shown. Further, the results from FEA and are presented and discussed. Finally, a conclusion is given based on the results and discussion and recommendations for future research is given.

2 Literature review

Nature has many ways to avoid failure of parts. Through evolution nature manage geometry to avoid high local stress, use fibers to strengthen components in the direction of the load and repair damaged structures (Taylor, 2015). Parts used in mechanical engineering can be improved by looking at and mimicking structures in nature. There are an estimated 8.7 million different animal species on earth (Mora et al., 2011). Many have similar structural components because they face similar challenges. These structural components can be divided into eight categories; fibrous, helical, gradient, layered, tubular, cellular, suture and overlapping (Naleway et al., 2015).

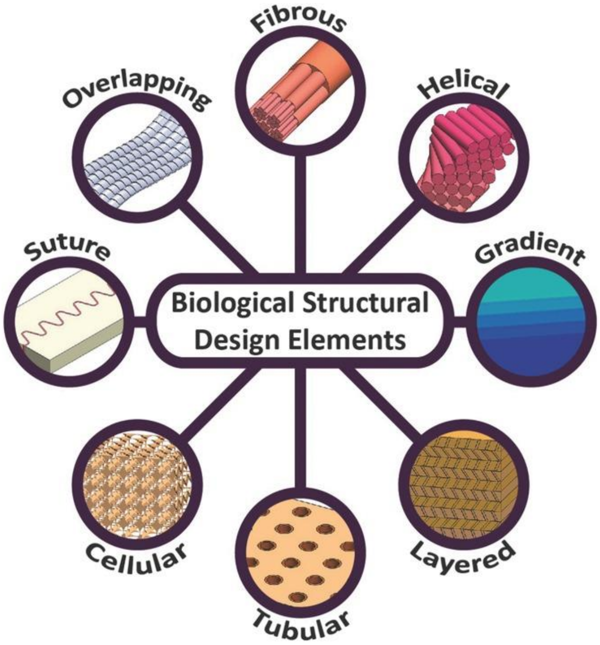


Figure 1: Illustration of the eight different biological structural design elements recognized through study carried out by (Naleway et al., 2015).

One of these structural elements is sutures. A suture consists of two parts with opposing faces with uneven surfaces that can interlock with each other. In nature, the parts are often made of stiff material with a thin layer of a compliant material, with several orders of magnitude lower elastic modulus, in between. This makes the component strong while maintaining high toughness (Zhang et al., 2012).

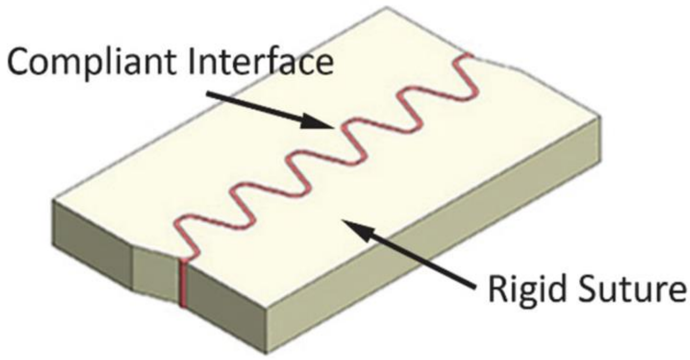


Figure 2: Schematic of typical non-interlocking suture connection (Naleway et al., 2015)

Strength and toughness are usually mutually exclusive in homogenous materials. Toughness is a materials resistance to fracturing. A material's ability to yield in small regions is crucial to avoid failure because it releases locally concentrated stress that would otherwise cause fracture (Ritchie, 2011). Biological components fix the problem of the brittle materials by introducing mechanisms such as sutures with compliant interface, macromolecular deformation, chemical bond breakage and biomineral crystal imperfections at the atomic scale among others (Lin et al., 2014) (Huang et al., 2019).

Sutures are seen in nature from micro to macro level. They can offer strong and tough connections between separate parts and some flexibility in otherwise rigid parts. At the micro scale they are found in several types of diatoms, a type of algae (Bahls et al., 2009) (De Stefano et al., 2009). At the macro scale sutures can be seen in skeleton parts such as turtle exoskeleton (Achrai & Wagner, 2015) and mammalian skulls (Herring, 2008). A recent article, published after the start of this project, reviewing the diabolical ironclad beetle, *Phloeodes diabolicus*, found that it has a single suture going lengthwise along its elytra. This suture is part of what gives this beetle its immense strength-to-weight ratio of 39,000 (Rivera et al., 2020).

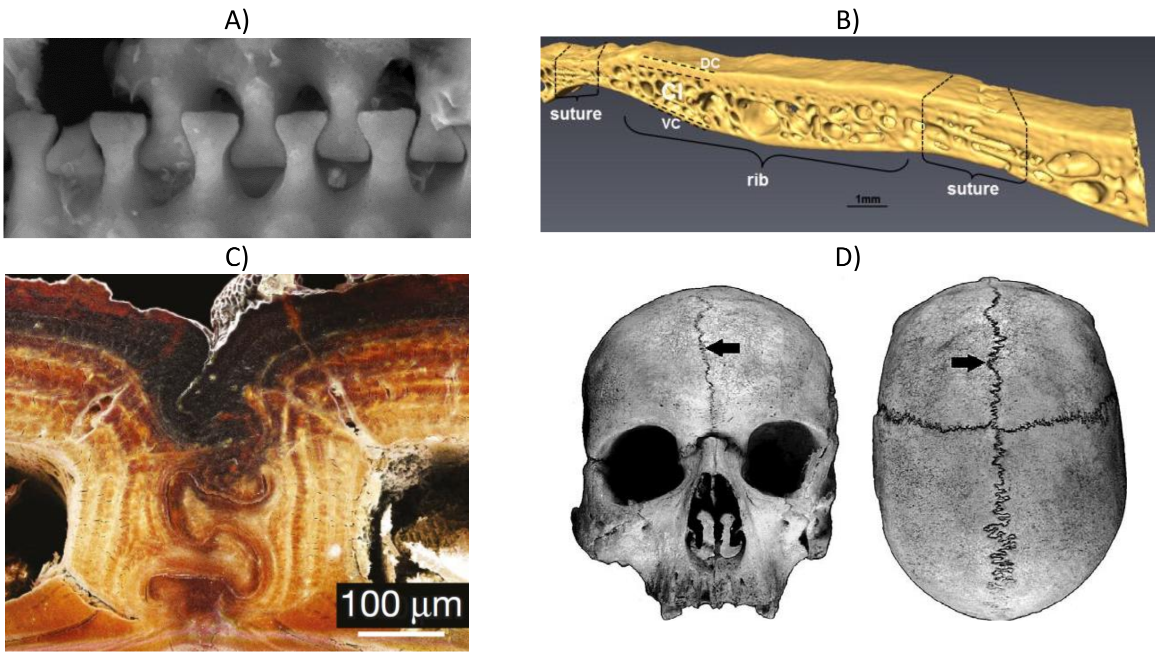


Figure 3: A) Linking of valves in diatom, Aulacoseira canadensis (Bahls et al., 2009). B) Suture in red-eared slider turtle exoskeleton (Achrai & Wagner, 2015). C) Suture in diabolical ironclad beetle, Phloeodes diabolicus (Rivera et al., 2020). D) Sutures in human skull (Skrzat et al., 2004).

Most synthetic composites that try to mimic this to achieve the combination of high strength and high toughness has not been able to reach the same potential as the nature-made composites because of poor connection between stiff and compliant material. The intended use of the composite is important for how the interface between the two materials should be designed. Some composites use geometrical interlocking in the junction between the stiff and the compliant material. The geometrical shape, size, ratio, and hierarchical order has a large impact on the mechanical properties of the composite. Tensile strength can reach 70% of a perfectly bonded case by choosing the right geometry even without bonding nor friction between the materials (Zhang et al., 2012).

Some systems in nature use hierarchical sutures to perform tasks such as carrying load, taking up energy, breathing, growing, or moving. The order of hierarchy can determine how well the suture can perform these tasks and can be fitted to suit a special case. Changing the order of hierarchies can change the performance of the suture by orders of magnitude. By manipulating order of hierarchy in sutures in designed parts the sutures can be made to fit different cases depending on needs, available space, and mass. Hierarchical sutures also have the added benefit that they can prevent failure in the larger parts of the hierarchical structure by allowing small failures in the smaller parts and thereby creating a resistance against defects (Li et al., 2012).

Sutures without a compliant material can also improve material toughness. “Jigsaw”-like interlocking sutures with homogeneous stiff material has been tested to find optimal shape and friction for pull-out and fracture. The stress in the suture and force needed to pull it out was found analytically and through finite element analysis (FEA) by Malik et al. (2017). The resistance to pull-out is found to be higher when the interlocking angle α is higher, and friction is higher. Stress in the suture tab increases when friction is higher. That stress introduces another mode of failure, namely fracture of the tab. Thus, the optimal design for “jigsaw”-like sutures is a high interlocking angle α and low coefficient of friction. The results from analytical analysis and FEA was confirmed by static testing of 3D printed specimen (Malik et al., 2017).

When designing a component one needs to consider all possible modes of failure. Even though a part is designed to take all the static loading it will be exposed to in its lifetime it can still fail after cyclic loading due to fatigue. A staggering 25% of all engineering components and 55% of all aircraft components fail due to fatigue (Findlay & Harrison, 2002). Thus, components exposed to cyclic loading should be subject to fatigue life studies. The fatigue behavior of the suture in the component must be known to extend fatigue life of the component. The scope of this project is to explore how cyclic loading affects the life of different suture geometries.

3 Method

A numerical and an experimental approach is chosen to evaluate the effect of suture angle on the mechanical performance of the interlocking joint, specifically, the fatigue properties. For this aim, four different geometries are studied using both FEA and experimental testing. The experimental testing consists of static testing and cyclic loading.

3.1 Overview of geometry

The suture researched in this report consists of two parts. One upper part and one lower part. They are geometrically interlocked with each other. The line that defines the border between the two parts is the suture line and the protruding segment of the lower part is the tab. The angle α define the geometry and the radius r define the size as seen in Figure 4.

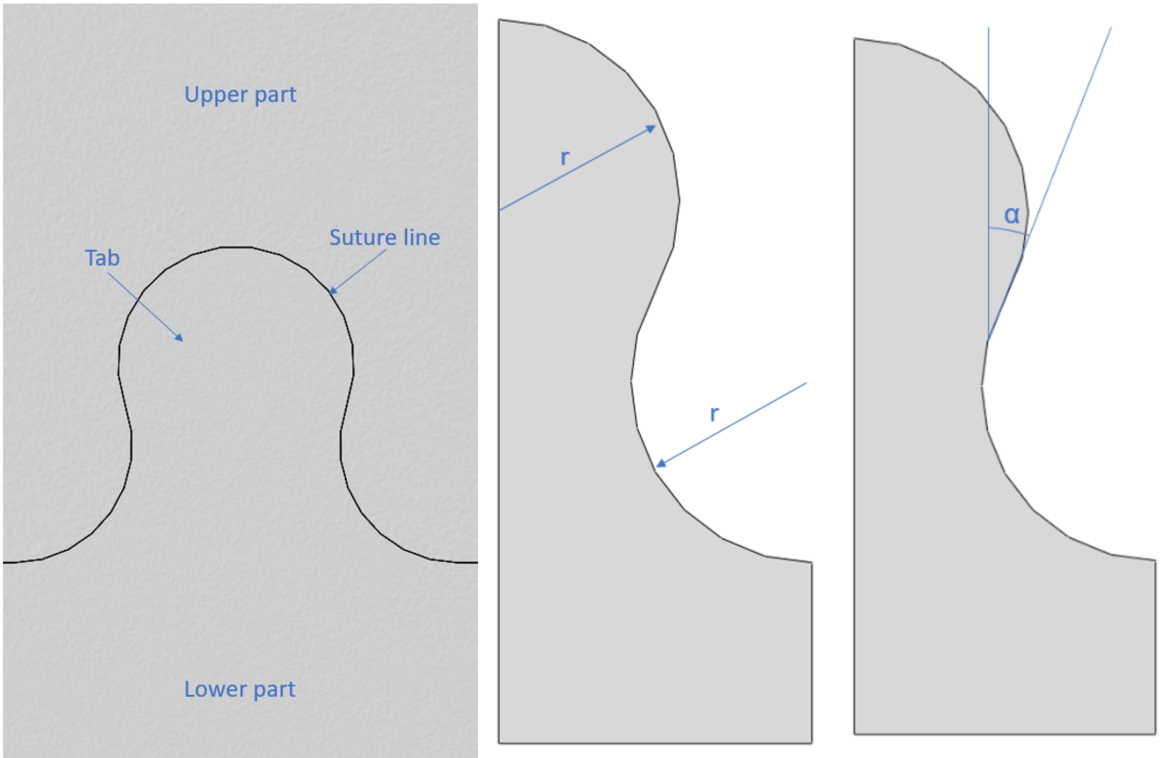


Figure 4: Defining features of the suture. α defines the geometry and r defines the size.

3.2 Numerical analysis

The numerical analyses are performed in Abaqus. A python script is run for each analysis with the angle α as the only changing variable. These analyses have a constant displacement rate applied to them with varying load as the two parts are moved away from each other.

Earlier in this project some FEA analyses were conducted to find where stress occur in the suture. Some of the results from those analyses will be reproduced in the results and discussion section.

3.2.1 Script

A script is written for this project to minimize the effort and ensure that each case is simulated equally. The script can be found in Appendix B – Python script for FEA. The script run in a similar sequence to Abaqus CAE. Inspiration for the script was found in *Python scripts for Abaqus: Learn by example* (Puri, 2011). First the parts are constructed by sketching the arcs and lines outlining the parts. Then the parts are created. Second, a section is created for each of the parts and the material is made and added to the sections. A step where the load will be applied is made and field output and history output are requested for the step. The variables chosen for the field output is S, E, U, RF and CF. The contact between the two parts, load and boundary conditions are defined. Then, the mesh and mesh bias are specified, and the mesh is generated. The job is run and all postprocessing is handled by the script as well as saving the relevant data from the analysis in a separate file.

3.2.2 Model

Four cases are studied in this report. One model is made for each of the cases with 10-, 20-, 30- and 40-degree angle. Only half of the specimen is modelled because it is symmetric along the x-axis, so the stress is symmetric too.

The model is created in Abaqus with a radius of 2 mm and an angle α corresponding to each of the four cases. The model has a thickness of 2 mm.

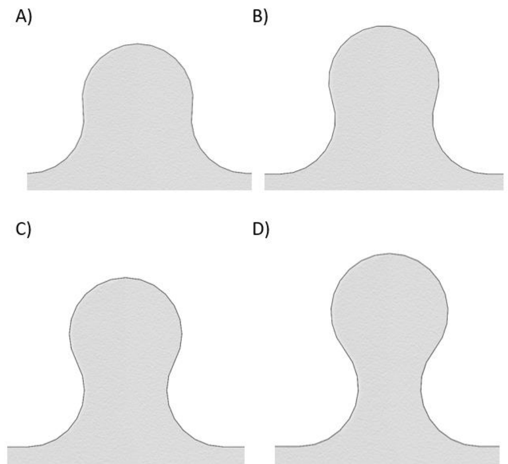


Figure 5: The cases are A) 10-degrees B) 20-degrees C) 30-degrees D) 40-degrees.

3.2.3 Boundary conditions and load cases

The male part is fixed at the bottom with encastre and has symmetry in x direction on both sides. This simulates that the suture line continues in both negative and positive x direction. The female part also has symmetry in both x directions, but it has a displacement boundary condition connected to a point on the top edge. The point is coupled to the rest of the top part.

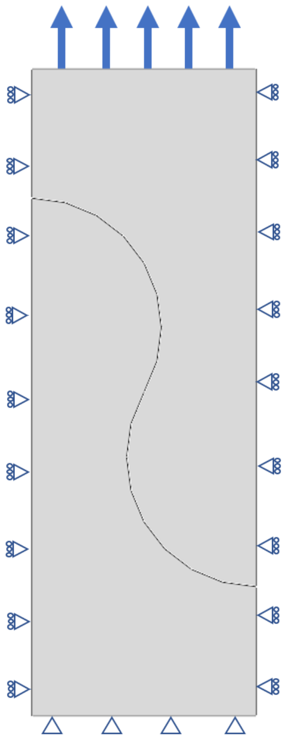


Figure 6: Boundary conditions and direction of load/displacement.

The material used for the analysis is aluminum with a density of 2700 kg/m^3, Young's modulus of 70 GPa and Poisson's ratio of 0.32 and plastic strain values for aluminum alloy 5754 found in "Investigation on the forming limits of 5754-O aluminum alloy sheet with the numerical Marciniak–Kuczynski approach" by Ma et al. (Ma et al., 2018).

The contact between the two parts is penalty surface-to-surface with finite sliding with varying friction for different tests. Surface-to-surface contact in Abaqus consider the shape of both contact surfaces and allows self-contact. Some penetration of individual nodes can occur but is largely mitigated with this discretization.

3.2.4 Mesh

The mesh is built up from mostly linear quadrilateral (CPS4R) elements and some linear triangular (CPS3) elements. The mesh is generated in Abaqus using the "free" technique and "advancing front" algorithm. The widths of unbiased elements are 0.1 mm. There is a bias along the suture line that makes the element size smaller towards the middle of the suture. The number of nodes along the suture line is 200 and the ratio between the largest and smallest elements is 100. The smallest elements are at the contact point in the middle of the suture line. They are 5.44×10^{-4} mm.

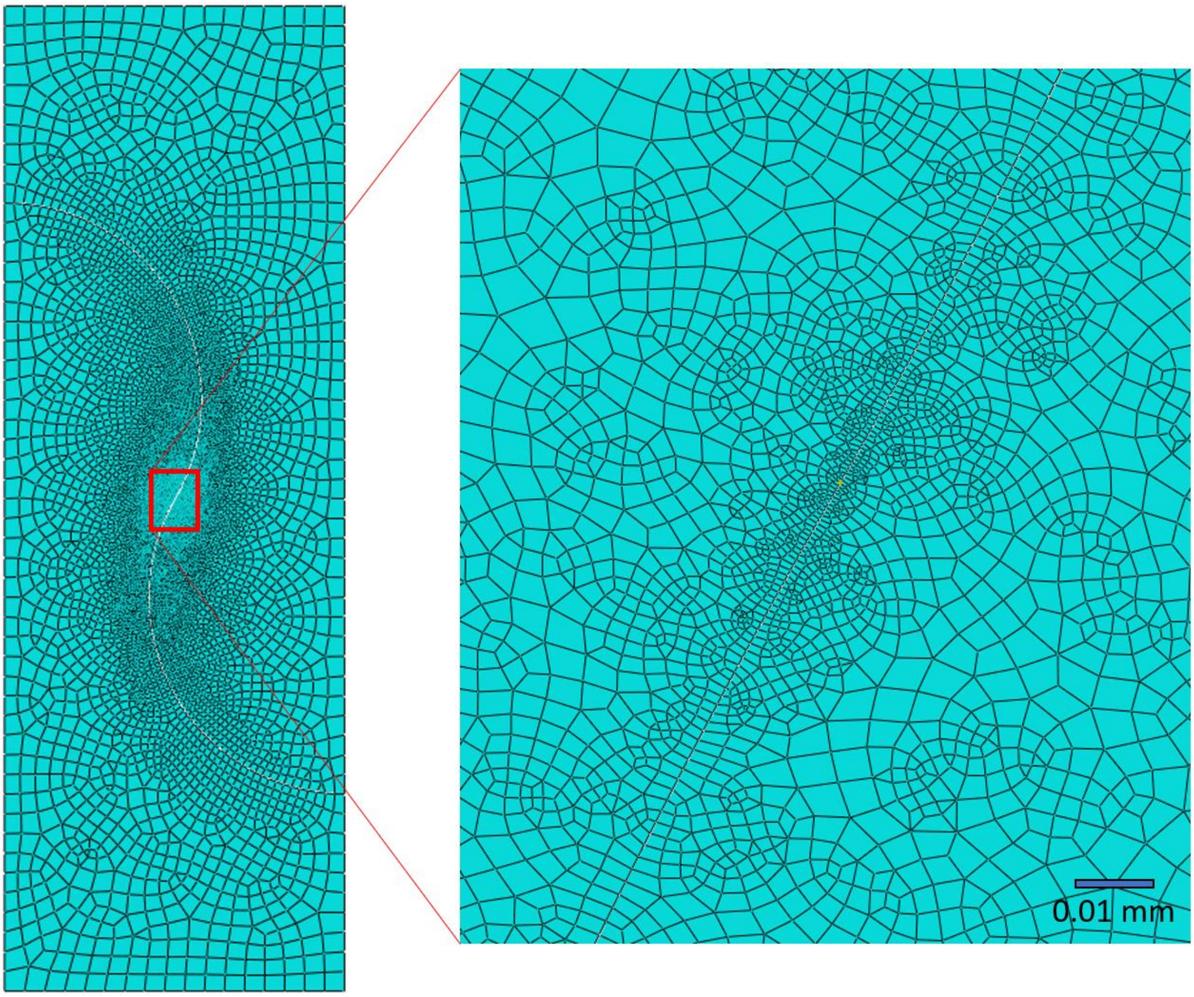


Figure 7: Typical mesh used in FEA. This one is from the 30-degree case.

3.3 Experiment

Experiments are conducted on aluminum specimens to determine the maximum static load and fatigue life of the different suture geometries. The specimens used in the experiments have a width and height of 40 mm, not including the tab, with the suture having a radius of 2 mm. The specimens are subjected to controlled tensile in the static tests and cyclic loads in the fatigue tests. These tests are performed in an Instron 10 kN ElectroPuls and an Instron 250 kN.

3.3.1 Electric discharge machining (EDM)

The experiments are conducted on aluminum-alloy specimens. They are cut from 2 mm thick rolled 5754 aluminum-alloy. The main features are cut with electric discharge machining (EDM) to ensure a good fit between specimens.

EDM is a high-precision type of machining that uses no-conventional methods of material removal. Material is removed by a series of sparks that vaporize material from the work piece through a wire. The work piece and the wire must be conductive to generate the spark. The wire and work piece does not touch so vibrations usually found in traditional machining can be avoided. This way the surface finish can be very precise (Mohd Abbas et al., 2007).

3.3.2 Static testing

Earlier in this project some static experiments were performed on similar sutures made from laser cut PMMA sheet. These experiments were carried out on the UTM, MTS Model 42 with a sampling rate of 10 Hz and a displacement rate of 0.085 mm/s.

The new experiments are carried out on Instron ElektroPuls 10 kN and Instron Hydraulic 250 kN. The sampling rate is 50 Hz and the displacement rate is 2 mm/min. The specimens are produced from aluminum alloy sheet. The male part is fastened to the bottom clamp in the machine and the female part is interlocked with the male part. It is important that the specimens are fastened vertically to avoid sideways loading. When clamping in the female specimen the suture will experience a small load. The clamp is then moved manually until the load is zeroed in order to ensure that the test starts with no load.

3.3.3 Digital image correlation (DIC)

DIC is a type of software used to measure displacement and strain by comparing pictures taken during the deformation of a specimen. The DIC analysis software divides the area of interest into subsets. Within each subset the software recognizes a dot on the specimen as a cluster of pixels and track the movement of those pixels compared to the first picture, where the specimen is not under load. DIC works best when these dots are randomly shaped and distributed (McCormick & Lord, 2010; Mobasher, 2016).

The specimens used in the static tests are prepared for DIC. Since rolled aluminum has a polished reflective surface it is not suited for DIC analysis without any modification. The specimens are first painted with a coat of white acrylic paint, then black dots with a size of between 0.001 mm and 0.1 mm are applied with an airbrush to achieve a random distribution of black dots.

A camera, positioned directly in front of the specimens, takes five pictures every second while the static test is running. VIC-2D is used to perform the DIC analysis. The subset size is 49 and the step size is 1. This means the software will compare the movement of all the pixels within a 49 by 49 square on the undeformed shape to a similar square on the deformed shape.

3.3.4 Fatigue testing

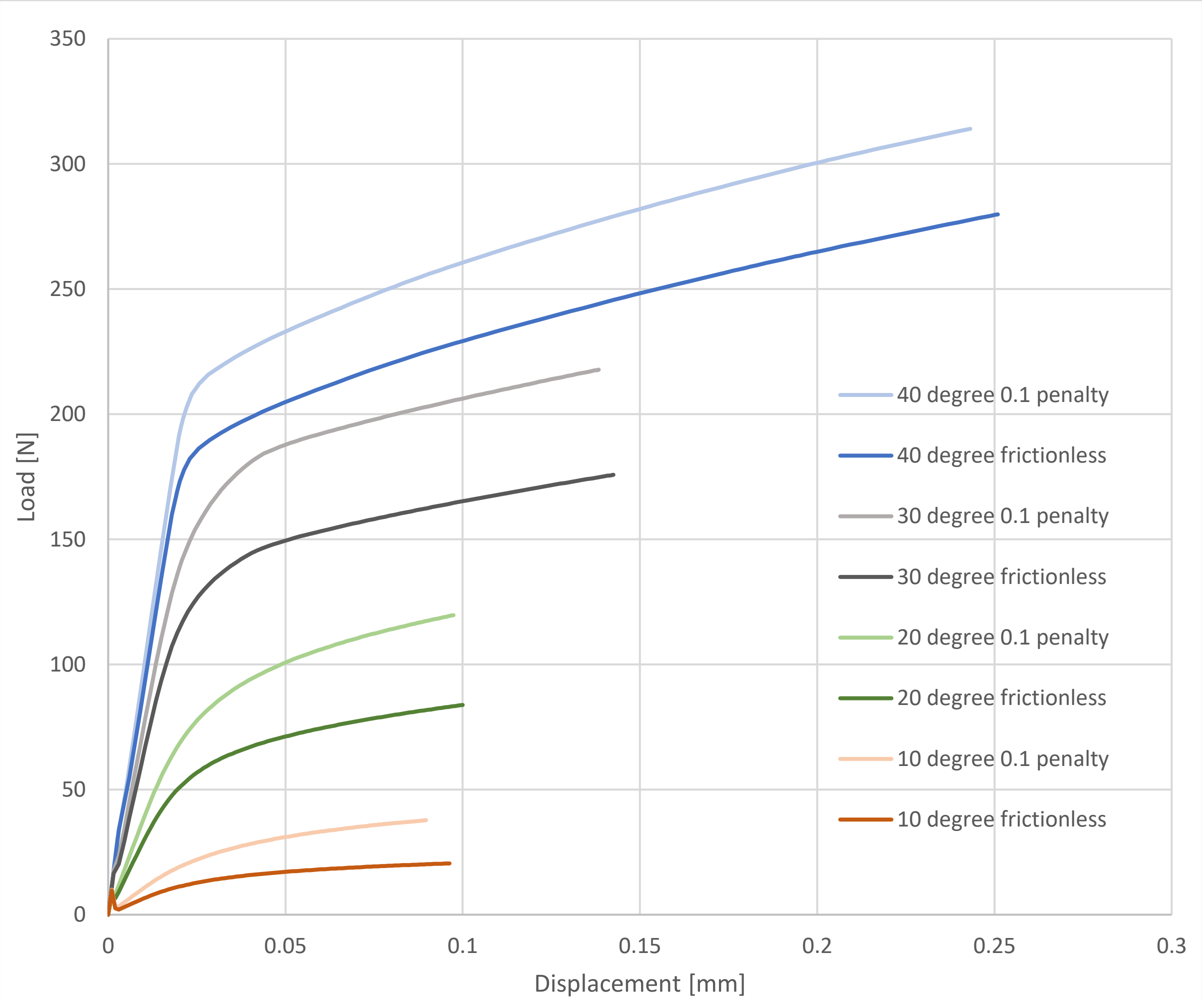
Most of the specimens are tested in a Instron 10 kN ElectroPuls. Because of availability some of the specimens are tested in Instron 250 kN machine. Use of the 250 kN machine is sub-optimal as it is not dimensioned for the subtle load changes needed for the tests. For the first test the specimens experience a cyclic load varying between 70% and 7% of the peak load from the static test of the same geometry. The frequency of cycles is 40 Hz. For the geometries that fail before 1000000 cycles the load is reduced and for the geometries that run out 1000000 cycles the load is increased.

4 Results and discussion

The result and discussion are separated into three parts. The first part will show the results from and discuss the FEA and compare it to the results from earlier FEA. The second part will show the results from and discuss the static experiments. It will also present and discuss DIC analysis of the static tests. The last part will cover the result and discussion of the fatigue tests.

4.1 FEA

The load increases as the parts are moved away from each other. The load also increases more for higher suture angles. The sutures with the lowest angles must deform less and has a smaller contact area than the higher angle sutures as seen in Plot 1.



Plot 1: Load and displacement from FEA for the four different geometries with and without friction.

The analysis does not account for damage mechanics so when displacement in an element becomes too large the analysis stops.

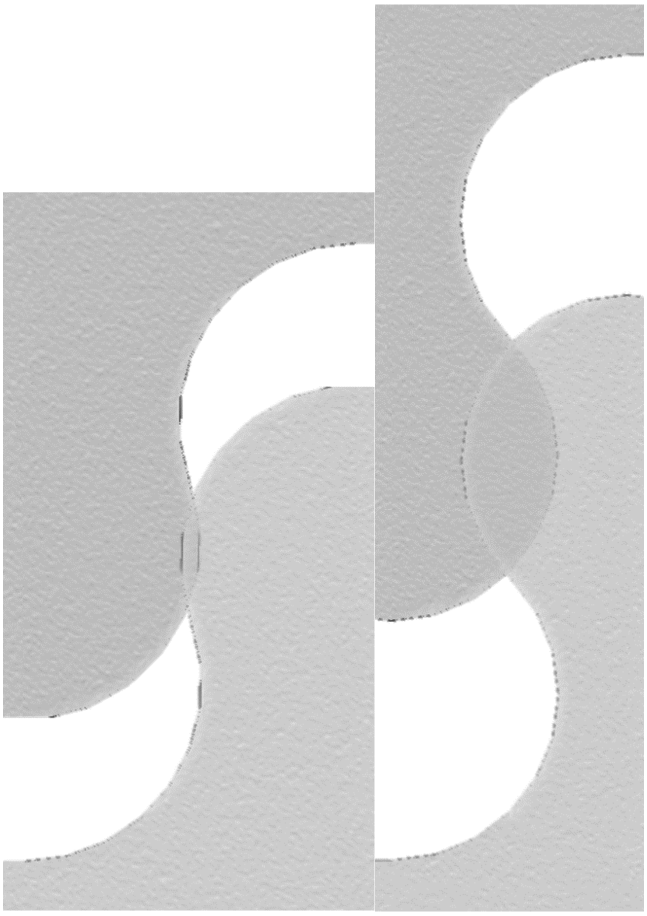
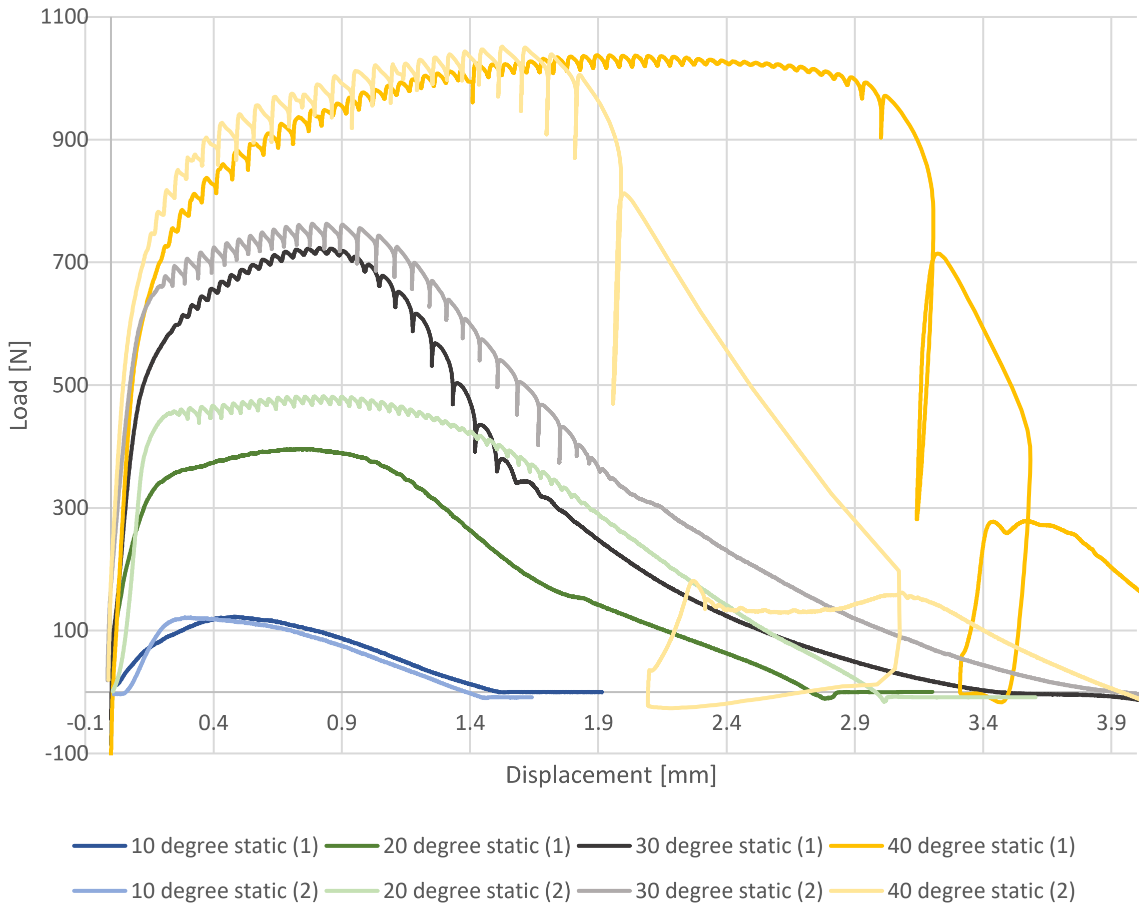


Figure 8: A 20-degree suture must deform less to pull out the male part from the female part than a 40-degree suture.

In the FEA with constant load performed earlier in this project it was found that higher angle sutures have higher stress in the smallest cross-section of the neck in the suture. It was also found that dividing the maximum negative tangential stress with the maximum normal stress yields a higher ratio for lower angles. This indicates that lower angle sutures have a higher chance of sliding than higher angle sutures.

4.2 Static testing

Two suture specimens from each geometry are tested. The results from the static testing of each suture are gathered and presented in the same graph to highlight differences between the cases.



Plot 2: Load and displacement from static tests for two specimens of each geometry.

The 10-degree suture has a peak load of 123 N in the first static test and 121 N in the second. The average peak load for the two cases is 122 N. The 20-degree suture has a peak load of 396 N in the first static test and 482 N in the second test. Thus, the average peak load for the two cases is 439 N. There are not enough specimens produced for two static tests and all the necessary fatigue tests. Consequently, the specimens used in the second static test for both 10-degree and 20-degree tests had already run out a fatigue test. The specimens only had very small local deformations before they were tested statically. The difference between the two 20-degree suture tests is unacceptably large and is considered inaccurate. The 30-degree suture has a peak load of 723 N in the first static test and 763 N in the second. The average static peak load is 743. For the 40-degree suture the peak load for the first test is 1037 N and for the second test it is 1051 N, making the average peak load 1044 N. For both tests of the tab bent out of plane. In Plot 2 both the 40-degree sutures show a large decrease in load with a small decrease in displacement. At this point the specimens deform drastically, stretching the neck of the suture and causing it to bend. The rectangular shapes in Plot 2 at the end of the 40-degree suture tests are a result of the tab moving out of the plane of the specimens.

	10-degree suture	20-degree suture	30-degree suture	40-degree suture
First static test	123 N	396 N	723 N	1037 N
Second static test	121 N	482 N	763 N	1051 N
Average peak load	122 N	439 N	743 N	1044 N

Table 1: Peak load from static tests.

As showed earlier, for the lower angle sutures the stress is concentrated at the contact point on the sides, while the higher angle sutures have a high stress distributed through the neck of the suture.

In the 20-degree suture the tab is only locally deformed at the suture line, while the 40-degree suture is deformed over the whole neck area.

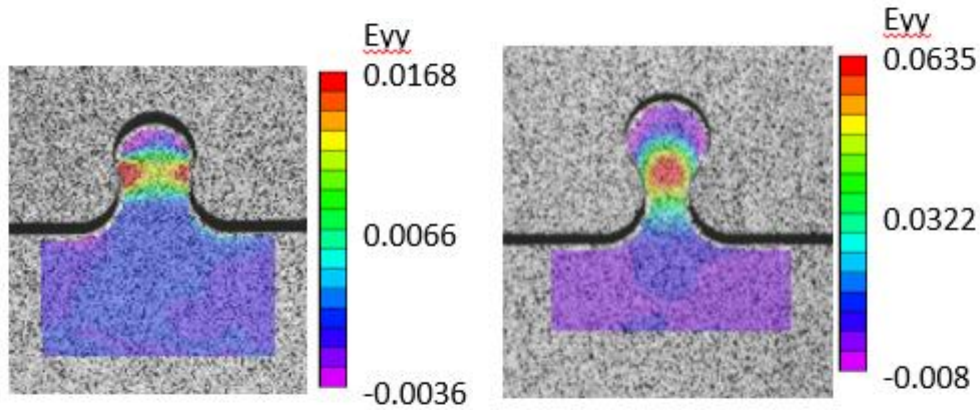


Figure 9: Local deformation in the y-direction in 20-degree suture and 40-degree suture.

The peak load for each case increases with higher angles as expected. This is due to the larger area that will be affected by stress. Only a local deformation is needed to accommodate a relatively large displacement for the lower angle sutures.

In brittle materials there are other failure modes. PMMA specimens were tested in the previous work of this project. The results from those tests showed that low suture angles (10- and 20-degree) will deform slightly to let the tab slip out. In the higher suture angles the tab will break clean off at the neck instead of being stretched as seen in ductile aluminum.

4.3 Fatigue testing

Only one static test was carried out for each geometry before fatigue testing started. The fatigue test loads were calculated from the first static peak loads. Another static test is preformed because of a suspicion that the static loads might be deceivingly low because of the specimen's resistance to fatigue even at high load. An average static peak load is calculated, and the loads used in the fatigue tests are shown as a percentage of the average of the two peak loads.

The results from the 250 kN tests are presented below but the results from these tests should be considered with apprehension.

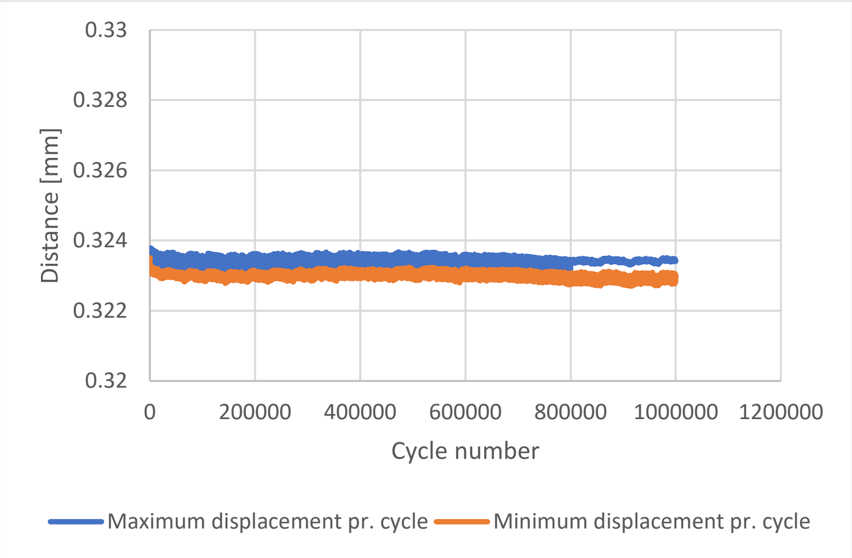
4.3.1 10-degree suture

The peak load from the first static test is 123 N and the second static test is 121 N. Thus, the average peak load is 122 N. As mentioned, the cyclic load for the first fatigue test is between 70% and 7% of the peak load from the first static test. Thus, the cyclic load will vary between 86 N and 9 N. The test finished 1000000 cycles without any displacement. The next tests are done at 80% and 90% of the peak load from the static test. The loads varied between 98 N and 10 N for the 80% test, and 111 N and 11 N for the 90% test. Both tests finished 1000000 cycles without any significant displacement.

	Load % of first static peak	Load % of average static peak	Max / Min load [N]	Cycles without significant displacement	Machine
1st test	70%	70%	86 / 9	1 000 000	10 kN ElectroPuls
2nd test	80%	80%	98 / 10	1 000 000	Instron 250 kN
3rd test	90%	91%	111 / 11	275	Instron 250 kN
4th test	80%	80%	98 / 10	1 000 000	10 kN ElectroPuls
5th test	90%	91%	111 / 11	1 000 000	10 kN ElectroPuls

Table 2: Table of fatigue tests carried out on 10-degree specimens.

It is likely that the 90% test did not fail on the 10 kN machine because it never reached a load high enough to overcome the static friction to move the suture. The 250 kN machine, however, might have given a slightly larger load before the control system was able to stop the large machine.



Plot 3: Plot of maximum and minimum displacement for every cycle during fatigue test of 10-degree suture at 70% load.

There is a possibility that the load in the lower angle fatigue tests is so low that it cannot overcome the static friction between the two parts. This can be a possible exploit as the suture will have long fatigue life when the parts stay together without any sliding, but still be able to sustain large displacement if they occur on rare occasions.

4.3.2 20-degree suture

The peak load from the first static test is 396 N and the second static test is 482 N. Consequently, the average static peak load is 439 N. Similar to the 10-degree suture, the tests are carried out at 70, 80 and 90% of the peak load from the first static test. The last specimen is tested at 90% load to make sure the results stay the same. All the tests run out 1000000 cycles. The 20-degree suture was only tested to maximum 81% of average peak load because of the difference between the peak loads in the two static tests.

	Load % of first static peak	Load % of average static peak	Max / Min load [N]	Cycles without significant displacement	Machine
1st test	70%	63%	277 / 28	1 000 000	10 kN ElectroPuls
2nd test	80%	72%	317 / 32	1 000 000	Instron 250 kN

3rd test	90%	81%	357 / 36	1 000 000	10 kN ElectroPuls
4th test	90%	81%	357 / 36	1 000 000	10 kN ElectroPuls

Table 3: Table of fatigue tests carried out on 20-degree specimens.

4.3.3 30-degree suture

For the 30-degree suture, the first static test peak load is 723 and for the second static test the peak load is 763, making the average peak load 743. Tests are performed at 70, 80 and 90% of peak load from the first static test. The second fatigue test finished 1000000 cycles, but it had a large crack through almost all of the neck of the suture. It was evident that the crack would propagate, and the specimen would fail within a few thousand cycles. Another specimen is tested to confirm that the specimen cannot cycle 1000000 times at 80% load.

	Load % of first static peak	Load % of average static peak	Max / Min load [N]	Cycles without significant displacement	Machine
1st test	70%	68%	506 / 51	1 000 000	10 kN ElectroPuls
2nd test	80%	78%	579 / 58	998 000	10 kN ElectroPuls
3rd test	90%	88%	651 / 65	1 000 000	10 kN ElectroPuls
4th test	80%	78%	579 / 58	763 000	10 kN ElectroPuls

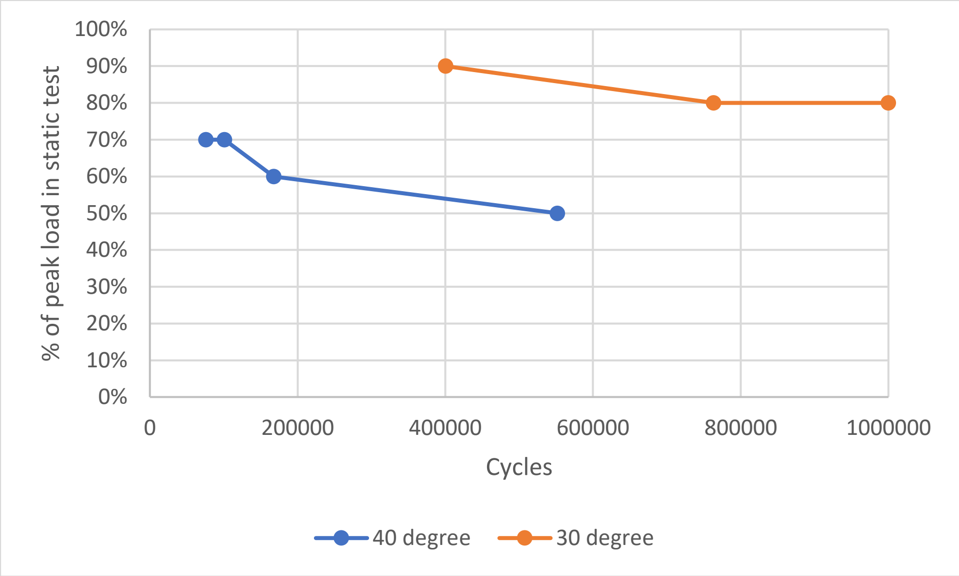
Table 4: Table of fatigue tests carried out on 30-degree specimens.

4.3.4 40-degree suture

The peak load from the first static test is 1037 N and the second static test is 1051 N. The average peak load is 1044 N. The specimen in the first test failed before 100000 cycles so the next test is conducted at 60% load. This specimen also failed so another test is run at 50% load. The last test is done with 70% load again to have two data points on the fatigue life at 70%.

	Load % of first static peak	Load % of average static peak	Max / Min load [N]	Cycles without significant displacement	Machine
1st test	70%	70%	726 / 73	69 199	10 kN ElectroPuls
2nd test	60%	60%	622 / 62	160 941	10 kN ElectroPuls
3rd test	50%	50%	519 / 52	527 243	10 kN ElectroPuls
4th test	70%	70%	726 / 73	94 690	10 kN ElectroPuls

Table 5: Table of fatigue tests carried out on 40-degree specimens.



Plot 4: The failed specimens plotted for cycles against percentage of peak load in static test for that geometry.

The specimens that failed before 1000000 cycles can be seen in Plot 4. There is an indication that the lower angle sutures have longer fatigue life than higher angle sutures. There are few datapoints to support this claim but combined with the fact that none of the 10- and 20-degree sutures failed (apart from one test in the Instron 250 kN machine) it seems likely.

5 Conclusion

The aim of this thesis is to explore the use and limitations for interlocking joints by performing cyclic and static tests on suture geometries and concluding how sutures are suited for use in engineering.

A literature review of nature inspired interlocking joints was conducted. Sutures are found many places in nature and can yield incredible strength to structures. Sutures also give some leeway to otherwise brittle structures by allowing deformation that can be reverted.

FEA was conducted for four different cases in Abaqus using python script. The results were gathered and analyzed. The specimens experience no stress in the top part of the tab. There is some stress around the neck, but the maximum stress in the specimen occurs at the contact point. The stress that appears here are higher than the elastic modulus of aluminum so the part would deform plasticly, and the stress would affect a larger area around the contact point. The tangential stress divided by the normal stress at the contact point is higher for the lower angles. That indicates that the lower angle sutures are more likely to slide.

Experimental testing with static and cyclic loading is carried out. Two specimens from each of the four cases are tested in static loading to find static peak load. Through the data from the tests and DIC analysis it is found that the maximum load the suture structure can withstand before failure increases with higher angle α. The failure happens at the neck of the tab because of high stress through the small cross-section. For brittle materials, the neck will fracture and for ductile materials the neck will deform drastically. For lower angles, the suture is pulled out without failing but can sustain plastic deformation at the contact point.

Results from fatigue tests indicate that lower angle sutures handle cyclic loading better than higher angle sutures. Sutures with high angles are well suited for a structure consisting of a brittle material that needs to be able to have some displacement in a joint and withstand high non-cyclic loads. A low angle suture is better suited for applications with a high number of cycles.

Based on tests conducted in this project, it can seem like lower angle sutures are better suited for engineering applications that are exposed to cyclic loading. However, more research is needed to find an optimal design that combine static loading and fatigue loading for a specific case.

6 Recommendations, improvements, way forward?

The conclusion is reached based on the data from the tests being correct. More tests are needed to have solid evidence for the conclusion reached in this thesis. Specifically, more static tests to confirm the peak loads and more fatigue tests at other loads.

Experiments must be conducted to see how joining dissimilar materials affect the mechanical properties of the connection. Other experiments will need to be run to determine if the strongest material should make up the male or female part of the suture? What are the advantages and disadvantages of joining dissimilar materials with sutures?

Sutures with different geometries and properties including hierarchical structures, clearance, friction, and different material properties are interesting subjects that should be considered for future research. Designing metamaterials consisting of rows of sutures is another possible way forward.

7 Sources

Achrai, B., & Wagner, H. D. (2015). The red-eared slider turtle carapace under fatigue loading: The effect of rib–suture arrangement. *Materials Science and Engineering: C*, *53*, 128–133. https://doi.org/10.1016/j.msec.2015.04.040

Bahls, L., Potapova, M., Fallu, M.-A., & Pienitz, R. (2009). Aulacoseira canadensis and Aulacoseira crassipunctata (Bacillariophyta) in North America. *Nova Hedwigia*, *135*, 167–184.

De Stefano, M., De Stefano, L., & Congestri, R. (2009). Functional morphology of micro- and nanostructures in two distinct diatom frustules. *Superlattices and Microstructures*, *46*(1–2), 64–68. https://doi.org/10.1016/j.spmi.2008.12.007

Findlay, S. J., & Harrison, N. D. (2002). Why aircraft fail. *Materials Today*, *5*(11), 18–25. https://doi.org/10.1016/S1369-7021(02)01138-0

Herring, S. W. (2008). Mechanical influences on suture development and patency. *Frontiers of Oral Biology*, *12*, 41–56. https://doi.org/10.1159/0000115031

Huang, W., Restrepo, D., Jung, J., Su, F. Y., Liu, Z., Ritchie, R. O., McKittrick, J., Zavattieri, P., & Kisailus, D. (2019). Multiscale Toughening Mechanisms in Biological Materials and Bioinspired Designs. *Advanced Materials*, *31*(43), 1901561. https://doi.org/10.1002/adma.201901561

Li, Y., Ortiz, C., & Boyce, M. C. (2012). Bioinspired, mechanical, deterministic fractal model for hierarchical suture joints. *Physical Review E*, *85*(3), 031901. https://doi.org/10.1103/PhysRevE.85.031901

Lin, E., Li, Y., Ortiz, C., & Boyce, M. C. (2014). 3D printed, bio-inspired prototypes and analytical models for structured suture interfaces with geometrically-tuned deformation and failure behavior. *Journal of the Mechanics and Physics of Solids*, *73*, 166–182. https://doi.org/10.1016/j.jmps.2014.08.011

Ma, B. L., Wan, M., Cai, Z. Y., Yuan, W. N., Li, C., Wu, X. D., & Liu, W. (2018). Investigation on the forming limits of 5754-O aluminum alloy sheet with the numerical Marciniak–Kuczynski approach. *International Journal of Mechanical Sciences*, *142–143*, 420–431. https://doi.org/10.1016/j.ijmecsci.2018.05.013

Malik, I. A., Mirkhalaf, M., & Barthelat, F. (2017). Bio-inspired “jigsaw”-like interlocking sutures: Modeling, optimization, 3D printing and testing. *Journal of the Mechanics and Physics of Solids*, *102*, 224–238. https://doi.org/10.1016/j.jmps.2017.03.003

McCormick, N., & Lord, J. (2010). Digital Image Correlation. *Materials Today*, *13*(12), 52–54. https://doi.org/10.1016/S1369-7021(10)70235-2

Mobasher, B. (2016). Textile fiber composites. In *Textile Fibre Composites in Civil Engineering* (pp. 101–150). Elsevier. https://doi.org/10.1016/B978-1-78242-446-8.00006-9

Mohd Abbas, N., Solomon, D. G., & Fuad Bahari, Md. (2007). A review on current research trends in electrical discharge machining (EDM). *International Journal of Machine Tools and Manufacture*, *47*(7–8), 1214–1228. https://doi.org/10.1016/j.ijmachtools.2006.08.026

Mora, C., Tittensor, D. P., Adl, S., Simpson, A. G. B., & Worm, B. (2011). How Many Species Are There on Earth and in the Ocean? *PLoS Biology*, *9*(8), e1001127. https://doi.org/10.1371/journal.pbio.1001127

Naleway, S. E., Porter, M. M., McKittrick, J., & Meyers, M. A. (2015). Structural Design Elements in Biological Materials: Application to Bioinspiration. *Advanced Materials*, *27*(37), 5455–5476. https://doi.org/10.1002/adma.201502403

Puri, G. (2011). *Python scripts for Abaqus: Learn by example* (1st ed). s.n.].

Ritchie, R. O. (2011). The conflicts between strength and toughness. *Nature Materials*, *10*(11), 817–822. https://doi.org/10.1038/nmat3115

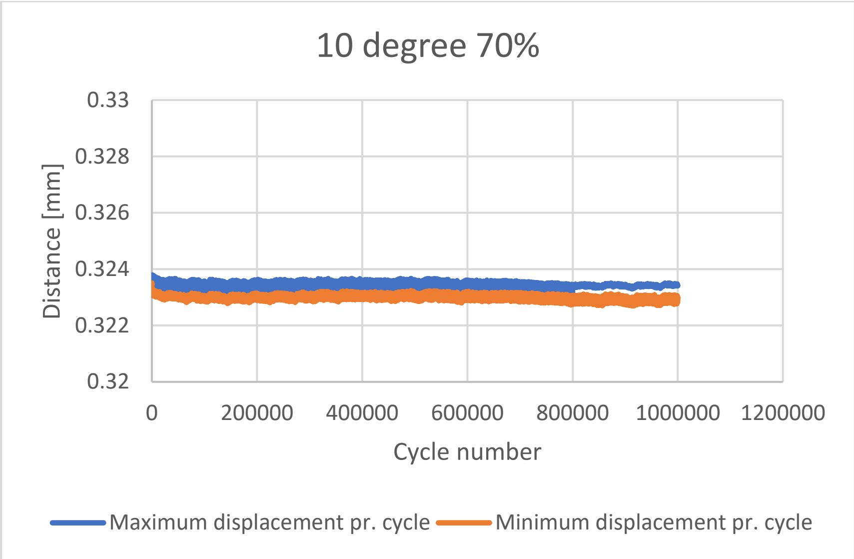
Rivera, J., Hosseini, M. S., Restrepo, D., Murata, S., Vasile, D., Parkinson, D. Y., Barnard, H. S., Arakaki, A., Zavattieri, P., & Kisailus, D. (2020). Toughening mechanisms of the elytra of the diabolical ironclad beetle. *Nature*, *586*(7830), 543–548. https://doi.org/10.1038/s41586-020-2813-8

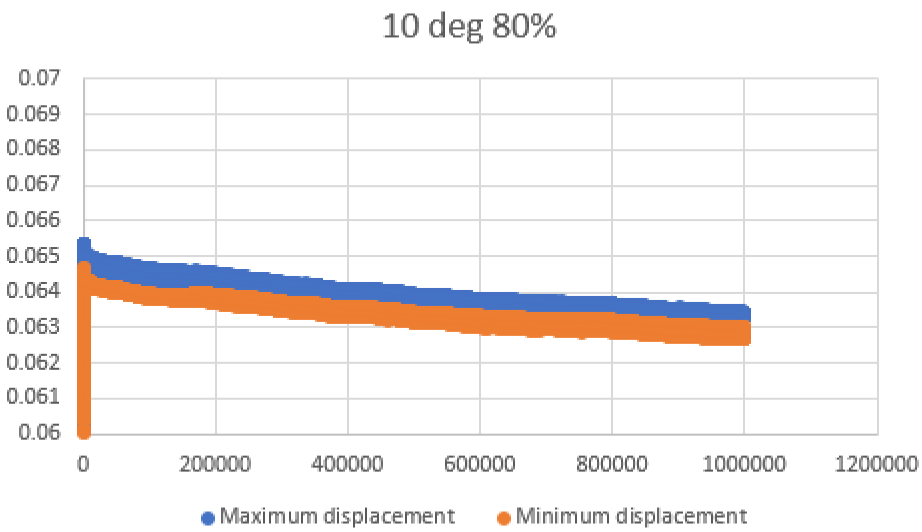
Skrzat, J., Walocha, J., & Zawiliński, J. (2004). A note on the morphology of the metopic suture in the human skull. *Folia Morphologica*, *63*(4), 481–484.

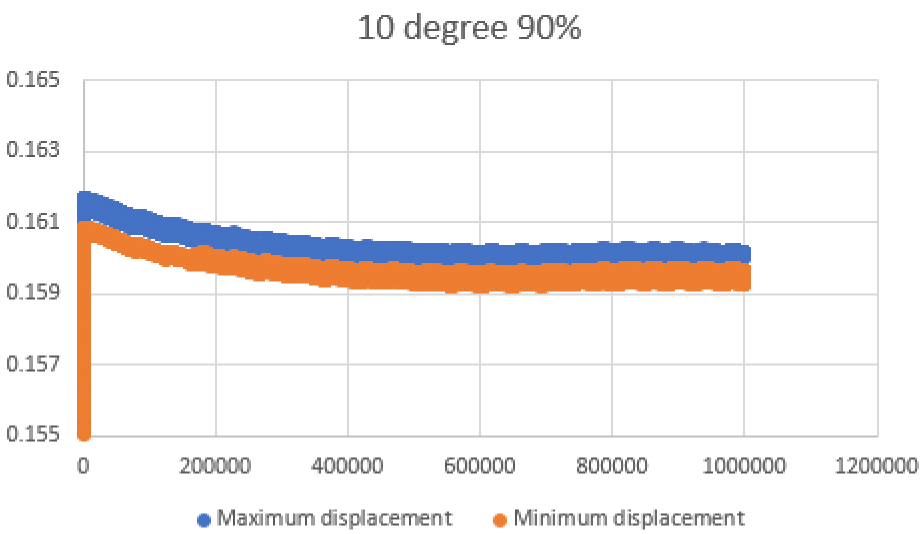
Taylor, D. (2015). Fatigue-resistant components: What can we learn from nature? *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *229*(7), 1186–1193. https://doi.org/10.1177/0954406214530881

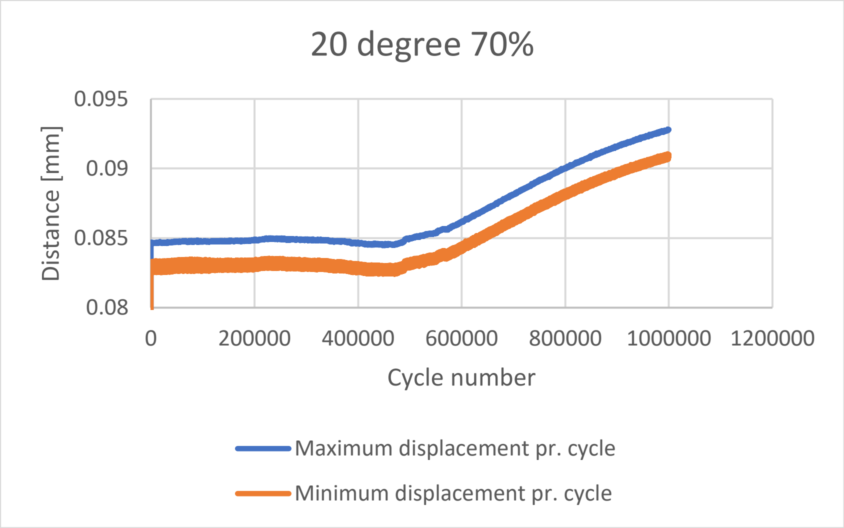
Zhang, Y., Yao, H., Ortiz, C., Xu, J., & Dao, M. (2012). Bio-inspired interfacial strengthening strategy through geometrically interlocking designs. *Journal of the Mechanical Behavior of Biomedical Materials*, *15*, 70–77. https://doi.org/10.1016/j.jmbbm.2012.07.006

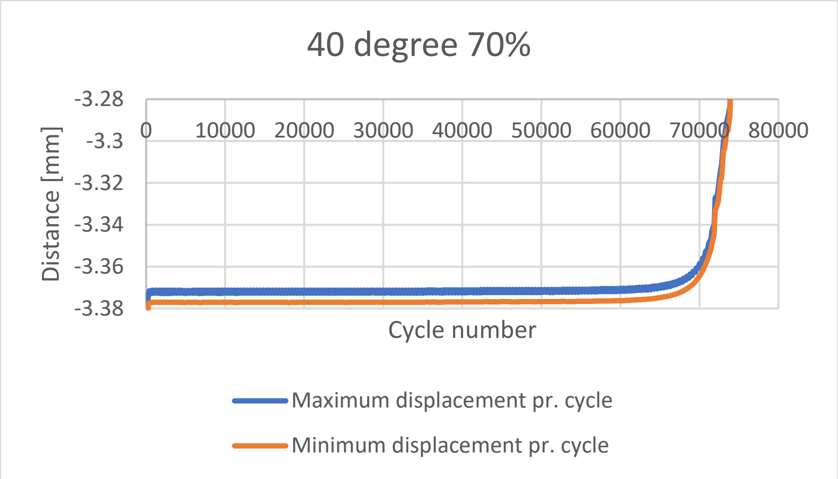
Appendix A – Some plots from fatigue testing








20 degree 70%
Distance [mm]
0.095
0.09
0.085
0.08
0
200000
400000
600000
800000
1000000
1200000
Cycle number
Maximum displacement pr. cycle
Minimum displacement pr. cycle


40 degree 70%
Distance [mm]
-3.28
-3.3
-3.32
-3.34
-3.36
-3.38
0
10000
20000
30000
40000
50000
60000
70000
80000
Cycle number
Maximum displacement pr. cycle
Minimum displacement pr. cycle

Appendix B – Python script for FEA

```
# Script for testing suture with python script in abaqus

# ----------------------------------------------------------------------

from abaqus import *
from abaqusConstants import *
from types import IntType
import regionToolset
import math
import numpy as np

session.viewports['Viewport: 1'].setValues(displayedObject=None)

# ----------------------------------------------------------------------

# Create the model
mdb.models.changeKey(fromName='Model-1', toName='Suture')
sutureModel = mdb.models['Suture']
sutureAssembly = sutureModel.rootAssembly

# ----------------------------------------------------------------------

# Create the part

import sketch
import part

# Parameters for suture
angle = 40
alpha = (angle*(math.pi))/180
radius = 2.0
#load = 1.5
displacement = 0.25

# X coordinates for geometry
leftEdge = 0
rightEdge = 2*radius*math.cos(alpha)
middleX = radius*math.cos(alpha)

# Y coordinates for geometry
bottomEdge = -radius
topEdge = radius+(2*radius+(2*radius*math.sin(alpha)))
topSutureHeight = 2*radius+(2*radius*math.sin(alpha))
bottomSutureHeight = 0
middleY = radius+(radius*sin(alpha))
bottomSutureCurveCenter = radius+(2*radius*math.sin(alpha)) #this is higher
up than topSutureCurveCenter
topSutureCurveCenter = radius

a = (leftEdge, bottomSutureCurveCenter)         #center of first arc
b = (rightEdge, topSutureCurveCenter)           #center of second arc
c = (leftEdge, topSutureHeight)                 #top of suture connection
```

```
d = (rightEdge, bottomSutureHeight)                #bottom of suture
connection
e = (middleX, middleY)                             #middle of suture
connection
f = (rightEdge, bottomEdge)                        #bottom rigth corner
g = (leftEdge, bottomEdge)                         #bottom left corner
h = (leftEdge, topEdge)                            #top left corner
i = (rightEdge, topEdge)                                 #top right corner
j = (leftEdge+radius, bottomSutureCurveCenter)           #point along top
suture line
k = (rightEdge-radius, topSutureCurveCenter)             #point along bottom
suture line

# Sketching the curve of the bottom suture
sutureSketchBottom = sutureModel.ConstrainedSketch(name='Bottom Curve',
sheetSize=5)
sutureSketchBottom.ArcByCenterEnds(a, e, c)
sutureSketchBottom.ArcByCenterEnds(b, e, d)
sutureSketchBottom.Line(d, f)
sutureSketchBottom.Line(f, g)
sutureSketchBottom.Line(g, c)

# Sketching the curve of the top suture
sutureSketchTop = sutureModel.ConstrainedSketch(name='Top Curve',
sheetSize=5)
sutureSketchTop.ArcByCenterEnds(a, e, c)
sutureSketchTop.ArcByCenterEnds(b, e, d)
sutureSketchTop.Line(c, h)
sutureSketchTop.Line(h, i)
sutureSketchTop.Line(i, d)

# Creating the bottom suture
sutureBottomPart=sutureModel.Part(name='Suture Bottom',
dimensionality=TWO_D_PLANAR, type=DEFORMABLE_BODY)
sutureBottomPart.BaseShell(sketch=sutureSketchBottom)

# Creating the top suture
sutureTopPart=sutureModel.Part(name='Suture Top',
dimensionality=TWO_D_PLANAR, type=DEFORMABLE_BODY)
sutureTopPart.BaseShell(sketch=sutureSketchTop)

# ----------------------------------------------------------------------

# Create material
sutureMaterial = sutureModel.Material(name='Aluminum')
sutureMaterial.Density(table=((2.7E-9, ), ))
sutureMaterial.Elastic(table=((65E3, 0.33), ))
sutureMaterial.Plastic(table=((122.0, 0.0), (
    122.9602856, 0.002132108), (128.1869567, 0.004850309), (133.680921,
    0.007650879), (139.6514567, 0.01028671), (144.4386993, 0.012892588), (
    149.4234869, 0.015657215), (153.2734543, 0.018165131), (158.3525364,
    0.020634405), (162.4168354, 0.023296279), (166.8616791, 0.026200539), (
    171.0134806, 0.028455735), (173.986597, 0.030898519), (177.7259595,
```

```
    0.03386504), (181.2737798, 0.035831617), (184.6957897, 0.038515987), (
    187.1591942, 0.040963545), (191.341173, 0.043893553), (194.6594536,
    0.045604309), (197.0881245, 0.048109853), (199.8985905, 0.051251914), (
    202.6942288, 0.053720893), (206.4397878, 0.055910861), (207.4276772,
    0.058512009), (211.3610867, 0.061345636), (214.1537478, 0.063909393), (
    215.9672194, 0.066473151), (218.8963683, 0.068163201), (219.9836773,
    0.070771424), (222.802691, 0.07218755), (223.7751717, 0.074683352), (
    226.3714137, 0.077716617), (228.3133039, 0.080107397), (230.3634323,
    0.082878654), (232.9386539, 0.085462802), (235.1134604, 0.088604863), (
    236.4540465, 0.091020796), (238.2124844, 0.093577934), (241.3115242,
    0.096657857), (243.1101089, 0.099034192), (246.5286723, 0.104753004), (
    248.836198, 0.10732338), (250.2227789, 0.110310998), (251.532982,
    0.112764324), (253.4277356, 0.115607227), (255.3593565, 0.118344007), (
    256.2134838, 0.120881778), (256.9618292, 0.12394147), (258.3012702,
    0.126400549), (260.8860025, 0.128934398), (261.8887844, 0.132035482), (
    263.9996752, 0.134685569), (265.103743, 0.137241972), (265.7430859,
    0.139950367), (267.445409, 0.142891652), (268.387202, 0.145077478), (
    268.7483032, 0.14713026), (269.0482769, 0.149467299), (270.127366,
    0.15260613), (272.3406727, 0.155092666), (273.1925671, 0.157906964), (
    275.5704091, 0.160759702), (275.8157405, 0.163234912), (276.2261334,
    0.166253762), (277.514634, 0.168989613), (278.5626341, 0.171637211), (
    279.2963374, 0.174413512), (279.9049693, 0.17711009), (281.6698728,
    0.179649137), (282.2475649, 0.182495769), (283.7715846, 0.185411057), (
    284.6772156, 0.187903268), (284.8328099, 0.190779344), (285.4569003,
    0.193394583), (286.4313822, 0.196112783), (287.4802637, 0.198830984), (
    288.7514984, 0.201549184), (289.4199413, 0.20395171), (289.9943469,
    0.205997149)))

# ---------------------------------------------------------------------

# Create sections and assign sutures to it

import section

# Create a section to assign to each suture in case they will have
different materials or thickness
sutureBottomSection = sutureModel.HomogeneousSolidSection(name='Bottom
Suture Section', material='Aluminum', thickness=2.0)
sutureTopSection = sutureModel.HomogeneousSolidSection(name='Top Suture
Section', material='Aluminum', thickness=2.0)

# Assign the sutures to the sections
sutureBottomRegion = (sutureBottomPart.faces,)
sutureTopRegion = (sutureTopPart.faces,)
sutureBottomPart.SectionAssignment(region=sutureBottomRegion,
sectionName='Bottom Suture Section')
sutureTopPart.SectionAssignment(region=sutureTopRegion, sectionName='Top
Suture Section')

# ---------------------------------------------------------------------

# Create the assembly
```

```
import assembly

# Create part instances
sutureAssembly = sutureModel.rootAssembly
sutureBottomInstance = sutureAssembly.Instance(name='Suture Bottom
Instance', part=sutureBottomPart, dependent=ON)
sutureTopInstance = sutureAssembly.Instance(name='Suture Top Instance',
part=sutureTopPart, dependent=ON)

# -----------------------------------------------------------------------

# Create the step

import step

# Create a static general step
sutureModel.StaticStep(name='Apply Load', previous='Initial',
description='Load is applied during this step', minInc=1e-5)
sutureModel.steps['Apply Load'].setValues(initialInc=0.01, minInc=1e-50,
maxInc=0.01)

# -----------------------------------------------------------------------
# Create the field output request

# Change the name of field output request 'F-Output-1' to 'Selected Field
Outputs'
sutureModel.fieldOutputRequests.changeKey(fromName='F-Output-1',
toName='Selected Field Outputs')

# Setting required outputs
sutureModel.fieldOutputRequests['Selected Field
Outputs'].setValues(variables=('S', 'E', 'U', 'RF', 'CF'))

# -----------------------------------------------------------------------
# Create the history output request

# Change the name of the history output request 'H-Output-1 to 'Selected
History Outputs'
sutureModel.historyOutputRequests.changeKey(fromName='H-Output-1',
toName='Selected History Outputs')

# Setting output request
sutureModel.historyOutputRequests['Selected History
Outputs'].setValues(variables=('IRF1', 'IRF2', 'IRF3', 'IRM1', 'IRM2',
'IRM3', 'ALLAE', 'ALLCD', 'ALLDMD', 'ALLEE', 'ALLFD', 'ALLIE', 'ALLJD',
'ALLKE', 'ALLKL', 'ALLPD', 'ALLQB', 'ALLSE', 'ALLSD', 'ALLVD', 'ALLWK',
'ETOTAL'))

# -----------------------------------------------------------------------
# Create interaction

import interaction
```

```
# Selecting edges
top_suture_edge_top_part = sutureTopInstance.edges.findAt(((rightEdge-
radius, topSutureCurveCenter,0),))
bottom_suture_edge_top_part =
sutureTopInstance.edges.findAt(((leftEdge+radius,
bottomSutureCurveCenter,0),))
top_suture_edge_bottom_part =
sutureBottomInstance.edges.findAt(((rightEdge-radius,
topSutureCurveCenter,0),))
bottom_suture_edge_bottom_part =
sutureBottomInstance.edges.findAt(((leftEdge+radius,
bottomSutureCurveCenter,0),))

# Selecting the top edge
top_edge = sutureTopInstance.edges.findAt(((radius,topEdge,0),))
top_edge_region=regionToolset.Region(edges=top_edge)        # change edges
to side1Edges when using load

# Making surfaces from edges
surface1 = sutureAssembly.Surface(side1Edges=top_suture_edge_top_part,
name='Top Suture Line Top Part')
surface2 = sutureAssembly.Surface(side1Edges=bottom_suture_edge_top_part,
name='Bottom Suture Line Top Part')
surface3 = sutureAssembly.Surface(side1Edges=top_suture_edge_bottom_part,
name='Top Suture Line Bottom Part')
surface4 =
sutureAssembly.Surface(side1Edges=bottom_suture_edge_bottom_part,
name='Bottom Suture Line Bottom Part')

# Merging surfaces
suture_edge_top_part = sutureAssembly.SurfaceByBoolean(name='Suture Line
Top Part', surfaces=(surface1, surface2))
suture_edge_bottom_part = sutureAssembly.SurfaceByBoolean(name='Suture Line
Bottom Part', surfaces=(surface3, surface4))

# Create interaction property
frictionless_interaction = sutureModel.ContactProperty('Frictionless')
frictionless_interaction.TangentialBehavior(formulation=PENALTY,
directionality=ISOTROPIC, slipRateDependency=OFF,
    pressureDependency=OFF, temperatureDependency=OFF, dependencies=0,
table=((
    0.1, ), ), shearStressLimit=None, maximumElasticSlip=FRACTION,
    fraction=0.005, elasticSlipStiffness=None)
#frictionless_interaction.setValues(initialClearance=OMIT,
adjustMethod=OVERCLOSED, sliding=FINITE, enforcement=SURFACE_TO_SURFACE,
thickness=ON, contactTracking=TWO_CONFIG, tied=OFF, bondingSet=None)

frictionless_master_surface_region = sutureAssembly.surfaces['Suture Line
Top Part']
frictionless_slave_surface_region = sutureAssembly.surfaces['Suture Line
Bottom Part']

# Create interaction
```

```
sutureModel.SurfaceToSurfaceContactStd(name='Frictionless Contact',
createStepName='Apply Load', master=frictionless_master_surface_region,
slave=frictionless_slave_surface_region, sliding=FINITE,
interactionProperty='Frictionless')

# Create reference point
sutureEdges = sutureAssembly.instances['Suture Top Instance'].edges
ReferencePoint1 =
sutureAssembly.ReferencePoint(point=sutureAssembly.instances['Suture Top
Instance'].InterestingPoint(edge=sutureEdges[2], rule=MIDDLE))

# Couple reference point with top edge
region1 =
sutureAssembly.Set(referencePoints=(sutureAssembly.referencePoints[11], ),
name='m_Set-7')
region2 = sutureAssembly.Surface(side1Edges=top_edge, name='s_Surf-7')
sutureModel.Coupling(name='Reference Point Coupling', controlPoint=region1,
    surface=region2, influenceRadius=WHOLE_SURFACE, couplingType=KINEMATIC,
    localCsys=None, u1=ON, u2=ON, ur3=ON)

# Request reaction forces
regionDef=sutureAssembly.sets['m_Set-7']
sutureModel.historyOutputRequests['Selected History Outputs'].setValues(
    variables=('RF1', 'RF2', 'RF3', 'RM1', 'RM2', 'RM3', 'RT', 'IRF1',
'IRF2',
    'IRF3', 'IRM1', 'IRM2', 'IRM3', 'ALLAE', 'ALLCD', 'ALLDMD', 'ALLEE',
    'ALLFD', 'ALLIE', 'ALLJD', 'ALLKE', 'ALLKL', 'ALLPD', 'ALLQB', 'ALLSE',
    'ALLSD', 'ALLVD', 'ALLWK', 'ETOTAL'), region=regionDef,
    sectionPoints=DEFAULT, rebar=EXCLUDE)

# ---------------------------------------------------------------------
# Apply pressure load to the top surface

# Apply the pressure load on this region inn the 'Apply Load' step
#sutureModel.Pressure(name='Uniform Applied Pressure',
createStepName='Apply Load', region=top_edge_region,
distributionType=UNIFORM, magnitude=-load, amplitude=UNSET)
sutureModel.DisplacementBC(name='Top Edge Movement', createStepName='Apply
Load', region=region1, u1=0, u2=displacement, ur3=0)

# ---------------------------------------------------------------------
# Apply boundry conditions

# Select all edges with boundary conditions
left_edge_top =
sutureTopInstance.edges.findAt(((0,(radius/2)+(2*radius+(2*radius*math.sin(
alpha))),0),))
left_edge_bottom = sutureBottomInstance.edges.findAt(((0, 0, 0),))
right_edge_top = sutureTopInstance.edges.findAt(((2*radius*math.cos(alpha),
radius,0),))
right_edge_bottom =
sutureBottomInstance.edges.findAt(((2*radius*math.cos(alpha),-
(radius/2),0),))
```

```
bottom_edge = sutureBottomInstance.edges.findAt((((radius/2),-radius,0),))

# Make regions from edges
left_edge_top_region = regionToolset.Region(edges=left_edge_top)
left_edge_bottom_region = regionToolset.Region(edges=left_edge_bottom)
right_edge_top_region = regionToolset.Region(edges=right_edge_top)
right_edge_bottom_region = regionToolset.Region(edges=right_edge_bottom)
bottom_edge_region = regionToolset.Region(edges=bottom_edge)

# Set boundary conditions on regions
sutureModel.XsymmBC(name='Left Edge Top Symmetry',
createStepName='Initial', region=left_edge_top_region)
sutureModel.XsymmBC(name='Left Edge Bottom Symmetry',
createStepName='Initial', region=left_edge_bottom_region)
sutureModel.XsymmBC(name='Right Edge Top Symmetry',
createStepName='Initial', region=right_edge_top_region)
sutureModel.XsymmBC(name='Right Edge Bottom Symmetry',
createStepName='Initial', region=right_edge_bottom_region)
sutureModel.EncastreBC(name='Bottom Edge Encaster',
createStepName='Initial', region=bottom_edge_region)

# ---------------------------------------------------------------------
# Create mesh

import mesh

# Create global mesh on both parts
sutureTopPart.seedPart(deviationFactor=0.1, minSizeFactor=0.1, size=0.1)
sutureBottomPart.seedPart(deviationFactor=0.1, minSizeFactor=0.1, size=0.1)

# Create quadratic elements            Commented out because it made some
weird artifacts
#elemType1 = mesh.ElemType(elemCode=CPS8R, elemLibrary=STANDARD)
#elemType2 = mesh.ElemType(elemCode=CPS6M, elemLibrary=STANDARD)
#face1 = sutureBottomPart.faces.findAt(((0, 0, 0),))
#face2 = sutureBottomPart.faces.findAt(((leftEdge, bottomSutureCurveCenter,
0),))
#face3 = sutureTopPart.faces.findAt(((rightEdge, topSutureCurveCenter,
0),))
#pickedRegions1 = (face1, face2)
#sutureBottomPart.setElementType(regions=pickedRegions1,
elemTypes=(elemType1, elemType2))
#pickedRegions2 = (face3, )
#sutureTopPart.setElementType(regions=pickedRegions2, elemTypes=(elemType1,
elemType2))

# Select suture line edges (Already done in interaction step)
#top_suture_edge_top_part = sutureTopInstance.edges.findAt(((rightEdge-
radius, topSutureCurveCenter,0),))
#bottom_suture_edge_top_part =
sutureTopInstance.edges.findAt(((leftEdge+radius,
bottomSutureCurveCenter,0),))
```

```
#top_suture_edge_bottom_part =
sutureBottomInstance.edges.findAt(((rightEdge-radius,
topSutureCurveCenter,0),))
#bottom_suture_edge_bottom_part =
sutureBottomInstance.edges.findAt(((leftEdge+radius,
bottomSutureCurveCenter,0),))

# Create seed edge bias along suture line
sutureTopPart.seedEdgeByBias(biasMethod=SINGLE, constraint=FINER,
end1Edges=top_suture_edge_top_part, number=200, ratio=100)
sutureTopPart.seedEdgeByBias(biasMethod=SINGLE, constraint=FINER,
end1Edges=bottom_suture_edge_top_part, number=200, ratio=100)
sutureBottomPart.seedEdgeByBias(biasMethod=SINGLE, constraint=FINER,
end1Edges=top_suture_edge_bottom_part, number=200, ratio=100)
sutureBottomPart.seedEdgeByBias(biasMethod=SINGLE, constraint=FINER,
end1Edges=bottom_suture_edge_bottom_part, number=200, ratio=100)

# Generate mesh
sutureTopPart.generateMesh()
sutureBottomPart.generateMesh()

# ----------------------------------------------------------------------
# Create and run the job
import job

# Create the job
mdb.Job(name='SuturePulloutJob', model='Suture', type=ANALYSIS,
explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE,
description='Simulate pullout of suture',
parallelizationMethodExplicit=DOMAIN, multiprocessingMode=DEFAULT,
numDomains=1, userSubroutine='', numCpus=1, memory=50,
memoryUnits=PERCENTAGE, scratch='', echoPrint=OFF, modelPrint=OFF,
contactPrint=OFF, historyPrint=OFF)

# Run the job
mdb.jobs['SuturePulloutJob'].submit(consistencyChecking=OFF)

# Do not return control till job is finished running
mdb.jobs['SuturePulloutJob'].waitForCompletion()

# ----------------------------------------------------------------------
# Post processing

import visualization
import displayGroupOdbToolset as dgo

# Open ODB and set viewport display
suture_viewport = session.Viewport(name='Suture Results Viewport')
suture_Odb_Path = 'SuturePulloutJob.odb'
an_odb_object = session.openOdb(name=suture_Odb_Path)
suture_viewport.setValues(displayedObject=an_odb_object)
suture_viewport.odbDisplay.display.setValues(plotState=(CONTOURS_ON_DEF, ))
```

```
suture_viewport.odbDisplay.commonOptions.setValues(deformationScaling=UNIFO
RM, uniformScaleFactor=1, visibleEdges=FEATURE)
suture_viewport.odbDisplay.setPrimaryVariable(variableLabel='S',
outputPosition=INTEGRATION_POINT, refinement=(COMPONENT, 'S22'), )
suture_viewport.odbDisplay.basicOptions.setValues(mirrorAboutYzPlane=True,
mirrorDisplayBodies=True)
session.viewports['Suture Results Viewport'].makeCurrent()
session.graphicsOptions.setValues(backgroundStyle=SOLID,
backgroundColor='#FFFFFF')
leaf = dgo.LeafFromPartInstance(partInstanceName=('Suture Top Instance', ))
session.viewports['Suture Results
Viewport'].odbDisplay.displayGroup.remove(leaf=leaf)

# ***********************************************************************

# Create a path along suture neck and create data
numPoints = 30
neckLinePoints = []
i = 0
while i <= numPoints:
    neckLinePoints.append(((i*(rightEdge-radius)/numPoints),
(topSutureCurveCenter), (0)))
    i += 1
suture_line_path = session.Path(name='Neck Line Path', type=POINT_LIST,
expression=neckLinePoints)
pth1 = session.paths['Neck Line Path']
sutureNeckData = session.XYDataFromPath(name='Suture Neck Data', path=pth1,
includeIntersections=True, projectOntoMesh=False, pathStyle=PATH_POINTS,
numIntervals=100, projectionTolerance=0, shape=DEFORMED,
labelType=TRUE_DISTANCE)
sutureNeckPositions = []
for neckPositions in sutureNeckData:
    sutureNeckPositions.append(neckPositions[0])
sutureNeckPositions = str(sutureNeckPositions).replace('[','')
sutureNeckPositions = sutureNeckPositions.replace(']','')
sutureNeckStress = []
for neckStress in sutureNeckData:
    sutureNeckStress.append(neckStress[1])
sutureNeckStress = str(sutureNeckStress).replace('[','')
sutureNeckStress = sutureNeckStress.replace(']','')

# ***********************************************************************

# Make cylindrical coordinate system for bottom suture line
bottomCylinderCoordinates =
session.odbs[suture_Odb_Path].rootAssembly.DatumCsysByThreePoints(name='Bot
tom Suture Line Coordinates', coordSysType=CYLINDRICAL, origin=(rightEdge,
topSutureCurveCenter, 0), point1=(rightEdge, topSutureHeight, 0),
point2=(rightEdge-radius, topSutureCurveCenter, 0))
suture_viewport.odbDisplay.basicOptions.setValues(transformationType=USER_S
PECIFIED, datumCsys=bottomCylinderCoordinates)

# Create a path along bottom suture line
```

```
suture_line_path = session.Path(name='Bottom Suture Line Path',
type=CIRCUMFERENTIAL, expression=((rightEdge, topSutureCurveCenter,
0.0),(rightEdge, topSutureCurveCenter, 1.0),(rightEdge, topSutureHeight,
0.0)), circleDefinition=ORIGIN_AXIS, numSegments=200, startAngle=90-angle,
endAngle=180, radius=radius)
pth2 = session.paths['Bottom Suture Line Path']

# Set variable to S22 and create XY data
suture_viewport.odbDisplay.setPrimaryVariable(variableLabel='S',
outputPosition=INTEGRATION_POINT, refinement=(COMPONENT, 'S22'), )
bottomSutureLineS22Data = session.XYDataFromPath(name='Bottom Suture Line
Data', path=pth2, includeIntersections=True, projectOntoMesh=False,
pathStyle=PATH_POINTS, numIntervals=100, projectionTolerance=0,
shape=UNDEFORMED, labelType=NORM_DISTANCE)

# Save S22 data from bottom path
bottomSutureLineS22Positions = []
for bottomS22Positions in bottomSutureLineS22Data:
    bottomSutureLineS22Positions.append(bottomS22Positions[0]/2+0.5)
bottomSutureLineS22Positions =
str(bottomSutureLineS22Positions).replace('[','')
bottomSutureLineS22Positions = bottomSutureLineS22Positions.replace(']','')
bottomSutureLineS22Stress = []
for bottomStress in bottomSutureLineS22Data:
    bottomSutureLineS22Stress.append(bottomStress[1])
bottomSutureLineS22Stress = str(bottomSutureLineS22Stress).replace('[','')
bottomSutureLineS22Stress = bottomSutureLineS22Stress.replace(']','')

# Set variable to S11 and create XY data
suture_viewport.odbDisplay.setPrimaryVariable(variableLabel='S',
outputPosition=INTEGRATION_POINT, refinement=(COMPONENT, 'S11'), )
bottomSutureLineS11Data = session.XYDataFromPath(name='Bottom Suture Line
Data', path=pth2, includeIntersections=True, projectOntoMesh=False,
pathStyle=PATH_POINTS, numIntervals=100, projectionTolerance=0,
shape=UNDEFORMED, labelType=NORM_DISTANCE)

# Save S11 data from bottom path
bottomSutureLineS11Positions = []
for bottomS11Positions in bottomSutureLineS11Data:
    bottomSutureLineS11Positions.append(bottomS11Positions[0]/2+0.5)
bottomSutureLineS11Positions =
str(bottomSutureLineS11Positions).replace('[','')
bottomSutureLineS11Positions = bottomSutureLineS11Positions.replace(']','')
bottomSutureLineS11Stress = []
for bottomStress in bottomSutureLineS11Data:
    bottomSutureLineS11Stress.append(bottomStress[1])
bottomSutureLineS11Stress = str(bottomSutureLineS11Stress).replace('[','')
bottomSutureLineS11Stress = bottomSutureLineS11Stress.replace(']','')


# **********************************************************************

# Make cylindrical coordinate system for top suture line
```

```
topCylinderCoordinates =
session.odbs[suture_Odb_Path].rootAssembly.DatumCsysByThreePoints(name='Top
Suture Line Coordinates', coordSysType=CYLINDRICAL, origin=(leftEdge,
bottomSutureCurveCenter, 0), point1=(leftEdge, bottomSutureHeight, 0),
point2=(middleX, middleY, 0))
suture_viewport.odbDisplay.basicOptions.setValues(transformationType=USER_S
PECIFIED, datumCsys=topCylinderCoordinates)

# Create a path along top suture line
suture_line_path = session.Path(name='Top Suture Line Path',
type=CIRCUMFERENTIAL, expression=((leftEdge, bottomSutureCurveCenter,
0.0),(leftEdge, bottomSutureCurveCenter, -1.0),(leftEdge,
bottomSutureHeight, 0.0)), circleDefinition=ORIGIN_AXIS, numSegments=200,
startAngle=180, endAngle=270+angle, radius=radius)
pth3 = session.paths['Top Suture Line Path']

# Set variable to S22 and create XY data
suture_viewport.odbDisplay.setPrimaryVariable(variableLabel='S',
outputPosition=INTEGRATION_POINT, refinement=(COMPONENT, 'S22'), )
topSutureLineS22Data = session.XYDataFromPath(name='Top Suture Line Data',
path=pth3, includeIntersections=True, projectOntoMesh=False,
pathStyle=PATH_POINTS, numIntervals=100, projectionTolerance=0,
shape=UNDEFORMED, labelType=NORM_DISTANCE)

# Save S22 data from top path
topSutureLineS22Positions = []
for topPositions in topSutureLineS22Data:
    topSutureLineS22Positions.append(topPositions[0]/2)
topSutureLineS22Positions = str(topSutureLineS22Positions).replace('[','')
topSutureLineS22Positions = topSutureLineS22Positions.replace(']','')
topSutureLineS22Stress = []
for topStress in topSutureLineS22Data:
    topSutureLineS22Stress.append(topStress[1])
topSutureLineS22Stress = str(topSutureLineS22Stress).replace('[','')
topSutureLineS22Stress = topSutureLineS22Stress.replace(']','')

# Set variable to S11 and create XY data
suture_viewport.odbDisplay.setPrimaryVariable(variableLabel='S',
outputPosition=INTEGRATION_POINT, refinement=(COMPONENT, 'S11'), )
topSutureLineS11Data = session.XYDataFromPath(name='Top Suture Line Data',
path=pth3, includeIntersections=True, projectOntoMesh=False,
pathStyle=PATH_POINTS, numIntervals=100, projectionTolerance=0,
shape=UNDEFORMED, labelType=NORM_DISTANCE)

# Save S11 data from bottom path
topSutureLineS11Positions = []
for topS11Positions in topSutureLineS11Data:
    topSutureLineS11Positions.append(topS11Positions[0]/2)
topSutureLineS11Positions = str(topSutureLineS11Positions).replace('[','')
topSutureLineS11Positions = topSutureLineS11Positions.replace(']','')
topSutureLineS11Stress = []
for topStress in topSutureLineS11Data:
    topSutureLineS11Stress.append(topStress[1])
```

```
topSutureLineS11Stress = str(topSutureLineS11Stress).replace('[','')
topSutureLineS11Stress = topSutureLineS11Stress.replace(']','')


# Write data into file
dataFile = open('SutureData.csv','w')
dataFile.write(str(angle))
dataFile.write(' degree suture. Normalized')
dataFile.write('\n')
dataFile.write('S22 along suture line')
dataFile.write('\n')
dataFile.write(topSutureLineS22Positions)
dataFile.write(',')
dataFile.write(bottomSutureLineS22Positions)
dataFile.write('\n')
dataFile.write(topSutureLineS22Stress)
dataFile.write(',')
dataFile.write(bottomSutureLineS22Stress)
dataFile.write('\n')
dataFile.write('S11 along suture line')
dataFile.write('\n')
dataFile.write(topSutureLineS11Positions)
dataFile.write(',')
dataFile.write(bottomSutureLineS11Positions)
dataFile.write('\n')
dataFile.write(topSutureLineS11Stress)
dataFile.write(',')
dataFile.write(bottomSutureLineS11Stress)
dataFile.write('\n')
dataFile.write('S22 along neck of suture')
dataFile.write('\n')
dataFile.write(sutureNeckPositions)
dataFile.write('\n')
dataFile.write(sutureNeckStress)
dataFile.close()
```

Norwegian University of
Science and Technology