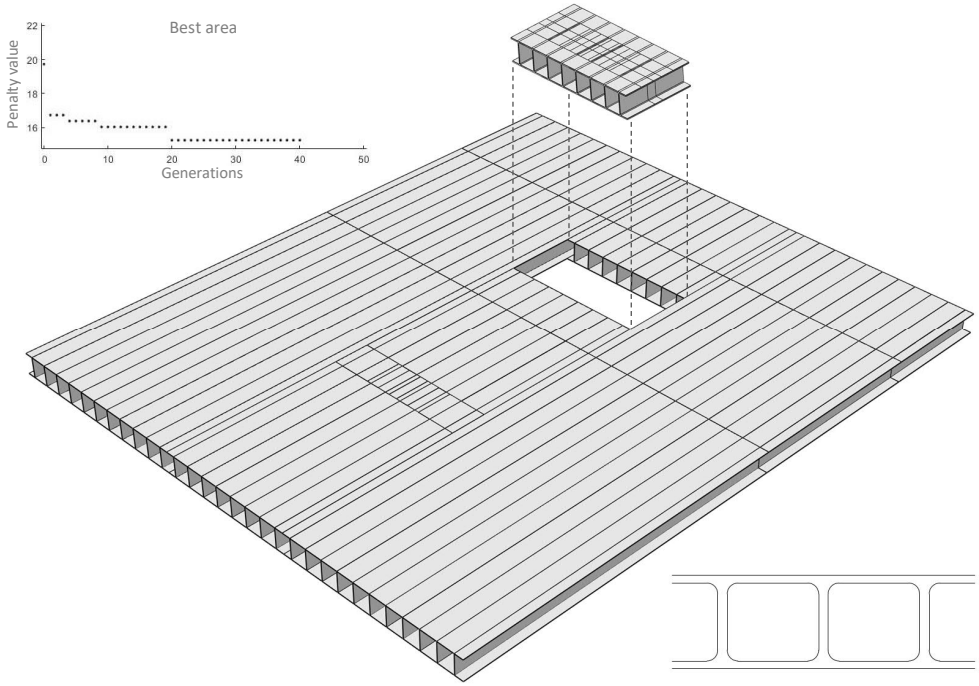
CHALMERS
UNIVERSITY OF TECHNOLOGY



Extruded aluminum decks for pedestrian bridges

Design optimization using genetic algorithm

Master's thesis in Master Program Structural engineering and building technology

EMELIE FALKENBERG
MARKUS FREDRIKSSON

Department of Architecture and Civil Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Master's thesis ACEX30
Gothenburg, Sweden 2018

Master's thesis ACEX30

Extruded aluminum decks for pedestrian bridges

Design optimization using genetic algorithm

EMELIE FALKENBERG
MARKUS FREDRIKSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Architecture and Civil Engineering
Division of Structural Engineering
Lightweight Structures
Chalmers University of Technology
Gothenburg, Sweden 2023

Extruded aluminum decks for pedestrian bridges
Design optimization using genetic algorithm

EMELIE FALKENBERG
MARKUS FREDRIKSSON

Supervisor: PhD Peter Nilsson Strand, WSP Sverige AB
Examiner: Professor Mohammad Al-Emrani, Department of Architecture and Civil Engineering

Department of Architecture and Civil Engineering
Division of Structural Engineering
Lightweight Structures
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover:
Left: Convergence plot of optimization resulting from this theses, see Section 5.3.
Middle: Shell and solid part, showing the sub-modelling approach used in the of the FE-model.
Right: Core configuration resulting from the optimization in this thesis, see Section 5.3.

Department of Architecture and Civil Engineering
Gothenburg, Sweden 2023

Extruded aluminum decks for pedestrian bridges
Design optimization using genetic algorithm
EMELIE FALKENBERG
MARKUS FREDRIKSSON
Department of Architecture and Civil Engineering
Chalmers University of Technology

Abstract

Material efficiency in bridge structures is an important research topic to reduce the climate footprint and initial costs. One way of increasing material efficiency is to use appropriate materials where their properties benefit the structure. Aluminium offers a high stiffness-to-weight ratio, high durability, and is to a large degree recyclable. These properties make aluminium an interesting material for bridge deck applications.

The initial cost of the bridge deck has high priority amongst bridge authorities in Europe. To minimize the initial cost, one target is to minimize material consumption. This thesis aims to develop an optimization procedure with the objective to minimize the material consumption of extruded aluminium profiles for pedestrian bridge deck applications. In the design of the deck, requirements stated in Eurocode are followed. The optimization is made using a genetic algorithm function from the global optimization toolbox in the software MATLAB. Cross-sectional geometries generated by the optimization procedure was evaluated separately by the implementation of a FE-module. The FE-module is controlled by parameterized Python scripting to create a FE-model and execute an analysis in the software ABAQUS CAE for each iteration.

The optimization provided a cross-sectional geometry for the bridge deck. These results were used for a cost comparison between the optimized cross-section and more conventional alternatives in C-Mn and duplex steel. The comparison showed that the weight per square meter of the resulting optimized profile was significantly lower compared to the deck alternatives in steel. It also showed that saving from 8% up to 27% can be made on the initial investment cost if choosing aluminium instead of a conventional deck alternative in stainless steel. Thus, aluminium is shown to be a potential alternative to steel.

Keywords: Optimization, extruded aluminium profiles, pedestrian bridge, bridge deck, genetic algorithm, aluminium bridge, finite element analysis.

Extruded aluminum decks for pedestrian bridges
Design optimization using genetic algorithm
EMELIE FALKENBERG
MARKUS FREDRIKSSON
Department of Architecture and Civil Engineering
Chalmers University of Technology

Sammanfattning

Materialeffektivitet inom brokonstruktion är ett viktigt forskningsområde för att minska klimatavtryck och initiala kostnader. Ett sätt att öka materialeffektiviteten är att använda lämpliga material där deras egenskaper gynnar konstruktionen. Aluminium har en hög styvhet i förhållande till vikt, hög beständighet och är till stor del återvinningsbart. Dessa egenskaper gör aluminium till ett intressant material för brodäck.

Den initiala kostnaden för brodäcket har hög prioritet bland broförvaltare i Europa. Ett sätt att minimera den initiala kostnaden är att minimera materialåtgången. Syftet med denna avhandling är att utveckla en optimeringsrutin med målet att minimera materialförbrukningen för extruderade aluminiumprofiler för gång-och cykelbrodäck. Däcket utformas för att uppfylla de krav enligt Eurokod. Optimeringen utförs med en genetisk algoritm från verktygslådan för global optimering i programvaran MATLAB. De genererade tvärsnittsgeometrierna utvärderas med implementering av en FE-modul. FE-modulen styrs av ett parametriserat Pythonscript för att skapa en FE-modell och utföra en analys i programvaran ABAQUS CAE för varje iteration.

Optimeringen resulterade i en tvärsnittsgeometri för brodäcket. Detta resultat användes för att göra en kostnadsjämförelse mellan det optimerade tvärsnittet och mer konventionella alternativ i stål och rostfritt stål. Jämförelsen visade att vikten per kvadratmeter för den resulterande optimerade profilen var betydligt lägre jämfört med däckalternativen i stål. Den visade också att besparningar med 8-27% kan göras om aluminiumalternativet väljs istället för ett konventionellt brodäck i rostfritt stål. Aluminium har visat sig vara ett potentiellt alternativ till stål.

Nyckelord: Optimering, extruderade aluminiumprofiler, gång-och cykelbro, brodäck, genetisk algoritm, aluminiumbro, finit elementanalys.

Acknowledgements

This master's thesis was carried out from January to June 2023, at the Department of Architecture and Civil Engineering, Chalmers University of Technology. The thesis comprises 30 credits and concludes the master's program Structural engineering and building technology. The thesis aims to develop a procedure for optimizing hollow aluminium extrusions for pedestrian bridge deck applications with the objective to minimize the area and thus the weight.

First and foremost, we would like to take the opportunity to thank our supervisor, Dr. Peter Nilsson Strand. With his many years of experience in bridge engineering and in research, he has provided invaluable advice throughout the project. We would also like to thank Professor Mohammad Al-Emrani, who with his expertise has supported us both technically and provided the resources required to complete this thesis. Furthermore, we would like to thank our assistant supervisors Benoit Cusson (WSP, Canada) and Henrik Nyström (Norsk Hydro), who with their great knowledge of aluminium in bridge construction and enthusiasm has been of tremendous help to us. Last but not least, we would like to thank everyone at WSPs Bridge department in Gothenburg, who has provided us with everything from office spaces to encouragement during both the ups and downs of this thesis.

Emelie Falkenberg and Markus Fredriksson, Gothenburg, June 2023

Contents

List of Figures

List of Tables

List of acronyms

BC Boundary Conditions.

CSC Cross-Section Class.

DT Destructive Testing.

FE Finite Element.
FSDT First-order Shear Deformation Theory.
FSW Friction Stir Welding.

GA Genetic Algorithms.
GMAW Gas Metal Arc Welding.
gr1 Load group 1 - traffic load.
gr2 Load group 2 - traffic load.

HAZ Heat Affected Zone.

LCC Life-Cycle-Cost.

MIG Metal Inert Gas.

NDT Non-Destructive Testing.

SEK Swedish krona.
SLS Serviceability Limit State.
STR Design situation in ULS.

ULS Ultimate Limit State.
UR Utilization Rate.

List of nomenclature

β	Width to thickness ratio
$\beta_1, \beta_2, \beta_3$	Limit parameters for slenderness
γ_i	Partial factor
γ_M	Partial safety factor for material
ν	Poisson´s ratio
$\phi_j(x)$	Equality constraint function
ψ_i	Factor for variable loads
$\psi_k(x)$	Inequality constraint function
ρ	Density
ρ_c	Local buckling factor
$\rho_{0.HAZ}$	Remaining proportion of proof strength in HAZ
$\rho_{u.HAZ}$	Remaining proportion of ultimate tensile strength in HAZ
$\sigma_{xx,Ed}$	Stress at considered position, transverse extrusion
$\sigma_{yy,Ed}$	Stress at considered position, through thickness
$\sigma_{zz,Ed}$	Stress at considered position, longitudinal to extrusion
$\tau_{xy,Ed}$	Shear stress at considered position, transverse extrusion-through thickness
$\tau_{yz,Ed}$	Shear stress at considered position, through thickness-longitudinal to extrusion
$\tau_{zx,Ed}$	Shear stress at considered position, longitudinal to extrusion-transverse extrusion
ε	Material factor for calculation of cross-section class
ξ_i	Reduction factor for unfavourable permanent loads
b	Width of cross-section part
b_{HAZ}	Extension of HAZ
C_1	Constant
C_2	Constant
E	Modulus of elasticity
$f_i(x)$	Objective function
$f_{0.HAZ}$	0.2% proof strength of aluminium in HAZ
f_0	0.2% proof strength of aluminium
$f_{u.HAZ}$	Ultimate tensile strength of aluminium in HAZ
f_u	Ultimate tensile strength of aluminium
G	Shear modulus
g	Gravitational constant
$G_{k,j}$	Characteristic value of permanent load
L	Length of bridge or loaded length
$q_{acrylate}$	Characteristic self-weight of surface coating

q_{fk}	Characteristic uniformly distributed load for pedestrian bridge
Q_{flk}	Characteristic horizontal load for pedestrian bridge
Q_{fwk}	Characteristic concentrated load for pedestrian bridge
$Q_{k,1}$	Characteristic value of leading variable load
$Q_{k,i}$	Characteristic value of other variable loads
Q_{serv}	Characteristic load representing a service vehicle for pedestrian bridge
R_d	Design value of resistance
R_k	Characteristic value of resistance
t	Thickness of cross-section part

1
Introduction

Aluminium is naturally corrosion resistant and problems related to deterioration and maintenance can consequently be reduced. However, aluminum has a high initial cost and the use of aluminium in bridge construction is not a conventional choice. Limiting the investment cost is of essence to make aluminium a competitive choice. This can be achieved by the use of optimization.

1.1 Background

Bridge authorities in Europe generally consider the initial costs more important to minimize over Life-Cycle-Cost (LCC) (PANTURA, 2011). Repair or replacement of bridge decks is one of the main priorities for concrete and steel bridges, which accounts for a large percentage of rehabilitation activities (PANTURA, 2011). About 70 percent of strengthening activities carried out in Sweden are related to the replacement of bridge decks (PANTURA, 2011). Exploring alternative solutions for bridge decks is therefore important in order to reduce LCC. One solution is to research less conventional materials.

The first application of aluminium in bridge deck construction was in 1933, Pittsburg, USA (Walbridge & de la Chevrotiére, 2019). The introduction of aluminium primarily came from the need to increase the load-carrying capacity by replacing the old and deteriorated bridge decks in concrete, steel, or timber with an aluminium deck (T. Siwowski, 2006; Walbridge & de la Chevrotiére, 2019). Aluminium is a lightweight material with a higher strength-to-weight ratio than steel, even if steel generally has a higher strength. The use of aluminium in modern bridge design tends to mainly be governed by applications where low self-weight is of importance such as in movable bridges, reconstruction of existing bridges and pedestrian bridges.

Aluminium has high corrosion resistance and does not require protective coating (T. Siwowski, 2006). One of the most frequent maintenance measures in Europe for a superstructure in concrete and steel bridges is related to protection against corrosion (PANTURA, 2011). Walbridge and de la Chevrotiére (2019) explains that in comparison with conventional building materials, an aluminium construction can be economically advantageous in terms of lifetime. The entire life-cycle period must be included in order to fairly evaluate costs, from construction and operation to maintenance and recycling. However, the initial material cost of aluminium is high and makes up for a large proportion of the total LCC.

Lightweight structures are of interest for many bridge deck applications and one widely studied structure is the Steel Sandwich element, shown in Figure 1.1. A Sandwich element consists of a top plate, a bottom plate, and a separating core. The separation of the two face plates leads to an increase of bending stiffness whilst the material use is limited, explained by Nilsson (2015), as the "sandwich effect". The Steel Sandwich element requires a great deal of welding using advanced welding techniques such as laser-welding. The core geometry is also limited.

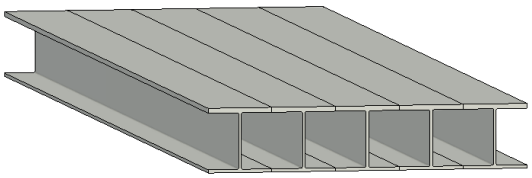


Figure 1.1: *Section of a sandwich element for bridge deck application*

Extrusion is a method for shaping aluminium, where the material is heated and pressed through a die. The production method allows the section to be made as one unit without welds to join the face plates and core. The extruded prismatic aluminum profiles, therefore, provide greater freedom in the shape of the cross-section than for conventional steel solutions (Djedid et al., 2020; Saha, 2000). Previous work on hollow extruded aluminium profiles in bridge decks has shown potential for good performance regarding both bending and torsional stiffness (T. Siwowski, 2006).

There exists a number of suppliers that offer prefabricated bridge deck systems in aluminium. However, the application of these products is either for highway bridges or for pedestrian bridges with an additional underlying load-carrying structure (T. Siwowski, 2006; Walbridge & de la Chevrotiére, 2019). Different aluminium bridge deck systems have also been the topic of several research projects. The performance and calculation methods in various standards and for different aluminium alloys have been evaluated. It has been stated that extruded aluminium is a viable option for vehicular bridge applications(Höglund & Nilsson, 2006; Misch et al., 1999; Saleem, Mirmiran, Xia, & Mackie, 2012; T. Siwowski, 2006).
However, besides that many studies have focused on vehicular bridges, the profiles studied have either been of existing extrusion geometries or direct derivatives of them (Dobmeier, Barton, Gomez, Massarelli, & McKeel Jr., 2001; Massicotte, El-Hage, Lagier, & Fafard, 2020; Misch et al., 1999; Raknes Brekke, 2017; Saleem et al., 2012; T. W. Siwowski, 2009).

Research aiming to develop methods to find optimal geometries for extruded aluminum profiles for pedestrian bridge applications is limited. Especially with regard to the application of Eurocode regulations.

1.2 Aim and Objective

The aim of this master's thesis is to develop an optimization routine for a pedestrian bridge deck constructed of hollow aluminium extrusions. The extruded sections are to be designed to meet the required structural performance according to Eurocode (SS-EN 1990, 2002) and optimized with regard to material consumption. Production costs are then compared with similar decks in steel to evaluate if aluminum is a competitive construction material in such a structure. The objectives are presented below.

- Study structural aspects of aluminium and limitations from production through a literature review. Define an initial parametrically defined core configuration for the optimization.
- Define the design basis and develop a parameterized Finite Element (FE) model that can be used throughout the optimization.
- Perform the optimization using Genetic Algorithms (GA).
- Analyse production costs for the optimized profile and compare with conventional orthotropic steel decks.

1.3 Methodology

In the first part of the project, a literature review is conducted. This is to gain background knowledge about aluminium as a structural material, sandwich structures, production methods, and limitations in production as well as optimization and GA. Existing aluminum profiles and previous work carried out in the field is also studied to identify developments and areas of limited investigation. Further, the basis of design is established, consisting of material properties, loads, design assumptions, and design verifications. This is done by studying Eurocode (SS-EN 1990, 2002) and national standards in addition to the literature review.

The second and largest part of the project is to develop an optimization routine that minimizes the cross-section of a pedestrian bridge deck in aluminium. The initial core configuration of the extruded aluminum profile used for the deck is defined and the cross-section is parameterized. A FE-model is created with ABAQUS for a segment of the bridge deck. The segment is meant to represent the global structural behaviour of the deck in the transverse direction of the bridge. The model is built using Python scripting to be fully adaptable for analysis of all possible cross-section dimensions limited only by the production process. A convergence study is conducted to determine the length of the bridge segment and mesh size.

The optimization routine is created in MATLAB R2021b. The optimization algo-

rithm used in the optimization was GA, which is a part of the Global Optimization Toolbox in MATLAB. The FE-model is implemented as a module to the optimization routine. The design assumptions made for the optimization procedure is then verified for the resulting optimized cross-section.

Finally, the production cost for the optimized profile is analyzed and compared with more conventional alternatives in C-Mn and duplex steels. This is to evaluate whether aluminum is a competitive alternative to conventional orthotropic steel decks.

1.4 Limitations

This thesis is carried out with regard to the listed limitations and simplifications.

- A life cycle perspective of the bridge deck is not considered in the analyses. However, it is discussed.
- The deck is oriented with the extrusions perpendicular to the bridge´s longitudinal direction.
- The bridge deck carries the load transversely to a main longitudinal load-carrying structure. The connections to the main structure are assumed to allow rotations. The deck is thus modeled as simply supported on two edges.
- Linear elastic analyses is performed.
- Welds are neglected in the analysis. Possible welding positions are estimated for the cost comparison of the optimized cross-section.
- Loads are limited to self-weight and traffic loads stated in Eurocode.
- The optimization is performed with aluminium alloy AW-6005A-T6.
- One core configuration is optimized.
- Transportation and on-cite assembly is not included in the cost comparison.

1.5 Outline

This thesis is divided into the following chapters.

Chapter 2 A literature review was carried out to establish a foundation of knowledge for the research topic. It contains information on the state of the art in the field, aluminium properties, production methods, structural behavior of bridge decks, and optimization theory.

Chapter 3 Contains the basis on which the pedestrian bridge deck was designed. Material properties, load and load combinations, design assumptions, and design verification is presented.

Chapter 4 The structural model used for stress analysis in the optimization routine is presented and verified in detail. The pedestrian bridge deck is put into a context and the geometry to be optimized is presented.

Chapter 5 The optimization routine is explained in detail and the resulting optimized cross-section is presented and verified. The structural response of the optimized cross-section is also analyzed. How the FE-model is implemented in the routine, settings of the Genetic Algorithm optimization used, and the design assumptions are verified for the optimized profile.

Chapter 6 The production cost of the optimized aluminium bridge decks is compared to conventional steel and stainless-steel decks in four different materials. The costs are compared, indicating that aluminium is competitive against both ordinary steel and stainless-steel alternatives.

Chapter 7 The results from both the optimization routine and the cost comparison are discussed. Areas of improvement are identified as well as how the limitations might have affected the results.

Chapter 8 Contains a brief summary of the thesis followed by the conclusions made from the findings of the project. Furthermore, topics for future studies are suggested based on the results of this thesis.

2

Literature Review

The literature review is carried out in order to gain knowledge about different parts related to aluminum bridge decks. Including important background information about aluminium as a structural material and its related production methods. Design and theory describing how a bridge deck behaves are presented. Finally, the procedure of optimization is studied.

2.1 Aluminium as a structural material

Aluminium has successfully been used as a structural material for decades in various applications. From an environmental point of view, aluminium, compared to other structural materials, has the advantage that it is to a high degree recyclable. Practically all produced aluminium can be recycled to find new usage areas, with only small material and quality losses. 5% of the energy used to produce primary aluminum is required to recycle used aluminum (Mazzolani, 2004; Norsk Hydro ASA, 2023; Walbridge & de la Chevrotiére, 2019). There are different aluminum alloys developed for different purposes and applications. In the following sections, the standardized designation system used in Europe is described. Alloys of interest and their mechanical properties are presented.

2.1.1 Aluminium alloys

There exist various numbers of aluminum alloys depending on the characteristics required. To categorize aluminum alloys, the designation system according to Eurocode (SS-EN-573-1, 2005) is followed. The designation system is constituted of the prefix *EN*, followed by *A* for aluminium, and *C* or *W*, depending on if the alloys are cast or wrought. Then the constituents of the alloys are specified with four digits (e.g. EN-AW-XXXX). As this report is limited to extruded aluminium profiles, only wrought alloys are under consideration.

Depending on the main alloying element, alloys are grouped into nine different series according to the first digit in the designation system (SS-EN-573-1, 2005). The nine groups as well as their main alloying element, are presented in the table 2.1.

Table 2.1: *Grouping of aluminium alloys and their main alloying element (SS-EN-573-1, 2005).*

Series	Main alloying element
1000	At least 99% aluminium
2000	Copper
3000	Manganese
4000	Silicon
5000	Magnesium
6000	Magnesium and silicon
7000	Zinc
8000	Other elements
9000	Unused series

Further modification of the properties of alloys can be done by changing the metallurgical stage by tempering with different fabrication processes (Mazzolani, 2003). Dependent on what method is used a letter and a number are added to the designation system (Mazzolani, 2003). The most common tempering for alloys used in civil engineering applications is heat treatment. A combination of the following processes is used for heat treatment: solution, tempering, natural aging, and quenching (Mazzolani, 2003). If the alloy has been heat treated the letter T followed by a digit is added behind the chemical composition symbol in the designation system. The digit represents which combination of the processes presented above that has been used.

In a paper by Mazzolani (2004), a comparison between steel and aluminium was made to determine applications in civil engineering where aluminium can be a competitive material. According to the paper mainly alloys within the 6000- and 7000-series have properties necessary for the topic of this thesis. The alloys considered have to be weldable, extrudable, and have properties suitable for load-bearing components in a bridge. According to Eurocode (SS-EN 1999-1-1, 2007), suitable alloys within the 6000-series are AW-6005A, AW-6061, AW-6082 and AW-6106. This corresponds to the alloys used for the aluminium extrusion decks found in the literature presented in the previous section.

Of the alloys mentioned in the 6000-series, temper T6 is most common. Temper T6 indicates that the alloys have been solution heat-treated and artificially aged (Mazzolani, 2003). According to Eurocode (SS-EN 1999-1-1, 2007), AW-6082-T6 provides a higher strength than AW-6005A-T6 while the latter can be extruded into more complex shapes with thinner wall thicknesses.

2.1.2 Aluminum properties

Compared to steel, aluminum has a considerably lower density, $\rho = 2700 kg/m^3$, which is about one-third of that of common mild steel. This property gives aluminium an exceptional strength-to-weight ratio compared to steel. According to

Mazzolani (2004), the weight savings can be as large as 50-70% by choosing aluminium instead of a steel structure. Another advantageous property of aluminum is that it forms a thin inert layer of oxides on its surface when subjected to most environments. Therefore it has natural corrosion resistant properties and does not need a protective coating. However, caution must be taken when in contact with steel as galvanic corrosion between the metals can initiate (Saleem et al., 2012; T. Siwowski, 2006).

The mechanical behavior of aluminum up to the elastic limit is comparable to steel, where it has a linear stress-strain relation although with a different slope, see Figure 2.1 (Mazzolani, 2004). The difference in slope inclination means that aluminum has a lower stiffness. The Young´s modulus is around three times smaller than for steel. After the elastic limit is reached, aluminum has continuous strain-hardening in contrast to steel which has a defined yielding plateau (Mazzolani, 2004). In the absence of a clear yielding plateau, the yield strength is commonly expressed as an elastic limit stress, f_0, called proof strength (Mazzolani, 2003). The proof strength is defined as the stress level where the stress-strain relation offsets from the initial linear relation by 0.2 percent (Mazzolani, 2003). The relation between stress and strain for aluminum can be described by the Ramberg-Osgood law (Mazzolani, 2004). Another characteristic resistant value of aluminium is the ultimate tensile strength f_u. The proof strength and the ultimate tensile strength vary for different aluminium alloys and material thicknesses (SS-EN 1999-1-1, 2007).

The linear thermal expansion coefficients of aluminium alloys are greater than of steel, $23 \cdot 10^{-6}K^{-1}$ and $12 \cdot 10^{-}6K^{-1}$, respectively (SS-EN 1993-1-1, 2023; SS-EN 1999-1-1, 2007). This makes aluminium more sensitive to temperature fluctuation, especially at fixed connections in composite structures where the materials have different thermal expansion coefficients (Mazzolani, 2003).

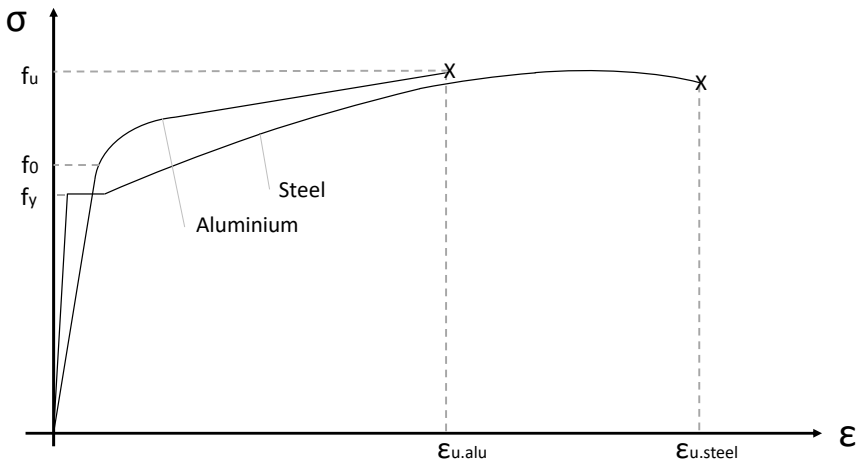


Figure 2.1: *Stress-strain relation for aluminium compared to steel (Mazzolani, 2004).*

2.1.3 Heat affected zone

When aluminium is exposed to high temperatures such as welding the material properties change, leading to a local loss of strength and ductility. This phenomenon occurs up to a certain distance away from the weld and is called Heat Affected Zone (HAZ), see Figure 2.2. The strength loss and the extension of this area are affected by the welding parameters and will therefore vary between different welding methods. Outside the HAZ, the strength values quickly transition to the base metal strength values. To reduce the effect of the HAZ in design welds can be placed in areas with low stresses or bending moments. Another alternative is to increase the material thickness in areas where welds are placed to compensate for the strength loss.

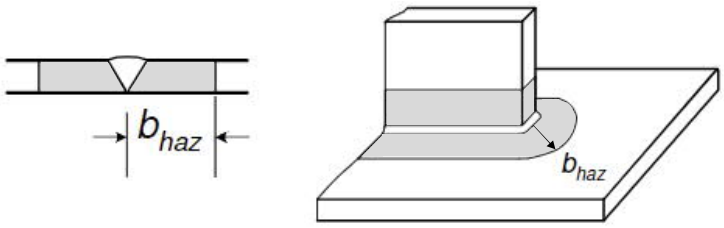


Figure 2.2: *Extent of HAZ in grey (SS-EN 1999-1-1, 2007).*

The softening behavior of the HAZ influence the cross-sectional resistance. To account for the strength loss in design reduction factors can be used to lower the 0.2% proof strength ($f_{0.HAZ}$) and ultimate tensile strength ($f_{u.HAZ}$) of the material, or it can be incorporated as an effective thickness t_{eff} using $\rho_{0.HAZ}$ and $\rho_{u.HAZ}$ (Mathers, 2002; Nazemi & Ghrib, 2019; SS-EN 1999-1-1, 2007). The relationship is visualized with the relation below for a simple rectangular cross-section part ($t \cdot b$). The extent of the area (b_{HAZ}) is determined from the thickness of the cross-sectional part, and the values are presented in Table 2.2 for Metal Inert Gas (MIG) welding. The measurement is defined from the welds center line and propagates in all directions (SS-EN 1999-1-1, 2007).

$$t \cdot b \cdot f_{0.HAZ} = t \cdot b \cdot (\rho_{0.HAZ} \cdot f_0) = (\rho_{0.HAZ} \cdot t) \cdot b \cdot f_0 = t_{0.eff} \cdot b \cdot f_0$$
$$t \cdot b \cdot f_{u.HAZ} = t \cdot b \cdot (\rho_{u.HAZ} \cdot f_u) = (\rho_{u.HAZ} \cdot t) \cdot b \cdot f_u = t_{u.eff} \cdot b \cdot f_u$$

Table 2.2: *Extension of HAZ for MIG welding (SS-EN 1999-1-1, 2007).*

Thickness [mm]	b_{HAZ} **[mm]**
$0 < t \leq 6$	20
$6 < t \leq 12$	30

2.2 Aluminium in bridge construction today

Aluminium has a long history in bridge applications, where the first aluminium bridge deck was constructed in the 1930s (T. Siwowski, 2006; Walbridge & de la Chevrotiére, 2019). In its first application, the original heavy steel deck of the Smithfield Street Bridge was replaced by a significantly lighter aluminium deck. In this manner, by reducing the self-weight of the bridge, allowance for higher live loads was possible (Walbridge & de la Chevrotiére, 2019). This is still one main application for aluminium in bridges, where the original deck is deemed functionally obsolete because of deterioration or increased loading needs(T. Siwowski, 2006; Walbridge & de la Chevrotiére, 2019).

Corrosion of existing concrete and steel bridges has been identified as the major cause of deterioration, especially in the presence of de-icing salt (T. Siwowski, 2006). According to Arrien et al. (2001), in the US alone there are more than 250 000 bridges in service that are in need of rehabilitation. An aluminium deck can replace most bridge decks where no composite action is required (Arrien et al., 2001). This means that aluminium bridge decks can be considered for applications where the deck is simply supported between steel beams. This configuration applies to more than 2700 of the bridges of the Province of Quebec in Canada, leaving the potential market for aluminium as a bridge material to be substantial (Arrien et al., 2001).

The lightness of aluminium decks has several benefits in retrofitting applications, it allows for the widening of the bridge and increasing its capacity without changing the rest of the structural system (Arrien et al., 2001). Furthermore, weight reduction can be a measure to rehabilitate bridges where the support conditions are inadequate without further reinforcing. Aluminium also allows transportation of larger prefabricated segments and hence quicker assembly time (Arrien et al., 2001; T. Siwowski, 2006). This affects not only the direct cost of the construction but also the indirect cost originating from the bridge being inoperable, causing traffic disturbance.

In the earliest implementations of aluminium in construction, there was no alloy available offering both sufficient strength and strong corrosion resistance (Arrien et al., 2001). This changed in the 1960s when new types of extrudable and weldable alloys were introduced whilst still offering exceptional corrosion resistance (Arrien et al., 2001; T. Siwowski, 2006). With this background, the interest in aluminium bridge decks increased and over the years several different concepts have been developed. In the following sections, a selection of existing aluminium deck profiles is presented.

2.2.1 Alcoa bridge deck system

The Alcoa bridge system resembles an orthotropic steel deck, where extruded AW-6061-T6 profiles are welded to an aluminium top plate (Arrien et al., 2001). The deck system was developed in the mid 1960s. The lower part of the main profile wedges is clamped between steel plates and the steel girder with galvanized bolts. The panels are prefabricated and connected together at the ends with bolts through a C-shape extrusion profile, see Figure 2.3. According to Arrien et al. (2001), due to extensive amounts of welding the deck system is prone to fatigue and expensive to produce.

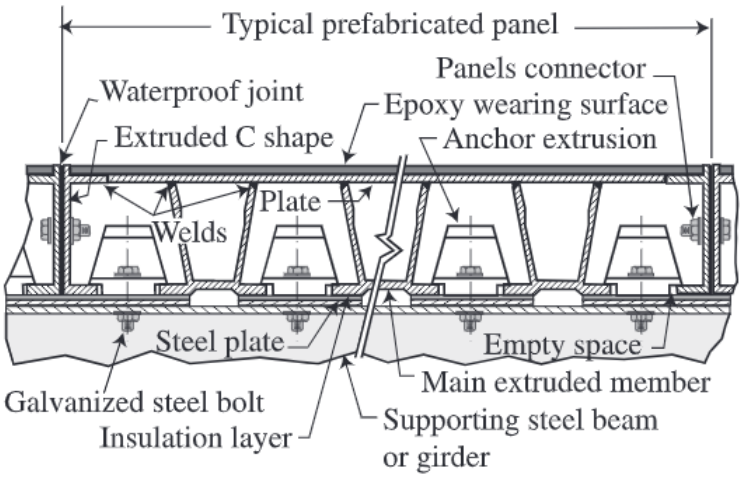


Figure 2.3: *Alcao bridge deck system, illustration of a typical prefabricated panel. The Figure is taken with permission from (Arrien et al., 2001).*

2.2.2 AlumaDeck / Reynolds deck system

The AlumnaDeck, also known as the Reynolds Deck system, was developed in the mid 1990s by the Reynolds Metal company (Arrien et al., 2001). In agreement with the Virginia Department of Transportation, the new deck system was implemented as a research project by replacing a functionally obsolete deck on a highway bridge in Virginia (Misch et al., 1999). The research project contained physical tests in a laboratory environment where Dobmeier et al. (2001) tested the static response of a deck panel, and installed on the bridge where Misch et al. (1999) tested the dynamic and static behavior of a deck in service.The studies showed that the Reynolds aluminium deck is a viable alternative to reinforce concrete decks for vehicular bridge applications.

The deck is constructed of two-voided extrusions in aluminium alloy 6063-T6, shown in Figure 2.4. The single extrusion profiles are welded together in the top and bottom flange forming prefabricated panels in the desired dimensions. The deck system is stiffer in the direction of the extrusions, hence the orientation of the extrusions is usually placed parallel to the bridge span (Misch et al., 1999; Zhang, Qiu, & Bai, 2012). The aluminum deck is placed on top of steel girders and connected with shear studs followed by an injection of grout in the compartments with shear studs to assure composite action between the material (T. Siwowski, 2006).

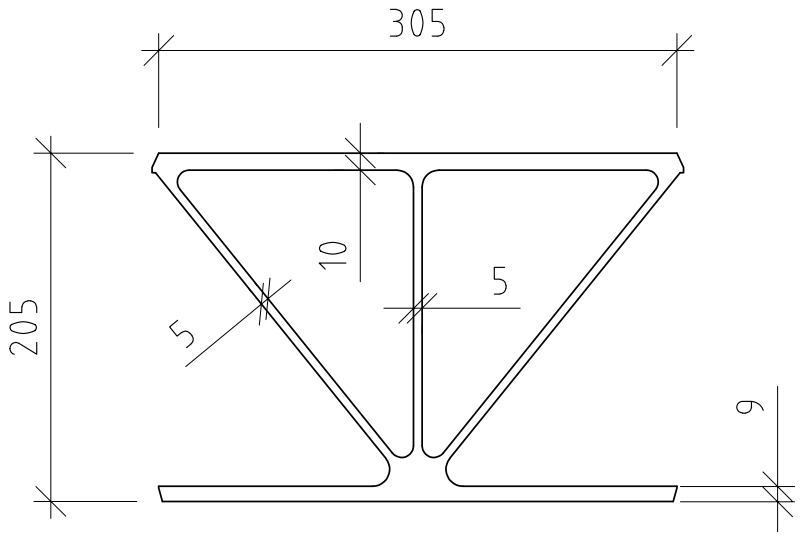


Figure 2.4: *AlumaDeck / Reynolds bridge deck system, illustration of the two-voided extrusion. The illustration is inspired by a figure in (Misch et al., 1999) that has been reworked by the authors.*

2.2.3 The Svensson deck / SAPA deck system

In the 1980's another bridge deck system of extruded aluminium was developed in Sweden called the Svensson deck (now being sold under the name SAPA) (Arrien et al., 2001). The deck was made in two different dimensions now named SAPA SFG250/53 and SAPA SFG300/100 (Hydro Building Systems Sweden AB, 2022). The cross-section, shown in Figure 2.5, consists of multi-voided extrusions that are connected to each other mechanically by a tongue and a groove connection in the top flanges while the bottom flanges are clamped to the main girders with bolts (T. Siwowski, 2006). This type of connection allows for rotation between the separate panels which can be beneficial compared to welded connections in terms of fatigue (Arrien et al., 2001). Furthermore, Arrien et al. (2001) claims, that limiting the amount of welding results in a significantly cheaper deck to produce. However, after years of service, problems with cracks in the wearing surface have appeared to originate from the allowance of the rotation between neighboring panels (Arrien et al., 2001). This is also confirmed by E. Rossell (personal communication, 16 February 2023) at the *Swedish Transport Administration*, who helped gather information from bridge inspectors.

The bridge decks are designed to be oriented perpendicular to the flow of traffic (Hydro Building Systems Sweden AB, 2022). The smaller of the two profiles are designed for a maximum distance between girders of 0,8 m, while the larger can be used with a distance is up to 2,6 mm between the girder. According to Arrien et al. (2001), this system is one of the lightest on the market where the self-weight is 41.7 kg/m^2 and 60 kg/m^2, for SFG100/53 and SFG300/100, respectively (Hydro Building Systems Sweden AB, 2022). The alloy used for the extrusions is AW-6063-T6 (Arrien et al., 2001).

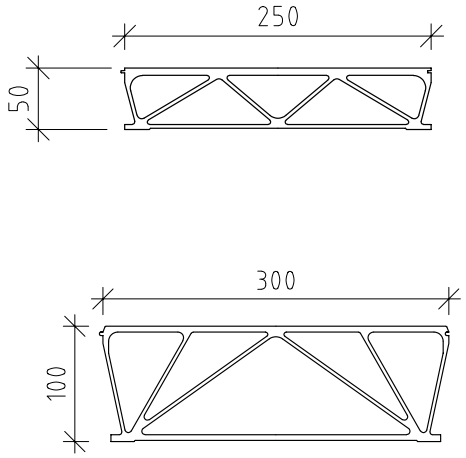


Figure 2.5: *Deck system SAPA SFG250/53 (top) and SFG300/100 (bottom). The illustration is inspired by a Figure in (T. Siwowski, 2006) that has been reworked by the authors.*

2.2.4 Vehicular bridge deck developed at Rzeszow University

In a research project reported by T. W. Siwowski (2009), the structural behavior of an extrusion cross-section developed for a vehicular bridge deck was evaluated. The cross-section of the extrusion is shown in Figure 2.6. The profile was designed on the basis of existing solutions but is now optimized for the production possibilities and domestic requirements in Poland. The chosen aluminium alloy for the profiles was AW-6005A-T6. This alloy was chosen because of its insensitivity to corrosion and its mechanical properties (T. W. Siwowski, 2009).

The deck is formed by MIG welded connection of the individual profiles forming a geometrically orthotropic deck (T. W. Siwowski, 2009). Furthermore, T. W. Siwowski (2009) explains that the main direction of the deck usually is oriented with the flow of traffic to generate composite action between girders and the deck. The deck was tested in a laboratory environment where a segment of the deck was tested to failure. In the experiment, the segment was set-up simply supported on the edges along the extrusion direction and the load was applied in the middle as a patch load. The Load tests of the developed deck panel showed that failure began to occur locally where the load was applied as the top flange began to yield. After further loading the ultimate load was reached when the material fractured in the HAZ.

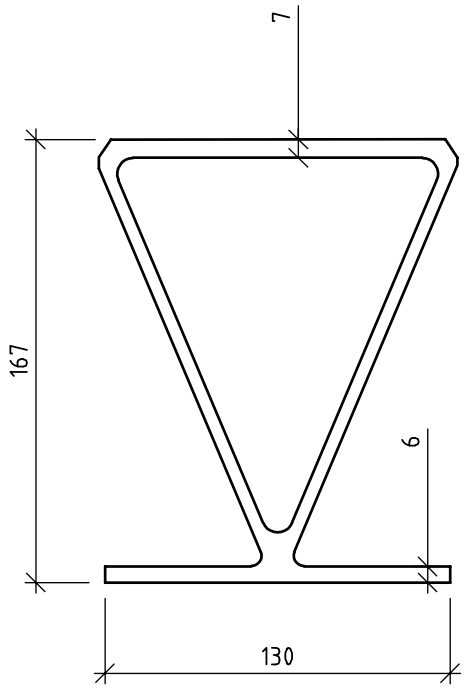


Figure 2.6: *Cross-section of extrusion profile developed by T. W. Siwowski (2009). The illustration is inspired by a Figure in (T. W. Siwowski, 2009) that has been reworked by the authors.*

2.2.5 Open-hollow bridge deck developed at Osaka University

Another concept for highway bridges was developed in Japan by Okura et al. (2006). The aluminium deck was designed to be supported on steel girders separated by a maximum distance of 4m from each other. The extrusion direction was oriented in the transverse direction of the span of the bridge. Contrary to the other presented bridge decks, the extruded profiles are open-hollow. Which is more like a conventional orthotropic plate with closed stiffeners. The motivation behind using open-hollow cross-section comes from that the Japanese Guideline for Quality Inspection requires that joints formed using Friction Stir Welding (FSW) should be checked from both sides (Okura et al., 2006). To be able to use the advantages of FSW while still being in agreement with the guidelines, the open-hollow cross-section shown in Figure 2.7 was developed. The extrusion profile presented is only welded in the top flange while a gap between the bottom flanges of each individual profile remains open. The chosen alloy for the deck profiles was AW-6061-T6. The dimensions of the cross-section were limited to a thickness of 15mm to assure quality in the extrusion and FSW process. Load testing of the deck confirms what has been reported in other studies as well, high local bending stresses appear at the position of applied wheel pressure.

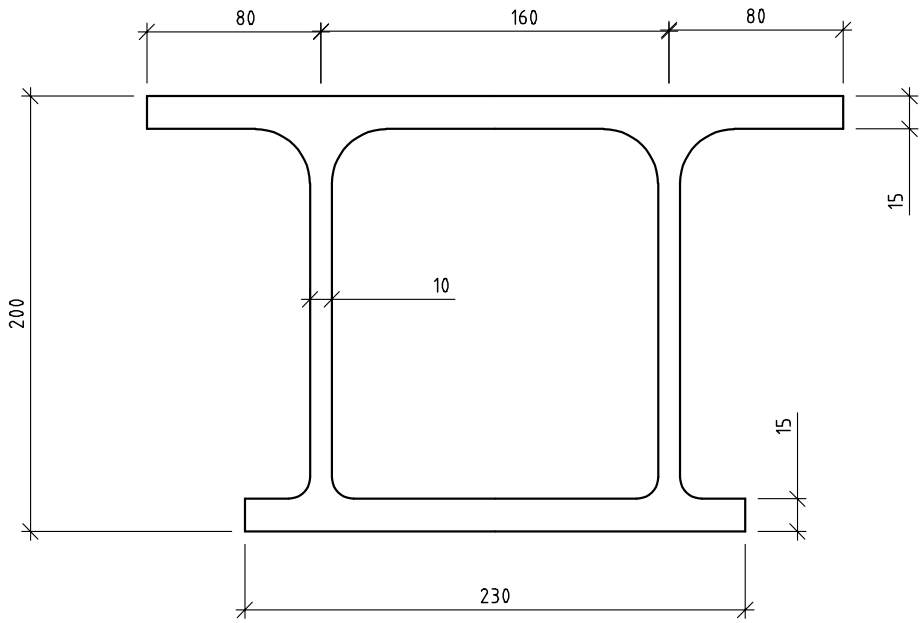


Figure 2.7: *Cross-section of the extrusion profile developed by Okura et al. (2006). The illustration inspired by a Figure in (Okura et al., 2006) that has been reworked by the authors.*

2.2.6 Bridge deck system developed at Laval University

A paper by Djedid et al. (2020) presents the design of an extruded aluminium bridge deck that is meant to be supported on steel girders for highway applications. The aluminium alloy used is AW-6005A-T61, for its mechanical properties, weldability, and extrudability. The profile is optimized with regard to fatigue strength in both welded and non-welded areas (Djedid et al., 2020). The extrusion direction of the deck system is designed to be oriented along the direction of the traffic flow. The system uses a combination of different mechanical joints where adjacent extrusion panels on intermediate girders are connected with a male-female joint and the deck is connected to the steel girders with a type of mechanical clamps. With the use of clamps, free expansion longitudinally of the deck is allowed (Cusson, 2022). Hence, no-composite action between the two components is possible. This is a design choice made because the thermal expansion of aluminium is approximately twice that of steel (Cusson, 2022). Individual extrusion profiles in the span between girders are welded together using gas metal arc welding (GMAW). The combination of different joints used in the system lead to the need for five different extrusion profiles. In Figure 2.8, the main profile used in the span between girders is presented.

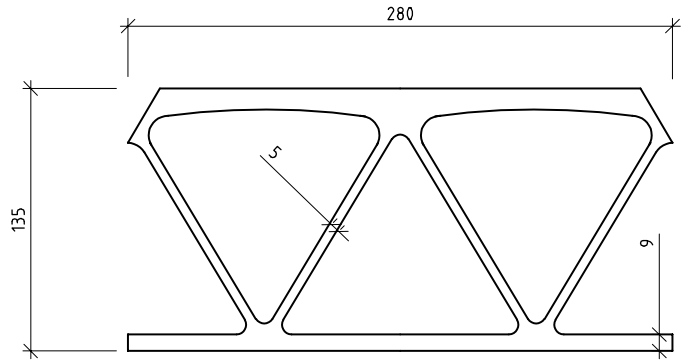


Figure 2.8: *Cross-section of the extrusion profile developed by Djedid et al. (2020). The illustration is inspired by a Figure in (Djedid et al., 2020) that has been reworked by the authors.*

2.3 Bridge deck

The obvious function of a bridge deck is to enable passage, on foot or by bicycle in this case. However, the bridge deck also has several load-bearing functions. It should be able to carry local loads, e.g. from a service vehicle, and through plate action, it should be able to distribute the loads to the supporting structure. In some cases, composite action is utilized where the deck act as a flange to the main load-carrying structure. This increases the global stiffness and resistance (Nilsson Strand, 2020). The structural system can be made with various shapes and different materials can be used, where the most conventional are concrete, steel and timber.

2.3.1 Sandwich panels

Sandwich panels have been used for over 100 years, but mass production for commercial applications started first during the Second World War for mainly aircraft production. In the following years, theoretical research on sandwich construction began to be carried out (Zenkert, 1997).

A sandwich panel is composed of three parts, two thin face plates with high strength and stiffness. Between these plates is a thick and lightweight core. This type of structure provides a high stiffness-to-weight ratio and the utilization of material is efficient. To obtain a lightweight core two parameters can be altered in the process, material properties and geometry. It is not uncommon to use a low-density material to limit the weight but if the same material is to be used in all parts of the element it makes the geometrical design a vital part of creating a lightweight core (Zenkert, 1997).

The structural behavior of a sandwich plate is somewhat like an I-beam. A large amount of material, which is represented by the flanges of the I-beam or the face plates of the sandwich panel, is located at a distance from the neutral axis. This results in an increased bending stiffness and the idea is that the face plates carry load in bending. The idea with the web of the I-beam and the core of the sandwich panel, besides separating the plates, is to withstand shear action with a minimum amount of material. The core in a sandwich panel, in contrast to an I-beam, gives rise to evenly spaced supports for the face plates (Nilsson, 2015; Zenkert, 1997).

2.3.2 Plate theory

Several papers and research cover the structural behavior and mechanisms in terms of bending, buckling, and dynamics of a sandwich plate (Allen, 1969; Reddy, 2004; Zenkert, 1995). An overview of aspects, theories, and literature is given by Vinson (2001). The material of which a sandwich panel is made may be isotropic but the structure has a certain level of orthotropy and the transverse shear stiffness is generally low. Therefore the addition of transverse force can not be neglected in the calculation of the total deformations. Libove and Batdorf (1948) derived a small-deflection theory for orthotropic sandwich plates which includes transverse shear deformation. This theory is following the first-order shear deformation theory (FSDT), also called Reissner-Mindlin theory, with the basis that plane sections originally normal to the mid-plane remain plane but not normal to the mid-plane. This implies a rotation of the plate section and constant shear strains over the thickness of the plate (Agarwal, Broutman, & Chandrashekhara, 2018; Zenkert, 1995).

2.3.3 Core configurations

A large number of core configurations exist for sandwich panels. As many materials require adhesive joints or welding to connect the parts, many profiles have been adapted with an extra area (flange) to make this possible, for example, a C-core or

trapezoidal-core. Aluminium does not require this extra area for joining since it can be extruded as one unit. Below some common core configurations and their main characteristics that could be suitable for an extruded sandwich plate are presented. The main difference between them relates to material intensity, the number of contact points between face plates and core, and shear stiffness perpendicular to the extrusion.

The most basic geometry is the web-core, consisting of only vertical parallel plates between the face plates, see Figure 2.9. This layout provides discrete support in the direction perpendicular to the extrusion and the level of orthotropy is high. (Romanoff, 2007). The transverse shear stiffness of sandwich plates with web-core is generally low (Romanoff & Varsta, 2005). This design also makes it vulnerable to local deflection which entails concentrated secondary bending stresses, studied by Romanoff (2007). Even if the stiffness-to-weight ratio is considered high for a sandwich plate with web-core, the overall bending stiffness is low compared to other core geometries (Nilsson, 2015). The simple design makes it possible to define the web plate with only two variables, thickness, and height.

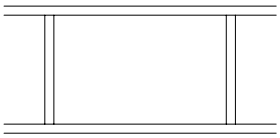


Figure 2.9: *Sandwich panel with web-core.*

Libove and Hubka (1951) were the first to derive stiffness constants for a sandwich plate with a corrugated core. This has since been investigated for triangular cores by Lok and Cheng (2000) and for sinusoidal cores by Bartolozzi, Pierini, Orrenius, and Baldanzini (2013). See Figure 2.10 for the two corrugated core geometries, triangular/truss and sinusoidal. The material intensity is significantly higher compared to the web-core. However, the transverse shear stiffness increase for an inclined web plate (Lok & Cheng, 2000; Nilsson, Al-Emrani, & Atashipour, 2017). Inclination of the core plates yields in a slight decrease of the longitudinal bending stiffness that, in combination of increased transverse shear stiffness, decreases the level of orthotropy (Nilsson, 2015). In total that yields in a increased over-all bending stiffness (Nilsson, 2015). A greater amount of variables are needed to define the core geometry, for instance, thickness and angle of inclination of the triangular core, and for the sinusoidal core also the radius is needed.

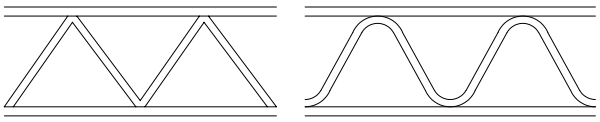


Figure 2.10: *Sandwich panel with triangular- (left) and sinusoidal-core (right).*

Another category of core geometries is the multi-supporting cores with V- and Y-

shaped core plates, see Figure 2.11. Both geometries provide twice as many contact points between the top face plate and core compared to the bottom face plate and core. They are less material dense than corrugated cores but are still beneficial for local loads due to the smaller distance of support positions to the top plate. The geometry of the V-core can be defined by the thickness and angle of inclination of the web plate and the distance between these plates. However, the Y-core requires several more variables; thicknesses, the height of the leg of the Y, the angle of the crown of the Y, and total height, as well as center distance.

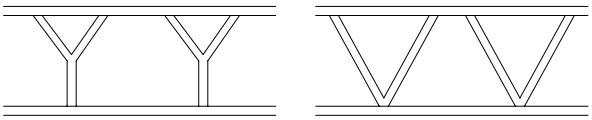


Figure 2.11: *Sandwich panel with Y- (left) and V-core (right).*

2.4 Production methodology

There exist several processes for forming and joining aluminium. In this part, only the production method of extrusion is studied and possible welding methods are overviewed. The entire bridge deck can be prefabricated for easy assembly. To form the bridge deck, extruded profiles can be joined together to create plate segments in manageable sizes. The segments can then be joined to the appropriate length suitable for the purpose, see Figure 2.12.

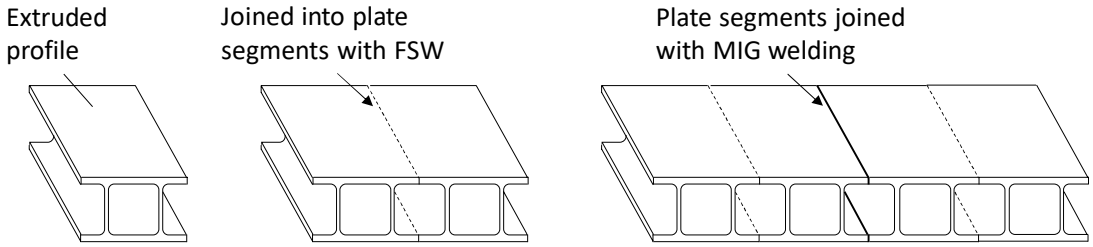


Figure 2.12: *Production process of a bridge deck made of extruded profiles.*

2.4.1 Extrusion

Formability is one main advantage of aluminium, enabling the production of complex profiles for a variety of applications. Extrusion is the plastic deformation process, where preheated aluminium billets are pressed through a die with the desired cross-section, see Figure 2.13. This process takes place under great compressive force. For hollow profiles a two-part die is used; an outer part that shapes the outside and an inner part named a mandrel to shape the cavity. The profile is continuously formed into segments of maximum 45 meters and with a constant cross-section (Nyström, n.d.). To prevent imperfections and stresses from production, the profiles are stretched immediately after extrusion and cooling (Norsk Hydro ASA, 2023;

Saha, 2000). The extruded sections can then be cut into desired lengths and welded with a suitable method.

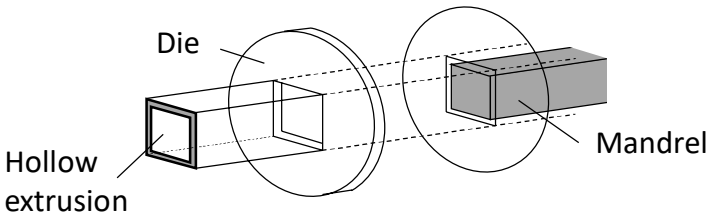


Figure 2.13: *Extrusion press with a two-part die.*

The industrial company Norsk Hydro ASA (Norsk Hydro ASA, 2023) has pointed out some design considerations regarding the extrusion process. Simple and symmetrical profiles are easier to extrude and are therefore less expensive. Less variation in the wall thickness of the profile also makes the extrusion process easier and cooling more even. However, it is possible to have a varying thickness. In some cases it can even be beneficial to have a varying thickness, for instance, the bending resistance can be increased by having a thicker top and bottom surface while the inner parts have a lower thickness. Finally, all corners of the profile should be rounded off, a radius as low as 0.5-1 mm is acceptable. The dimension of the press will limit the geometry of the extruded sections. According to Norsk Hydro ASA (Norsk Hydro ASA, 2023) the maximum outline is limited to a width of 620mm for a height of 50mm or a diameter of 320mm for their largest press. Figure 2.14 illustrates the shape of the press.

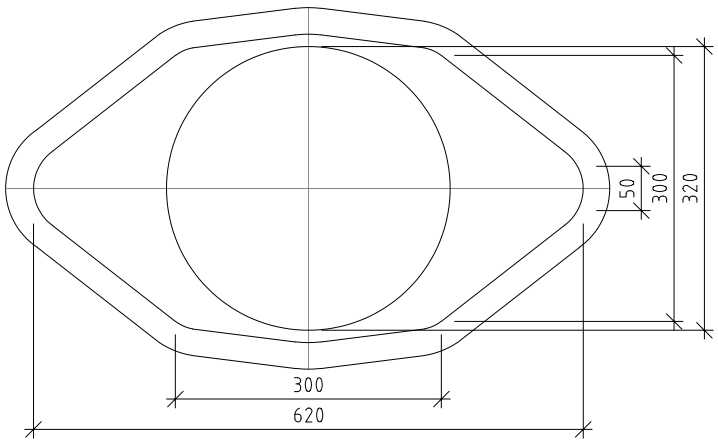


Figure 2.14: *Outlines of the extrusion press. Created by authors, inspired by Norsk Hydro ASA (2023).*

2.4.2 Welding

GMAW, where MIG is the most common welding method for joining parts of aluminium. It is a type of fusion welding process, where the parts are melted together.

The weld is created by an electric arc with a continuously fed wire as the electrode. This method does not only heat up and melt the base material, but the wire also provides filler material to the connection. Shielding gas is protecting the arc and weld pool during welding. Calculation and verification of GMAW is covered by Eurocode.

FSW is a relatively new method for joining metals, invented in 1991 by The Welding Institute in the UK (Mishra & Ma, 2005). The method adopts a rotating tool that both heats up and adds pressure to the material, suitable for flat plates. The tool is made of a non-consumable material consisting of a probe and shoulder. The heat is generated by friction from the rotating tool and causes the surrounding metal to plasticize. It is a solid-state process, meaning that the temperature is kept below the melting point of aluminium. The tool moves along the weld joint while the probe pushes softened material between the plates toward the moving direction. The process created a homogeneous weld with a flat surface (Mishra & Ma, 2005; Threadgilll, Leonard, Shercliff, & Withers, 2009). The heat input is limited which results in a smaller HAZ. The lower temperature also results in less strength loss even if a decrease does take place (Mathers, 2002). At present there is no published standard covering the calculation and verification of FSW.

As FSW provides lower strength loss of the base material it is a preferable method to use. However, the use of FSW is limited by the width of the production rig. A combination of the two methods can be used, FSWto join extruded profiles to create manageable deck segments and MIG welding to join the segments into the desired length of the bridge deck.

To provide for weldability, a minimum target for the plate thickness is normally specified to 3mm. Other preparations needed for single-sided welding of aluminum are backing and edge preparation (SS-EN 1011-4, 2001), see Figure 2.15. The cavity formed under the weld line is intended for contamination. These preparations are applied for MIG welding according to SS-EN 1090-3 (2019), but FSW may also need some preparation in terms of extra support under the weld line due to the pressure from production.

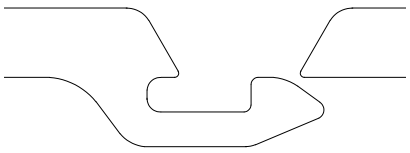


Figure 2.15: *Principal design of a permanent backing and edge preparation for single-sided welding. Created by authors, inspired by Eurocode (SS-EN 1090-3, 2019).*

2.5 Engineering optimization

Optimization means searching for an optimal solution. The method for finding an optimal solution varies for different problems and depends on what is being sought

and what limitations exist, which are called objectives and constraints. In this thesis, the objective is minimizing the use of material while meeting the required structural performance. The objectives and constraints of an optimization determine the complexity of the problem to be solved (Yang, 2010).

On generic form the optimization problem can be described with the following problem functions (2.1, 2.2, 2.3) and variable vector (2.4) (Yang, 2010). $f_i(x)$ is the objective function with M number of objectives, $\phi_j(x)$ and $\psi_k(x)$ are the equality and inequality constraint functions with J and K number of constraints, respectively. The vector x contains n number of independent decision variables. In addition, the domain normally needs to be limited by an upper and lower bound of the decision variables.

$$\text{Minimize} \quad f_i(x) \qquad i = 1, 2, ..., M \tag{2.1}$$

$$\text{Subject to} \quad \phi_j(x) = 0 \quad j = 1, 2, ..., J \tag{2.2}$$

$$\psi_k(x) \leq 0 \quad k = 1, 2, ..., K \tag{2.3}$$

$$\text{With} \quad x = (x_1, x_2, ..., x_n)^T \tag{2.4}$$

An optimization can be classified depending on its properties. One way is to separate between single objective ($M = 1$) and multiobjective ($M > 1$). Single objective implies that only one objective is sought while multiobjective consists of several objectives. A multiobjective optimization can be to optimize for costs and material consumption simultaneously. In extreme cases, with many conflicting targets, this kind of problem can be very complicated to solve and are according to Yang (2010) then called black-box problems. Further, optimization can also be classified depending on the number of defined constraints. One special case is when there are no defined constraints, unconstrained ($J = K = 0$), which according to Yang (2010) generally is a simpler problem to solve. All other cases with defined constraints fall into the subcategory constrained ($J + K \geq 1$). The problem is called equality constraint if $J \geq 1$ and $K = 0$, and inequality constraint if $K \geq 1$ and $J = 0$. The different classifications of optimization can be seen in Figure 2.16.

$$Optimization \begin{cases} Objective & \begin{cases} Singleobjectives \\ Multiobjectives \end{cases} \\ Constraint & \begin{cases} Unconstrained \\ Constrained \end{cases} \end{cases}$$

Figure 2.16: *Classification of optimization (Yang, 2010).*

An algorithm is a sequence of mathematical instructions, explaining how the search of a solution is to be performed. Optimization algorithms can either be classified

as deterministic or stochastic. A deterministic algorithm is repeatable and will follow the same path and give the same solution every time for a fixed set of decision variables and problem functions. However, a stochastic algorithm is based on some degree of randomness, where the solution will vary for each run (Yang, 2010). Stochastic algorithms can be subdivided into heuristic and metaheuristic. Heuristic processes aims to find a solution by trial and error, this method does not necessarily provide the most optimal solution but it is computationally efficient. Metaheuristic, on the other hand, is a higher-order method that uses both randomization and local search. Randomization reduces the risk of local search and this method gives a more optimal solution (Yang, 2010). Classification of optimization algorithms can be seen in Figure 2.17.

$$Algorithms \begin{cases} Deterministic \\ Stochastic \quad \begin{cases} Heuristic \\ Metaheuristic \end{cases} \end{cases}$$

Figure 2.17: *Classification of algorithms (Yang, 2010)*

2.5.1 Genetic algorithm

GA is a metaheuristic method within stochastic optimization algorithms. It was first developed in the 1960s by Holland (1992). He took inspiration from nature and Darwin´s evolutionary theory based on natural selection and reproduction. The process of natural selection implies that more suitable individuals survive and can reproduce, and reproduction provides new combinations of individuals based on the characteristics of the previous generations and with consideration to mutation (Holland, 1992). In a GA, a potential solution is named individual, while a set of individuals are named population. Sivanandam and Deepa (2008) describes that the optimization starts with an initial population randomly generated, the initial population will be different for each run of the program. After each round, every individual in the population will get a fitness value assigned which describes their suitability. This value is the basis for selecting the best solutions and passing them on to create new populations. New populations are guided by the fitness value but also generated with some randomization. The procedure is repeated with more and more suitable individuals until a final solution is reached (Sivanandam & Deepa, 2008).

According to Yang (2010), GA have the ability to solve complex optimization problems, and Holland (1992) argues that the algorithm can solve problems that humans can not even comprehend. However, Yang (2010) points out some drawbacks. In order for the algorithm to converge, appropriate choices must be made for the selection criteria and rate of mutation for new populations. If the initial population

size is too narrow the algorithm can get stuck on a local optima and miss the global optima. If it is too wide the computational time will increase.

3

Basis of Design

This chapter presents the design basis on which the following analyses are carried out. Material properties of the chosen aluminium alloy, loads acting on a pedestrian bridge, and load combinations are presented. Design assumptions and design verifications used during and after the optimization is described. The following sections are based on Eurocode and additional National regulations presented below.

- SS-EN 1990 (2002)
- EN 1990 Annex A2 (2004)
- SS-EN 1991-2 (2003)
- SS-EN 1993-1-1 (2023)
- SS-EN 1993-2 (2006)
- SS-EN 1999-1-1 (2007)
- SS-EN 1999-1-2 (2007)
- Krav brobyggande, Trafikverket (2019)

3.1 Material properties

The aluminium alloy used in the optimization is AW-6005A-T6. Its specific material properties are presented in Table 3.1. In the Table f_{HAZ} represent the reduced proof strength and ultimate tensile strength, respectively, in the the heat affected zone while ρ_{HAZ} (f_{HAZ} / f) represents the remaining proportion of strength compared to unwelded material. The proof strength and ultimate tensile strength are dependent on the thickness of the material. The remaining material constants used are presented in Table 3.2 (SS-EN 1999-1-1, 2007).

Table 3.1: *Characteristic strength values for unwelded material and material in the HAZ (MIG welding), (SS-EN 1999-1-1, 2007).*

	Thickness t [mm]	f_0 [MPa]	f_u [MPa]	$f_{0.HAZ}$ [MPa]	$f_{u.HAZ}$ [MPa]	$\rho_{0.HAZ}$	$\rho_{u.HAZ}$	Buckling class
6005A	$t \leq 5$	215	255	115	165	0.53	0.65	A
	$5 < t \leq 10$	200	250			0.58	0.66	

Table 3.2: *Material constants according to SS-EN 1999-1-1 (2007).*

Modulus of elasticity	E	70	[GPa]
Poisson´s ratio	ν	0.3	

According to SS-EN 1999-1-1 (2007), Aw-6005A has a durability rating B. For durability rating B and for plate thicknesses larger than $3mm$ no corrosion protective measures are needed except in harsh marine environments. In harsh marine environments it would have to be investigated from case to case.

3.2 Load and load combinations

In the design of pedestrian bridges, the loads acting on the structure are divided into permanent and variable loads. Permanent loads are such loads that constantly affect the structure, these consist of the structure's own weight. Variable loads are in accordance with (SS-EN 1991-2, 2003) for pedestrian bridges. The variable loads are divided into static vertical and horizontal loads. The static vertical load models consist of a uniformly distributed load, a concentrated load and a service vehicle. Vertical loads are free action, meaning that the load models should be placed so that the most unfavorable load effects are obtained. The horizontal force represents a breaking force. Vertical and horizontal forces are combined into load groups. The load groups should be treated as separate characteristics actions. Both the permanent and variable loads should be combined to determine the static load effects in Ultimate Limit State (ULS) and Serviceability Limit State (SLS), according to Eurocode (EN 1990 Annex A2, 2004; SS-EN 1990, 2002). The different loads are presented in detail below.

3.2.1 Self-weight

The self-weight acting on the structure includes the total weight of the bridge deck and surface coating. The load of the deck is calculated from the density of aluminium ρ multiplied by the gravitational constant g. The surface coating on top of the deck consists of a 10 mm thick layer of acrylate in accordance with the National Annex (Trafikverket, 2019).

$$\begin{aligned} \rho &= 2700 \quad [kg/m^3] \\ g &= 9.82 \quad [m/s^2] \end{aligned}$$

$$q_{acrylate} = 22 \quad [kN/m^3]$$

3.2.2 Uniformly distributed load

For dimensioning of the bridge deck, a uniformly distributed surface load q_{fk} is used in the design. This represents the load effect of pedestrian traffic and is based on pedestrians or cyclists regular or occasional presence on the bridge. The characteristic uniformly distributed load is obtained from equation 3.1 (SS-EN 1991-2,

2003).

$$2.5 \leq q_{fk} = 2.0 + \frac{120}{L + 30} \leq 5.0 \quad [kN/m^2] \tag{3.1}$$

Where,

L Loaded length in [m]

3.2.3 Concentrated load

The recommended value for the characteristic concentrated load Q_{fwk} is 10 kN, distributed over a square area corresponding to 0.01 m^2 (0.1 x 0.1 m). However, if a service vehicle is allowed to drive on the bridge, the concentrated load Q_{fwk} is replaced by the load model representing the service vehicle Q_{serv} (SS-EN 1991-2, 2003).

$$Q_{fwk} = 10 \quad [kN]$$

3.2.4 Service vehicle

If no permanent obstacle prevents a vehicle from passing the bridge, the load model representing a service vehicle Q_{serv} shall be considered. The model describes a vehicle with two axle-pairs at a distance of 3 m. The distance between the wheel-centres is 1.3 m and the contact area is 0.2 m x 0.2 m. Recommended characteristic load from the first and the second axle is $40kN$ and $80kN$, respectively (SS-EN 1991-2, 2003). The outline of the service vehicle is shown in Figure 3.1.

$$Q_{serv.1} = 80 \quad [kN]$$
$$Q_{serv.2} = 40 \quad [kN]$$

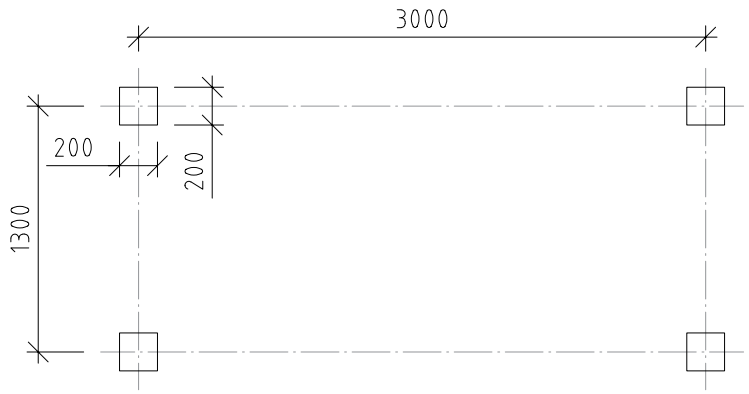


Figure 3.1: *Geometry of a service vehicle.*

3.2.5 Horizontal load

The static model for horizontal loads to be considered is the designing value of the following two cases. Either 10% of the total load coming from the uniformly distributed load on the bridge or 60% of the total weight of the service vehicle. The load should be applied on the surface of the deck and act in the direction of the bridge (SS-EN 1999-1-2, 2007).

3.2.6 Groups of traffic loads

Loads that are considered to act simultaneously should be combined in load groups. The load groups are treated as separate characteristic actions and should be combined in the same way as other loads. Two groups are defined for pedestrian bridges. Group 1 (gr1), consisting of uniformly distributed load q_{fk} and horizontal breaking force Q_{flk}. Group 2 (gr2), consists of the load representing a service vehicle Q_{serv} and horizontal breaking force Q_{flk} (SS-EN 1991-2, 2003). The load groups can be seen in Table 3.3.

Table 3.3: *Definition of load groups.*

Load type		**Vertical load**		**Horizontal load**
Load system		**Uniformly distributed load**	**Service vehicle**	
Groups	gr1	q_{fk}	0	Q_{flk}
of load	gr2	0	Q_{serv}	Q_{flk}

3.2.7 Load combinations

The procedure for combining loads used for verifying the design of the bridge is regulated in SS-EN 1990 (2002) and accompanying Annex A2 *Application for bridges* in EN 1990 Annex A2 (2004). The load combinations consist of both permanent and variable loads, where the permanent load is defined as either *unfavourable* or *favourable*, and the variable loads are divided into *leading* and *other* actions. Partial factors γ_i , factors for variable loads ψ_i, and reduction factor for unfavorable loads ξ_i , together with design values for actions can be found in Appendix A. Load combinations for ULS and SLS are presented below.

ULS is the state concerning the safety of people and the structure itself. In ULS, the design situation STR is considered which means that the strength of the material is governing failure and result in internal failure or excessive deformation. The least favorable load effects gained from the two equations (3.2 and 3.3) are to be used in the design.

$$\sum_{j\geq 1} \gamma_{G,j} G_{k,j} + \gamma_{Q,1}\psi_{0,1} Q_{k,1} + \sum_{i>1} \gamma_{Q,i}\psi_{0,i} Q_{k,i} \tag{3.2}$$

$$\sum_{j\geq 1} \xi_j \gamma_{G,j} G_{k,j} + \gamma_{Q,1} Q_{k,1} + \sum_{i>1} \gamma_{Q,i}\psi_{0,i} Q_{k,i} \tag{3.3}$$

Where,

$G_{k,j}$ Characteristic value of permanent load
$Q_{k,1}$ Characteristic value of leading variable load
$Q_{k,i}$ Characteristic value of other variable loads
γ_i Partial factor
ψ_i Factor for variable loads
ξ_i Reduction factor for unfavorable permanent loads

SLS is the state concerning user comfort and structural function at normal use. The frequent load combination is used for a reversible limit state, which means that no effects on the structure will remain after unloading. Equation 3.4 is to be used for verification of deflections in SLS. In contrast to ULS load combinations, all partial factors for material and loads are set to 1.0.

$$\sum_{j\geq 1} G_{k,j} + \psi_{1,1} Q_{k,1} + \sum_{i>1} \psi_{2,i} Q_{k,i} \tag{3.4}$$

Where,

$G_{k,j}$ Characteristic value of permanent load
$Q_{k,1}$ Characteristic value of leading variable load
$Q_{k,i}$ Characteristic value of other variable loads
ψ_i Factor for variable loads

3.3 Design assumptions

The focus of this thesis lies within the optimization routine, the core geometry chosen is therefore a web-core. This choice is made as the geometry is the most basic among the selection of core configurations and it is most favorable for parametrization. A segment of a plate with web-core is shown in Figure 3.2.

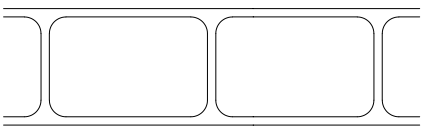


Figure 3.2: *Plate with web-core.*

The deck is oriented with the extrusions in the transverse direction of the span of the bridge. In this design configuration, the deck has a distinct orthotropic behavior where the stiffness along the extrusions is greater than that perpendicular to them. It is thereby assumed that the forces will rather directly distribute stresses along the extrusion direction to the supports, i.e. the main longitudinal load-carrying structure, see Figure 3.3. Stresses on the deck will thereby primarily come from global bending around an axis perpendicular to the extrusion direction and local load effects from the directly applied wheel.

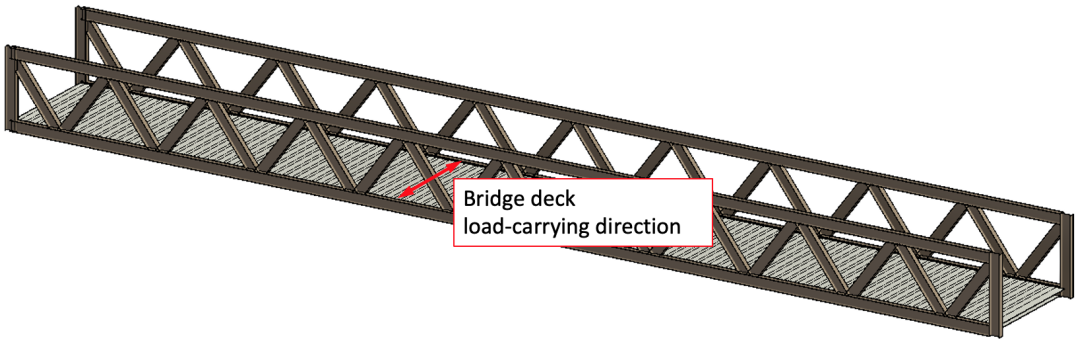


Figure 3.3: *The Figure illustrated the main load-carrying direction of the bridge deck.*

In an article by T. W. Siwowski (2009), the structural behavior of an aluminum plate with a triangular core was evaluated. The results showed that stresses at the top plate were 2-3 times greater than the stresses at the bottom plate and failure occurred due to yielding of the top plate under the wheel pressure. This indicates that combined load effects from global bending and the directly applied wheel pressure is governing the design. It is reasonable to believe that the same applies for orthotropic aluminium plates with web-core given that the span length and height of the deck is approximately the same.

Consequently, the focus in the optimization routine will be to evaluate stresses where the wheel pressure is applied as global and local load effects coincide in this area. Hence, the largest stresses are expected there. This implies an assumption that a cross-section optimized to withstand combined load effects in this area will also meet the other design requirements. Hence, the only design check that is included in the optimization routine is the condition that the stresses developed here are below the yield limit. After obtaining the optimized cross-section, the assumption needs to be confirmed. This is done in a later step where the remaining checks are performed.

Welds, and therefore strength loss is excluded from the optimization routine. Instead effects from the heat affected zone are assumed to be handled for the optimized cross-section by increasing the plate thickness locally to compensate for strength loss. Possible welding positions and additional material due to backing for welds are only estimated for the optimized cross-section to be used in the cost comparison. To summarize, the following design assumptions are made:

- The optimization routine is made for a plate with web-core.
- The design is assumed to be governed by stresses directly under the applied wheel pressure.
- Strength loss due to welding is excluded from the optimization routine.

3.4 Design verifications

In order to ensure the load bearing capacity of the bridge deck, the load effects should be less than the capacity of the structure. A number of design requirements have been identified, which are divided according to whether the check is carried out in ULS or SLS. The design verifications are also subdivided according to whether they are carried out inside the optimization routine or after for the optimized cross-section. The design verifications can be seen in Table 3.4.

Table 3.4: *Verification's in SLS and ULS.*

Limit state	**Verification**	
	During optimization	After optimization
ULS	Stress in maximum bending region	
		Stress in maximum shear region
		Instability
SLS		Global deflections
		Local deflection

For calculating the resistance either proof strength (f_0) or ultimate tensile strength (f_u) can be used. As only linear elastic analysis is performed, the characteristic value of the proof strength f_0 is used to ensure no yielding of the material.

For design verifications of aluminium structures the partial factor method is used (SS-EN 1999-1-1, 2007). The design resistance R_d for a cross-section is calculated with equation 3.5, where R_k is the characteristic value of resistance and γ_M the partial safety factor for material (SS-EN 1990, 2002; SS-EN 1999-1-1, 2007). In ULS, γ_M varies depending on whether it is resistance of a cross-section regardless of Cross-Section Class (CSC), instability of member, or a cross-section in tension. γ_{M1} is recommended for the first two cases and γ_{M2} for the latter. Values are presented below. In SLS all partial factors are set to 1.0 (SS-EN 1990, 2002).

$$R_d = \frac{R_k}{\gamma_M} \tag{3.5}$$

$$\gamma_{M1} = 1.1$$
$$\gamma_{M2} = 1.25$$

3.4.1 Stress in maximum bending region

As previously described in Section 3.3, the combined load effects from global bending and the directly applied wheel pressure is assumed to govern the design. Verification of stresses is thus the only check performed inside the optimization routine. To obtain maximum stresses, one axle-pair from the service vehicle is placed in the center of the bridge width and moved over several positions in the longitudinal direction. The vehicle is centrally placed in the transverse direction to include large global bending. Figure 3.4 shows bending moment and shear force distribution for a

centrally placed vehicle with wheel load P. The check is made for Von Mises stresses (Equation 3.6) according to the yield criterion in Eurocode (SS-EN 1999-1-1, 2007). Stresses are analyzed in the area under one of the locally applied loads in ULS.

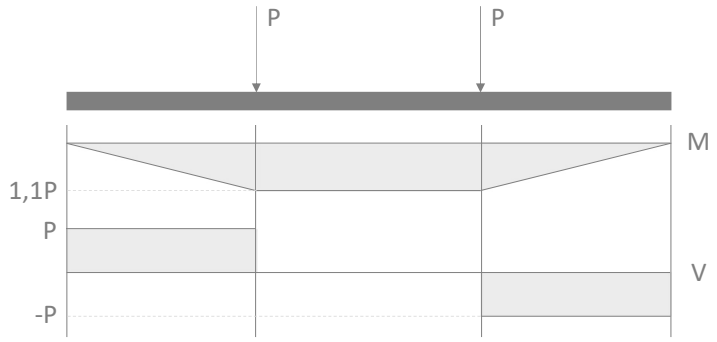


Figure 3.4: *Bending moment and shear force distribution for position when verifying stress due to maximum bending. The width of the bridge deck is 3.5m.*

$$\sqrt{\frac{1}{2}((\sigma_{xx,Ed}-\sigma_{yy,Ed})^2+(\sigma_{yy,Ed}-\sigma_{zz,Ed})^2+(\sigma_{zz,Ed}-\sigma_{xx,Ed})^2)+3(\tau_{xy,Ed}+\tau_{yz,Ed}+\tau_{zx,Ed})} \leq \frac{f_0}{\gamma_{M1}} \quad (3.6)$$

Where,

$\sigma_{xx,Ed}$	Stress at considered position, transverse extrusion
$\sigma_{yy,Ed}$	Stress at considered position, through-thickness
$\sigma_{zz,Ed}$	Stress at considered position, longitudinal to extrusion
$\tau_{xy,Ed}$	Shear stress at considered position, transverse extrusion-through thickness
$\tau_{yz,Ed}$	Shear stress at considered position, through thickness-longitudinal to extrusion
$\tau_{zx,Ed}$	Shear stress at considered position, longitudinal to extrusion-transverse extrusion

3.4.2 Stress in maximum shear region

To verify the shear capacity, the service vehicle is placed close to a support in the transverse direction of the bridge. Through this position large global shear is obtained close to the support as seen in Figure 3.5. The verification is made for Von Mises stresses (Equation 3.6) according to the yield criterion in Eurocode (SS-EN 1999-1-1, 2007). Checking the the von Mises stresses instead of the shear stresses is an conservative choice as the von Mises stresses contains contributions of both shear and normal stresses. The stresses are analysed in the area close to the supports with ULS load factors.

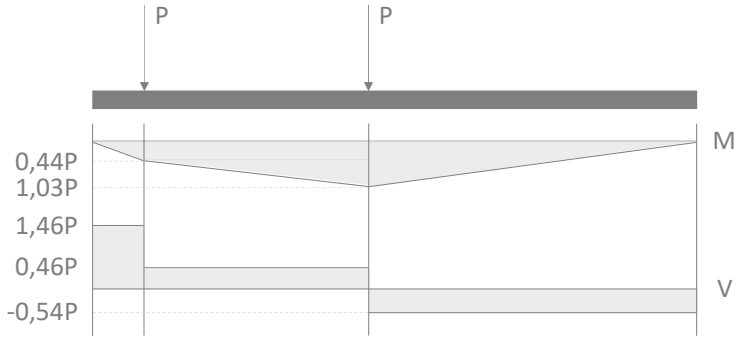


Figure 3.5: *Bending moment and shear force distribution for position when verifying stress due to maximum global transverse shear. The width of the bridge deck is 3.5m.*

3.4.3 Instability

To verify if local buckling can occur, a CSC is identified for all parts of the cross-section that fully are subjected to compression. See Table 3.5 for definitions of the four classes. The CSC is determined from the width-to-thickness ratio (β) of the part. Large values of the ratio imply a slender cross-section and a higher CSC. For sections in CSC 4 local buckling will reduce the capacity, while CSC 1 means that the section is not susceptible to local instablity. If the cross-section consists of several parts, the highest CSC identified will be used for all parts in compression. For calculation of (β), it is distinguished between internal and outstand parts according to SS-EN 1999-1-1 (2007). The limit parameters (β_1, β_2, β_3) for classification are in accordance with SS-EN 1999-1-1 (2007), where material, welds, and location of the part are taken into account. The only relevant buckling mode for the bridge deck to account for is the local buckling of the top plate. For the top plate, β is calculated with Equation 3.7.

$$\beta = b/t \tag{3.7}$$

Where,

b Width of face plate between web plates (radius excluded)
t Thickness of face plate

Table 3.5: *Definition of cross-section classes.*

Class	Definition
1	Ductile sections. Enough rotational capacity to form a plastic hinge with no resistance reduction.
2	Compact sections. Enough rotational capacity to develop plastic limit resistance.
3	Semi-compact sections. Enough rotational capacity to develop elastic limit resistance.
4	Slender sections. Local buckling will take place before the elastic limit stress is reached.

Since only linear analysis is performed in this thesis, it is only necessary to determine if the top plate of the cross-section is in class 4 or not. If the condition in Equation 3.8 is met, it implies that the cross-section is in class 1-3. The limit parameters β_1 are calculated with Equation 3.9 with the addition of Equation 3.10.

$$\beta \leq \beta_3 \tag{3.8}$$

$$\beta_3/\varepsilon = 18 \tag{3.9}$$

$$\varepsilon = \sqrt{250/f_0} \tag{3.10}$$

If the cross-section is in class 1-3, it also implies that the local buckling resistance is sufficient and that the local buckling factor p_c can be set to 1.0, see Equation 3.11.

$$\rho_c = 1.0 \quad if \quad \beta \leq \beta_3 \tag{3.11}$$

3.4.4 Local deflection

Local deflection is not a design verification according to either Eurocode or national standards. However, for the purpose of wearing surface durability, the local deflection of the face plate between support positions from the core plates should not be too large. As there are no limits regarding a maximum deflection a comparison to a steel orthotropic plate is made. In Eurocode (SS-EN 1993-2, 2006) Design of Steel Structures, recommendations are given for the design, including aspects of durability. For pedestrian bridges, the ratio between plate thickness and distance between stiffeners is limited to 40 (Equation 3.12), giving a maximum $cc - distance$ of 400 mm if the plate thickness (t) is 10 mm. The local deflection of a steel plate with these dimensions is therefore compared to the local deflection between web-cores for the optimized cross-section in aluminium.

$$\begin{aligned} t &\geq 10 \quad mm \\ cc/t &\leq 40 \end{aligned} \tag{3.12}$$

For calculation of the local deflection, the plate is simplified into a girder, continuous over 6 supports. The load is applied in the middle section. Both girders are assumed to have rectangular cross-sections with the same width (b), but different material properties, thickness, and *cc*-distance. The deflection of such a girder is calculated with Equation 3.13 (Avén, Lantz, & Lorentsen, 1983). Where k is a constant, Q is the point load, L is the span between supports, E is Young's modulus of elasticity, and I is the moment of inertia. The moment of inertia for a rectangular cross-section is calculated with Equation 3.14.

$$w = k \cdot \frac{Q \cdot L^3}{100 \cdot E \cdot I} \tag{3.13}$$

$$I = \frac{b \cdot h^3}{12} \tag{3.14}$$

The limitation condition for deflection can be formulated according to Equation 3.15 by simplifying Equation 3.13. The verification is made by modifying Equation 3.15 to calculate the maximum allowed cc-distance, where the only unknown variable is the resulting plate thickness, see Equation 3.16. If the condition in Equation 3.17 is fulfilled, local deflections are assumed to be small enough to assure the durability of the surface coating.

$$\frac{cc_A^3}{E_A \cdot t_{pu}^3} \leq \frac{cc_S^3}{E_S \cdot t_S^3} \tag{3.15}$$

$$cc_{A.max} = \frac{t_{pu} \cdot cc_S}{t_S} \cdot \sqrt[3]{\frac{E_A}{E_S}} \tag{3.16}$$

$$cc_A \leq cc_{A.max} \tag{3.17}$$

Where,

t_{pu}	Thickness of the aluminium plate
t_S	Thickness of the steel plate
cc_A	Distance between webs plates for the aluminium plate
cc_S	Distance between webs plates for the steel plate
E_A	Young's modulus of elasticity for aluminium
E_S	Young's modulus of elasticity for steel. (210GPa (SS-EN 1993-1-1, 2023)).

3.4.5 Global deflection

The vertical deflection is limited in accordance with the National Annex *Krav Brobyggande* (Trafikverket, 2019). Maximum deflection is calculated with equation 3.18, where L is the theoretical span length between supports. The deflection in the direction of the extrusions is checked in SLS and only for the uniformly distributed load presented in Section 3.2.

$$\delta \leq \frac{L}{400} \tag{3.18}$$

4
Structural model

In the following chapter, the structural model for the FE-model used in the optimization is presented. The model is based on material, loads, and design assumptions presented in Chapter 3. The aim of the model is to analyze stresses due to axle load from a service vehicle. The model is created in the FE-program ABAQUS and is later implemented in the optimization routine as a module.

4.1 Structural system

The bridge deck is oriented with the extrusions perpendicular to the bridge´s longitudinal direction. All load in the transverse direction is carried by the bridge deck, as this is the case, no additional transverse load carrying structure is used. The bridge deck is meant to carry the load to a main longitudinal load-carrying structure, exemplified by steel truss-girders in Figure 4.1. However, only a segment of the deck is modeled, representing the global response in the transverse direction. The deck is modeled to be simply supported as the connections are disregarded. Consequently, negative moments in the connection will be disregarded and no composite action between the members can be accounted for. In this configuration the deck has primarily two functions, to transfer loads to the main load-carrying system while resisting the locally applied load from wheel pressure.

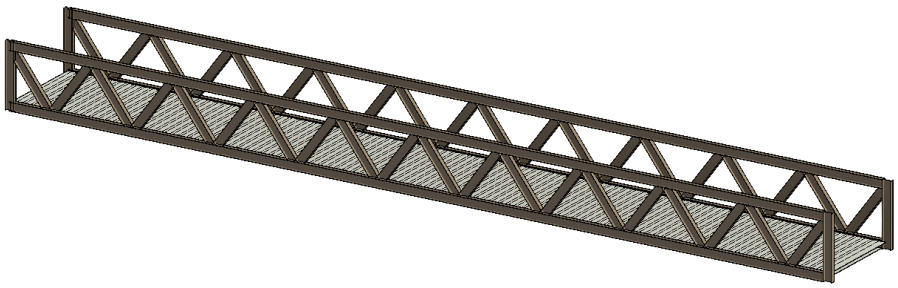


Figure 4.1: *Bridge deck supported on a longitudinal load-carrying structure.*

The three-dimensional coordinate system used for the model is shown in Figure 4.2. Where *X-direction* is perpendicular to the extrusions and the direction of traffic, also referred to as the length of the model. *Z-direction* is along the extrusions and in the transverse direction of the traffic, referred to as the width of the model. *Y-direction* is the vertical direction.

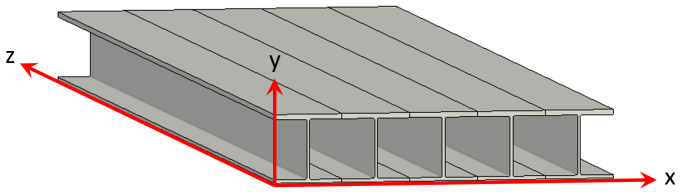


Figure 4.2: *Deck orientation.*

4.1.1 Model geometry

The width of the bridge is chosen to be 3,5 meters. According to data collected in a master thesis by Lindqvist and Svensson (2022) typical widths of the 57 studied pedestrian bridges in Sweden vary from 1.8 m up to 4.2 m. The width is a fixed parameter in the FE-model.

As a segment of the deck is modeled, the span length of the pedestrian bridge is not specified. The length of the segment is chosen to be as small as possible to keep the computational time short, while still accounting for the load distribution and not overestimating the stresses. The necessary segment length for accurate results is determined by a convergence study where the load distribution is evaluated for different segment lengths. The convergence study is presented in Section 4.3.

The model is made for a plate with hollow extrusions and a vertical web-core. The cross section is defined with the parameters in Figure 4.3. Dimensions are not specified as they are to be optimized, described in the following chapters.

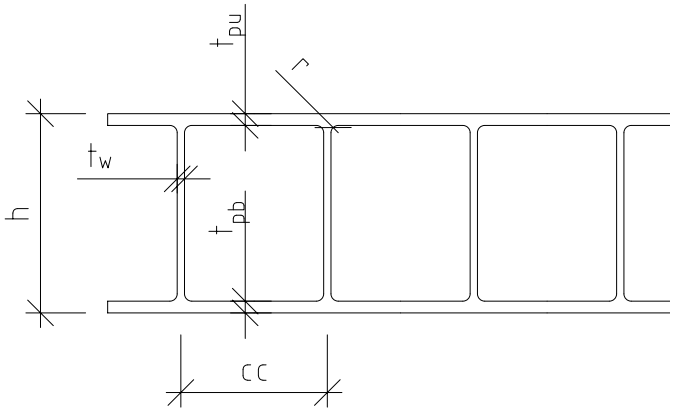


Figure 4.3: *Parameters used for the cross-section of the plate.*

4.2 FE-Model

The model was created in the FE-software ABAQUS, based on what is presented in Chapter 3. The aim of the model is to analyze stresses directly under the wheel load, as it is assumed to be governing the design. However, the global response needs to be captured and maximized as it affects the local stresses.

To analyze local stresses, a refined model made of solid elements is used for the detailed geometry of the cross-sections presented in Figure 4.3. To model the deck segment completely with solid elements would make the analysis time-consuming. To solve this, an alternative approach of sub-modelling is adopted. Instead of using a multi-level analysis approach for the design of the plate, where the result from the global model is used as input to the local model (Nilsson Strand, 2020). The analysis is performed in one step using both a local and a global model simultaneously.

The plate segment with outer dimensions according to Figure 4.5, are modeled using shell elements with a simplified cross-section neglecting the radius in the intersection of web-core and the top/bottom plate. Under one of the wheel pressures in the global shell model, a cut is made where the local solid model with detailed geometry is placed, illustrated in Figure 4.4. The models are placed such that the centerline of the top and bottom plate of the local solid model aligns with the top and bottom plate of the global shell model. The two models are connected using kinematic couplings between nodes at the surfaces/edges of the respective parts. The translations of the solid surface nodes are set to follow the translation of the adjacent shell nodes.

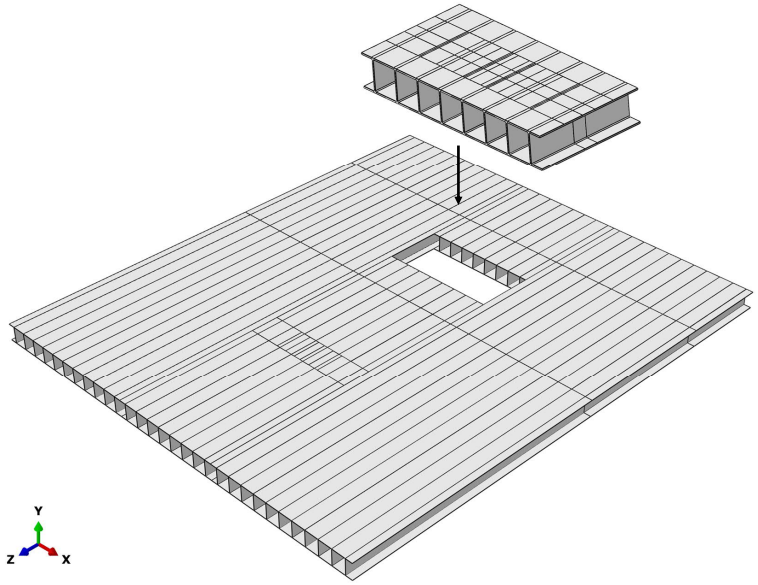


Figure 4.4: *Illustration of the sub-modeling approach used in the analysis.*

Figure 4.5 shows the geometric configuration of the model. The local solid model is centrally located in *x-direction*. In *z-direction* the solid model is placed such that

the wheel pressure can be added on top when the wheel axle is aligned with the center of the strip. The width of the solid model extends 100 mm outside the wheel pressure on each side, resulting in a fixed width of 400 mm. The length of the solid model depends on two factors, the cc-distance of the web-cores and the movement of the load in *x-direction*. The minimum length is set to 700 mm. The extension of the model is to avoid disturbance at the area of interest as false stresses can arise in the vicinity of couplings between the main and secondary nodes.

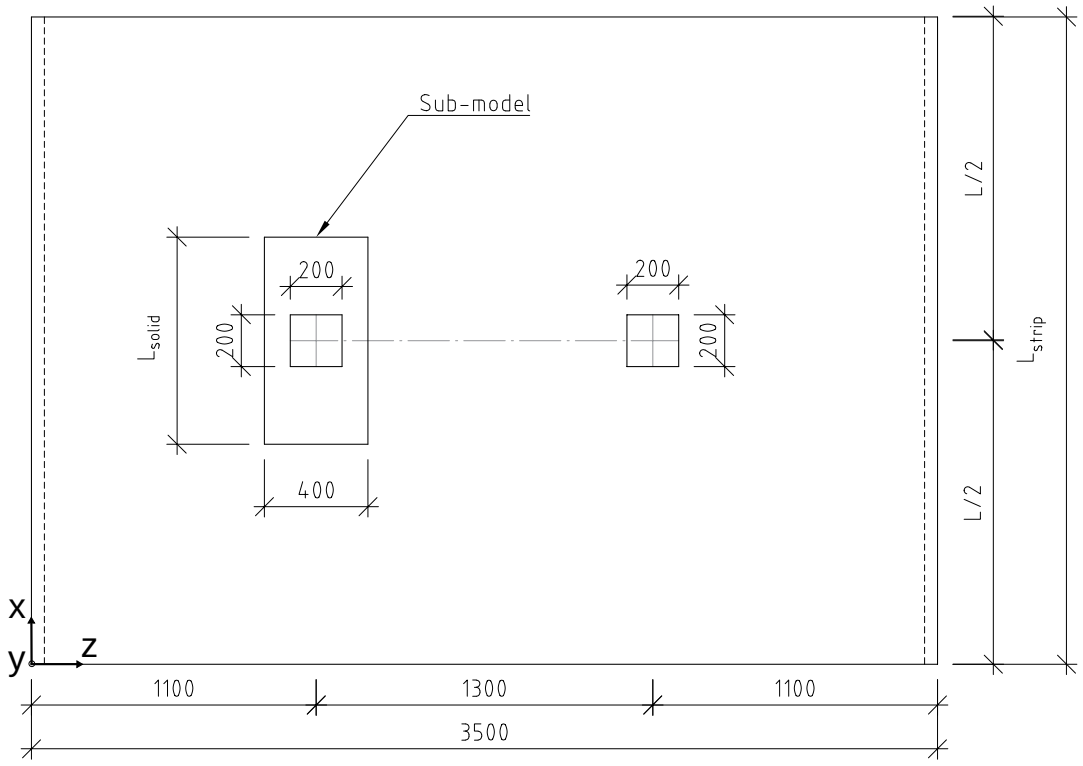


Figure 4.5: *FE-model geometry. Dashed lines at the edges symbolizes simply supported BCs.*

4.2.1 Area of interest

Stresses are analyzed in the region where local wheel pressure from the service vehicle coincides with maximum global moment. The results are extracted from the inside surface of the top plate, the inside surface of the bottom plate, and the inside surface of the cell under the applied load. These locations represent the area of interest and are highlighted in red in Figure 4.6.

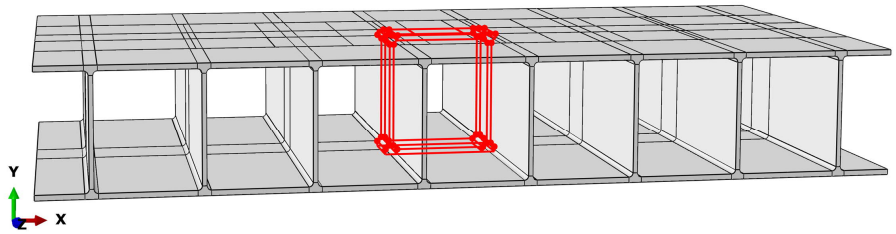


Figure 4.6: *Area of interest where largest stresses are expected, shown in red.*

4.2.2 Element types

The shell element type used in the analysis is *8-node doubly curved thick shell* with reduced integration. The solid model is made with element type *20-node quadratic brick* with reduced integration. Using reduced integration means that for a shell element, the stress is evaluated in 4 (2x2) integration points instead of 9 (3x3). For a solid element, the stress is evaluated in 8 (2x2x2) integration points instead of 27 (3x3x3).

4.2.3 Load application and boundary conditions

The load on the structure is based on load cases stated in Section 3 and Section 3.3. The self-weight of the bridge decks is applied as a gravity load, calculated with the density of aluminum and the gravitational constant. As the radius of the cross-section is disregarded for the shell model, the load is slightly underestimated. The surface coating of acrylate is applied as a surface pressure.

In order to find the most unfavorable stresses from local bending the service vehicle is moved in five load steps in *x-direction*. The first position is located with the edge of the wheel pressure centrally over one web plate. From there the load is moved in five equally long steps until the end of the wheel pressure is located centrally over the next web plate, see Figure 4.7. The load from the service vehicle is applied as a pressure on the contact area (200 mm x 200 mm).

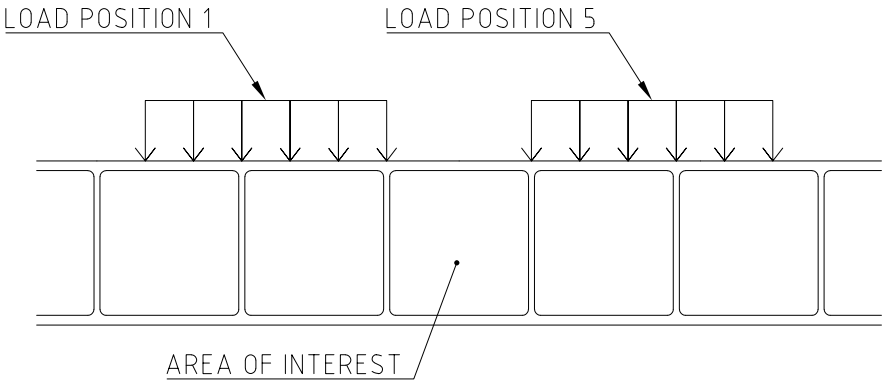


Figure 4.7: *First and last load position of the wheel pressure in x-direction. A total of 5 load positions are tested.*

The boundary conditions applied to the structure are set to resemble a simply supported deck on two edges. Translation in *y-direction* is prevented on all shell edge nodes on the web, top and bottom plate at each side of the deck strip in *z-direction*. To prevent rigid body motion, single nodes on the edges are restricted in *x-* and *z-direction*.

4.2.4 Mesh

The mesh is divided into different element sizes in different regions. The closer the mesh approaches the area of interest, the smaller element sizes are used. To be able to control this, each edge of both the global shell model and the local solid model is assigned a seed-size. For the shell model, seed-sizes in *x-direction* and *y-direction* are assigned to be consistent over the model. In *z-direction*, the model is divided into three regions where the aspect ratio is decreased in the area around the solid model, see Figure 4.8.

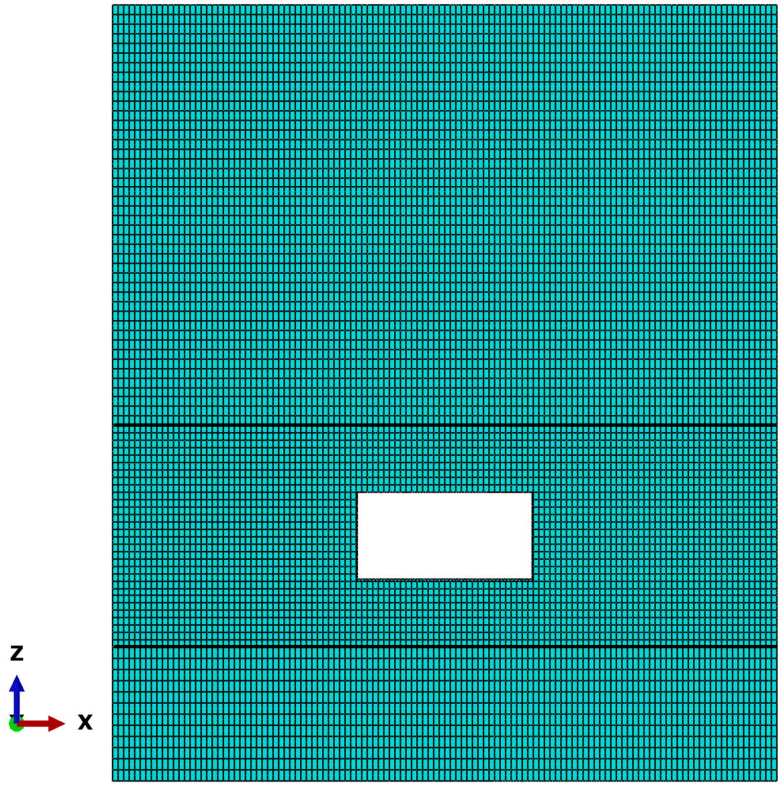


Figure 4.8: *Mesh of shell model. The seed-size in z-direction is divided into three regions where the aspect-ratio of the elements is decreased closer to the solid model.*

The solid model is divided into significantly more areas, where the number of elements is further increased in the regions of the applied load. In *z-direction* a strip in the middle of the model is meshed finer. This is also true in *x-* and *y-direction* for the cell where the wheel pressure is moved over. In Figure 4.9, the finer meshed areas can be seen. A coarser mesh is assigned to the other parts of the solid model. Seed-sizes for the different edges of the shell och solid model are determined with a convergence study, presented in section 4.3. The chosen seed-sizes can be seen in Appendix B.

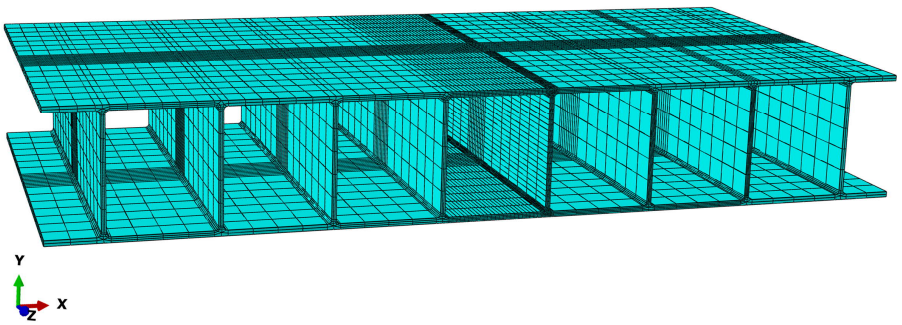


Figure 4.9: *Mesh of solid model. The number of elements is increased in the area of interest.*

4.3 Verification of model settings

Model settings regarding the length of the strip and mesh-size are verified through convergence studies. First, the strip length is decided by analyzing load distribution for different lengths. In a second convergence study, the mesh-setting was decided by analyzing the influence of seed-sizes. The convergence studies are made for a reference geometry of the cross-section, presented in Figure 4.10. Settings verified for the reference geometry are assumed to be valid for all combinations of dimensions generated later in the optimization.

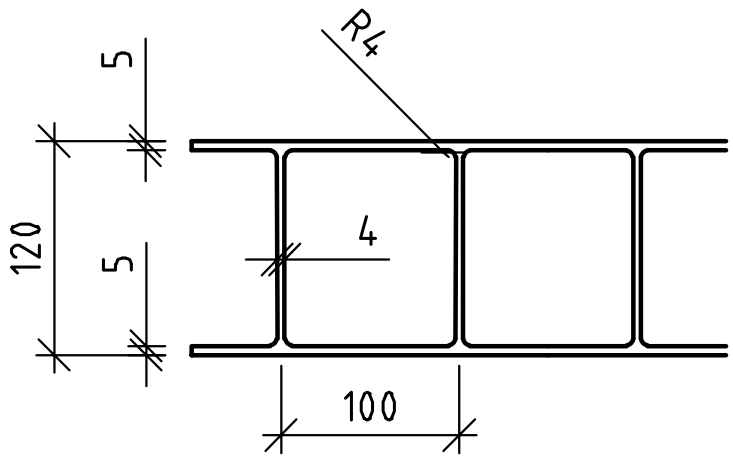


Figure 4.10: *Reference geometry used in the convergence studies, units in [mm].*

4.3.1 Strip length

The length of the strip is determined to take into account the load distribution along the span of the bridge. This is accomplished by evaluating how the stress changed in the area of interest when the length of the strip was changed. The strip length is obtained when a further increase in the length did not change the stresses significantly.

The change in stress is analyzed along the three paths shown in Figure 4.11. The first and the second path is defined centrally on the top and bottom surface of the solid model. The third path is defined on the inside surface of the cell where the wheel load is applied. One stress value was obtained at the edge and center nodes of each element along the paths. In this analysis, only load case 1 is used, see Figure 4.7. With this configuration, the load distribution is greater towards one edge of the strip.

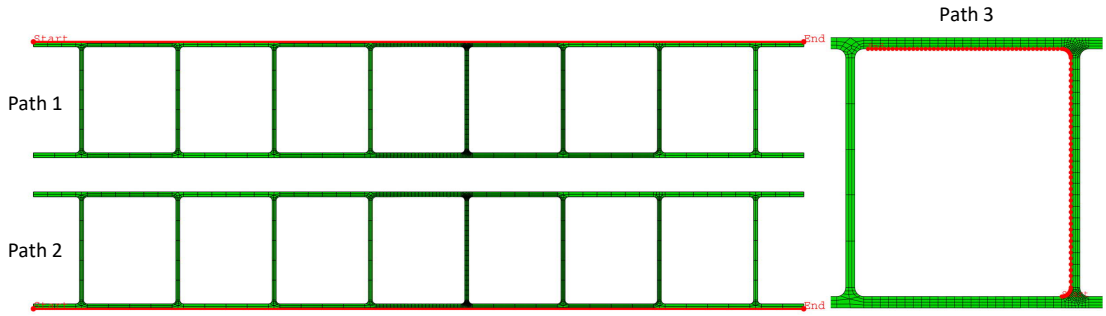


Figure 4.11: *Paths used in the convergence study to determine the strip length.*

Stresses over the solid model are analyzed for strip lengths of 1m, 2m, 3m, and 4m for the shell model. The stress distribution over the plate can be seen in Figure 4.12, for a strip with lengths 1m, 2m, and 3m, respectively. These results are visualized in graphs, one for each path, see Figure 4.13, 4.14 and 4.15. It can be seen how stresses decrease with increasing length, while differences in stresses become smaller. When differences between the steps becomes small enough or stops completely, a converged strip length is found. The results for lengths 1m, 2m, and 3m are compared to the results for length 4m. The quantitative comparison includes the maximum absolute change and percentage change. These results are presented in Table 4.1. The largest percentage difference of 3.3% between a 3m strip and a 4m strip is considered an acceptable error. Hence, a 3m long strip is used for the model.

Table 4.1: *Comparison of maximum absolute change and percentage change of the stress for strip length convergence.*

	Max. Absolute Difference			Max. Percentage Difference		
Strip Length	Path 1	Path 2	Path 3	Path 1	Path 2	Path 3
1m vs. 4m	44.4 MPa	44.0 MPa	31.6 MPa	114.7 %	112.6 %	56.8 %
2m vs. 4m	7.4 MPa	7.3 MPa	5.6 MPa	17.8 %	17.7 %	10.1 %
3m vs. 4m	1.6 MPa	1.6 MPa	1.3 MPa	3.2 %	3.3 %	2.1 %

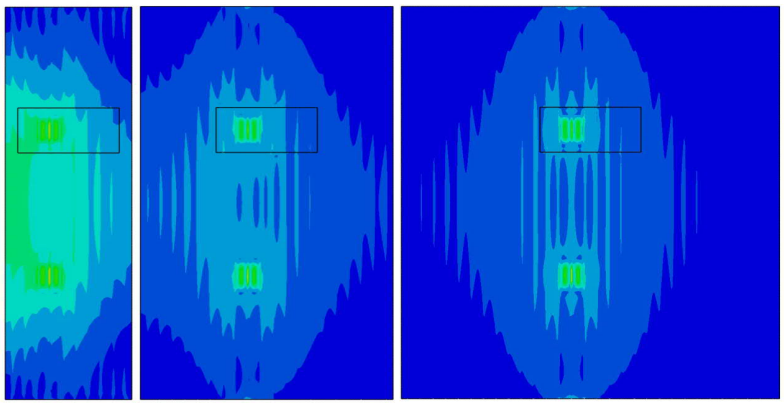


Figure 4.12: *Stress distribution for a 1m, 2m, and 3m long strip.*

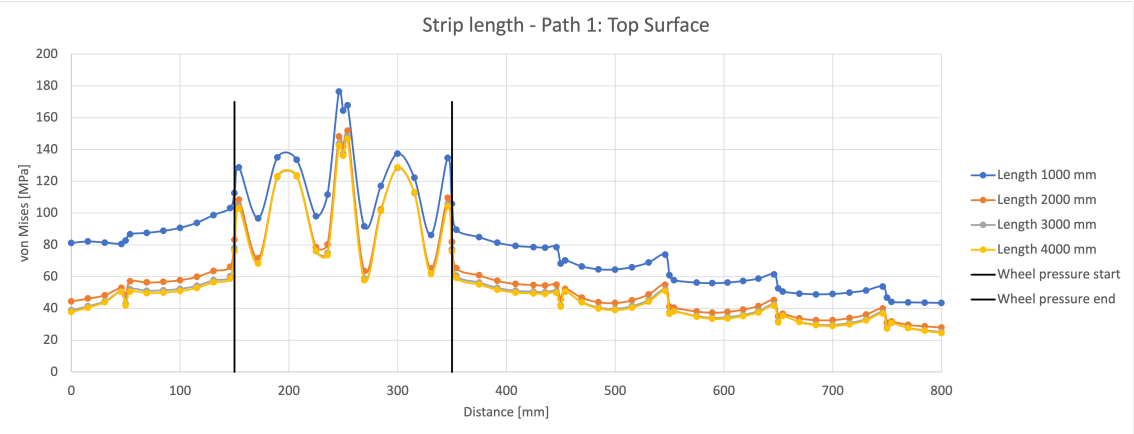


Figure 4.13: *Stress distribution along path 1 for a 1m, 2m, 3m, and 4m long strip. The two vertical black lines show where the wheel pressure is applied.*

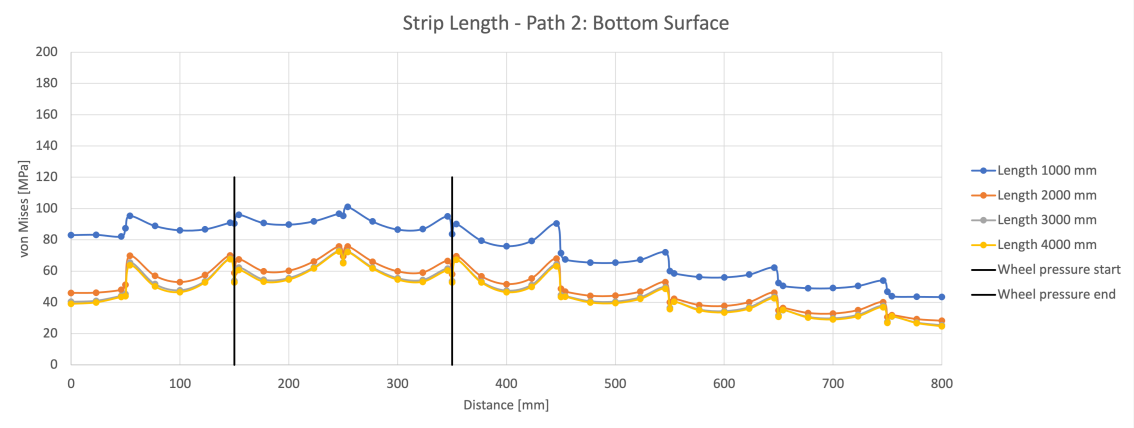


Figure 4.14: *Stress distribution along path 2 for a 1m, 2m, 3m, and 4m long strip. The two vertical black lines show where the wheel pressure is applied.*

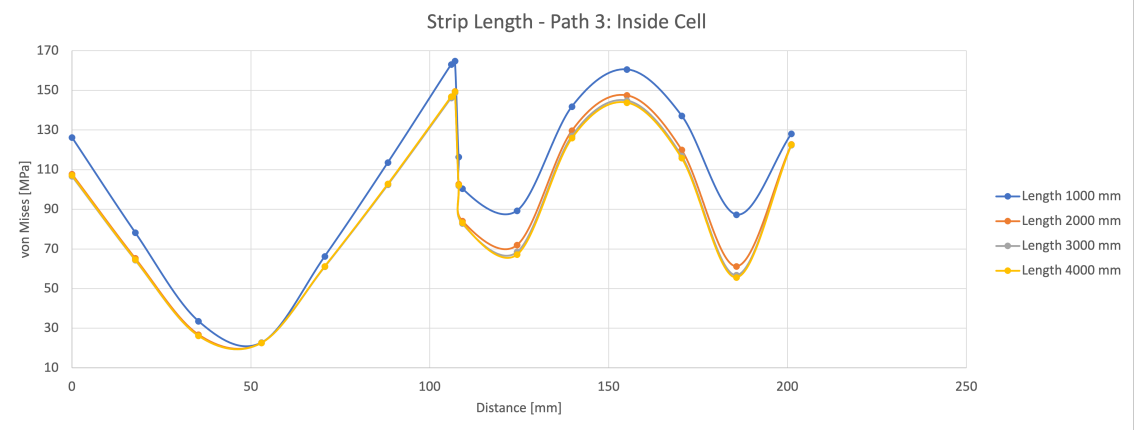


Figure 4.15: *Stress distribution along path 3 for a 1m, 2m, 3m, and 4m long strip.*

4.3.2 Mesh convergence

To determine the suitable mesh-size, stress distribution along the paths shown in Figure 4.16 is analyzed for different numbers of elements. The mesh is controlled by seeding of the edges and a decrease in seed-size means a finer mesh and more elements. A refinement of the mesh-size means that all seed-sizes in the specified mesh zone have been decreased. The analysis is made for load case 3. In load case 3 the wheel pressure is placed centrally between two web plates. It was also taken into account that a change in seed size changes the coordinates of the nodes. As the stresses are obtained at the nodes along the paths, only nodes located within a set tolerance from each other were compared quantitatively. For paths 1 and 2 the tolerance is set to 0.2mm, while for path 3 the tolerance is set to 0.1mm.

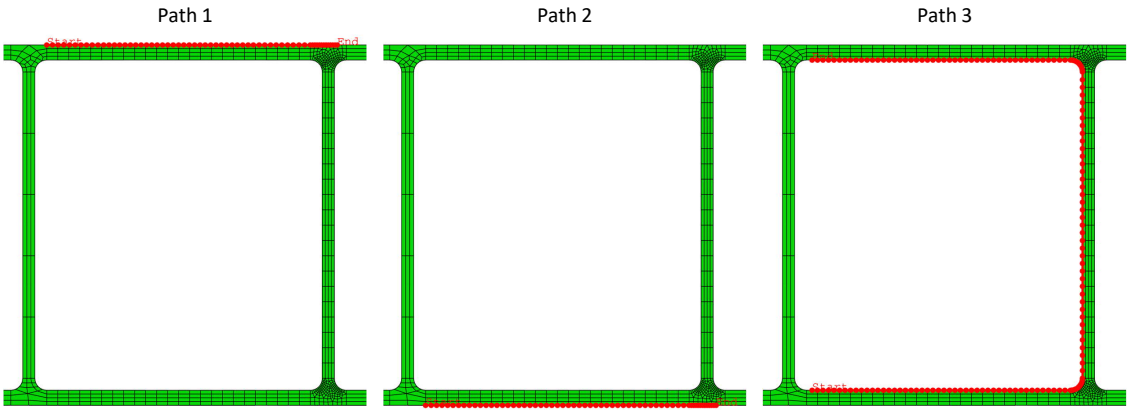


Figure 4.16: *Paths used in the convergence study to determine the mesh-size.*

The mesh-size is determined by step-wise refinement of the seed-size until the change in stress along the paths has converged to an acceptable degree. The refinement of the seed-sizes is done for one area at a time. The convergence study for the mesh settings is divided into the following three steps.

- Step 1: Determination of seed-size in the area of interest.
- Step 2: Determination of seed-size on remaining edges of the solid model, by increasing the number of elements and comparing with results from Step 1.
- Step 3: Determination of seed-size of the shell model, by increasing the number of elements and comparing with results from Step 1.

The mesh selected in Step 1 for further evaluation is assumed to have converged if the results from Steps 2 and 3 provide sufficiently small differences in stress. The number of elements used for mesh 1 to 5 are presented in Table 4.2. The first three meshes are used in Step 1 of the convergence study, Mesh 4 is used in Step 2, and Mesh 5 is used in Step 3.

Table 4.2: *Number of shell and solid elements used for the different mesh-size combinations.*

	Number of Shell elements	Number of Solid elements
Mesh 1	37 366	14 232
Mesh 2	37 366	25 320
Mesh 3	37 366	59 636
Mesh 4	37 366	67 222
Mesh 5	118 496	25 320

4.3.2.1 Step 1

The stress distribution for all nodes in Mesh 1, 2, and 3 along paths 1 and 2 are presented in graphs, see Figure 4.17 and 4.18. Figure 4.17 shows the resulting stresses on the top surface and Figure 4.18 shows the resulting stresses on the bottom surface. The seed-size on the shell model och outside of the area of interest on the solid model is kept constant. The resulting maximum absolute differences and percentage differences along paths 1 and 2 are presented in Table 4.3. The greatest difference is between Mesh 1 and Mesh 3, maximum percentage stress difference yields 1.7%. This was considered an acceptable error.

Table 4.3: *Maximum absolute difference and percentage difference of the stresses in paths 1 and 2 for Mesh 1 and 2 compared to Mesh 3.*

	Max. Absolute Difference		Max. Percentage Difference	
Compared Meshes	Path 1	Path 2	Path 1	Path 2
Mesh 1 and 3	3.0 MPa	1.0 MPa	1.7%	1.0%
Mesh 2 and 3	2.0 MPa	0.7 MPa	1.3%	0.7%

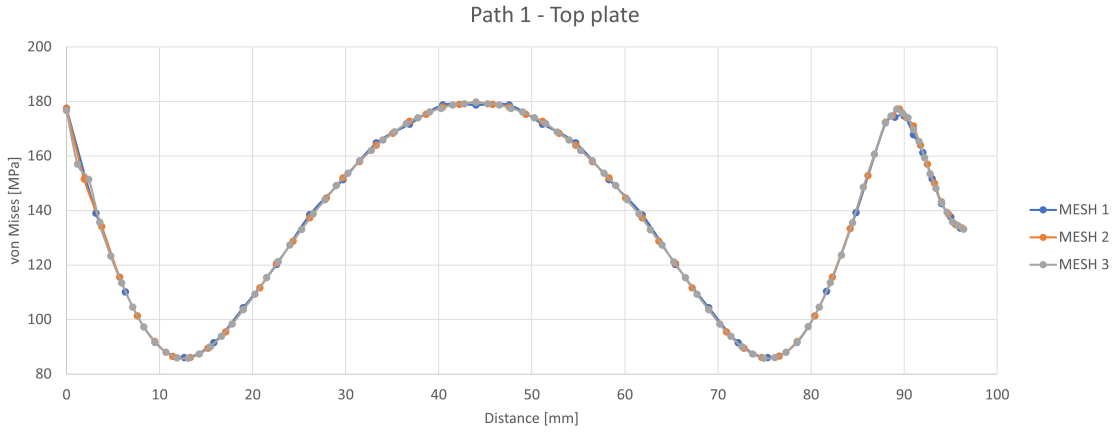


Figure 4.17: *Stress distribution along path 1 for Mesh 1, 2, and 3.*

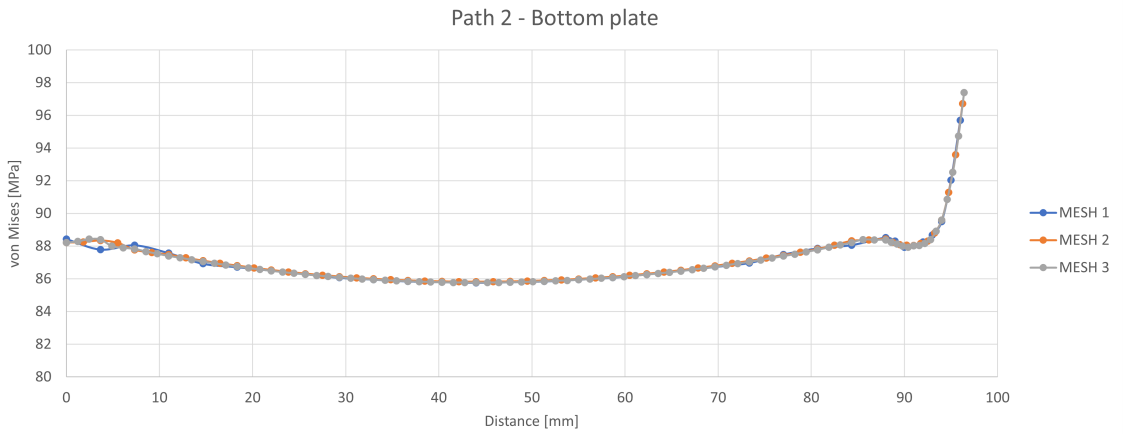


Figure 4.18: *Stress distribution along path 2 for Mesh 1, 2, and 3..*

The stress distribution for all nodes in Mesh 1, 2, and 3 along path 3 are presented in a graph, see Figure 4.19. Path 3 contains nodes on the curved surface of the two radiuses between the web plate and the top/bottom plate, hence the stress distribution is significantly affected by the mesh-size. Figure 4.20 and 4.21 shows a close-up of the stress distribution in the lower and upper radius respectively. The stress distribution for Mesh 2 and 3 are more similar, while Mesh 1 differs more.

The resulting maximum absolute difference and the percentage difference between the meshes are presented in Table 4.4. The greatest differences occur at the end of the top radius towards the top plate. As an increase in the number of elements leads to an increase in computational time, it was considered that the maximum differences of 5.3% was acceptable. Mesh 2 is evaluated further in Step 2 and Step 3 of the convergence study.

Table 4.4: *Maximum absolute difference and percentage difference of the stresses in path 3 for Mesh 1 and 2 compared to Mesh 3.*

Compared Meshes	Max. Absolute Difference Path 3	Max. Percentage Difference Path 3
Mesh 1 and 3	22.4 MPa	15.3%
Mesh 2 and 3	8.7 MPa	5.3%

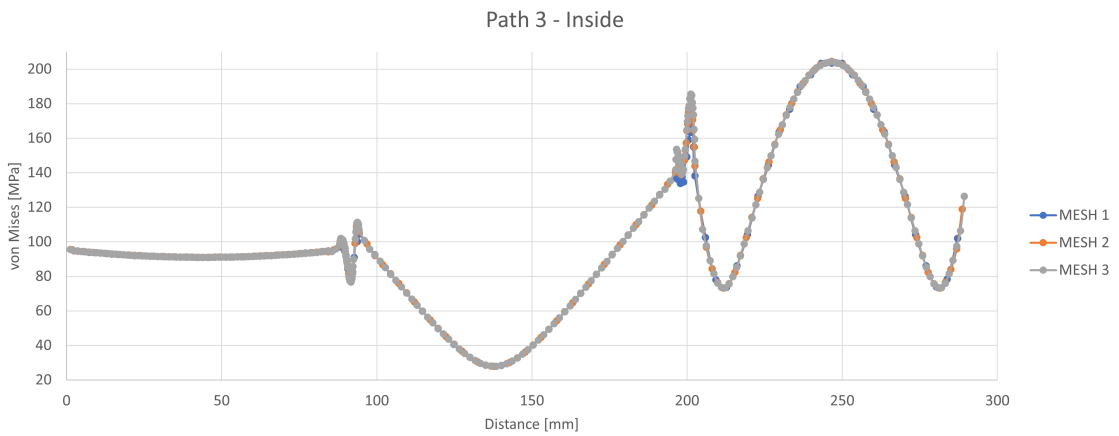


Figure 4.19: *Stress distribution along path 3 for Mesh 1, 2, and 3.*

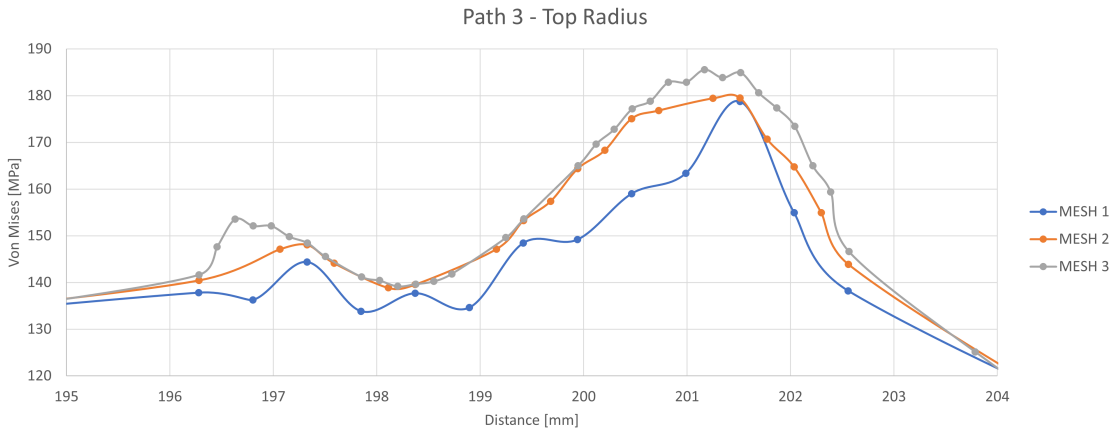


Figure 4.20: *Stress distribution in top radius of path 3 for Mesh 1, 2, and 3.*

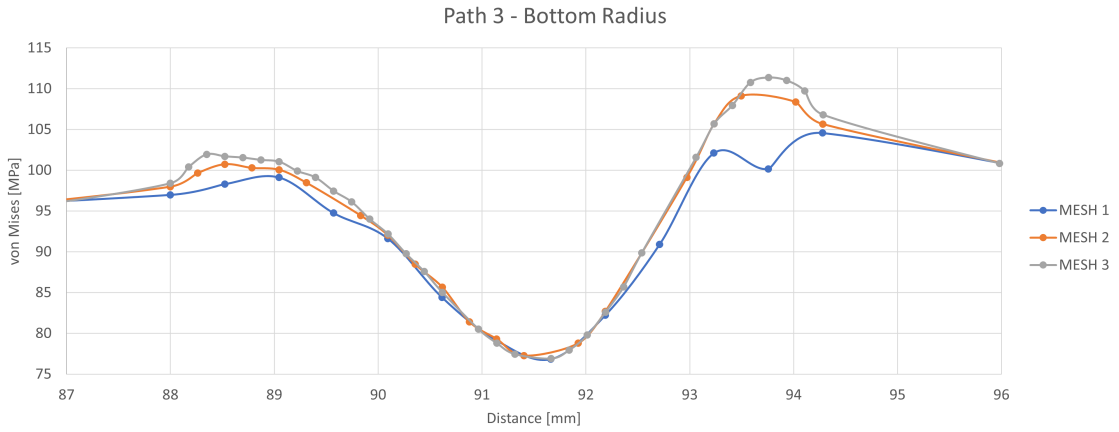


Figure 4.21: *Stress distribution in bottom radius of path 3 for Mesh 1, 2, and 3.*

4.3.2.2 Step 2

In the second step, the seed-sizes of edges outside of the area of interest are evaluated. This is done by keeping the seed-size in the area of interest constant and decreasing the seed-size of the remaining edges. The seed-sizes of the shell model are kept constant. The comparisons are made between Mesh 2 and Mesh 4. The resulting absolute difference and the percentage difference for paths 1, 2, and 3 are presented in Table 4.5. Changing to a finer mesh for the rest of the solid model had little effect on the area of interest. The maximum percentage change observed is 2.7%.

Table 4.5: *Maximum absolute difference and percentage difference of the stress in all paths for Mesh 2 compared to Mesh 4.*

	Max. Absolute Difference			Max. Percentage Difference		
Compared Meshes	Path 1	Path 2	Path 3	Path 1	Path 2	Path 3
Mesh 2 and 4	4.7 MPa	0.2 MPa	1.0 MPa	2.7%	0.3%	0.8%

4.3.2.3 Step 3

In the third and last convergence check the results from Mesh 2 are compared to the results from Mesh 5. The difference between Mesh 2 and Mesh 5 is that the seed-sizes for the shell model are decreased for Mesh 5. The resulting maximum absolute difference and the percentage difference for paths 1, 2, and 3 are presented in Table 4.6. The maximum percentage change observed was 2.0%, which is an acceptable error.

Table 4.6: *Maximum absolute difference and percentage difference of the stress in all paths for Mesh 2 compared to Mesh 5.*

	Max. Absolute Difference			Max. Percentage Difference		
Compared Meshes	Path 1	Path 2	Path 3	Path 1	Path 2	Path 3
Mesh 2 and 5	0.1 MPa	0.1 MPa	1.6 MPa	0.1%	0.1%	2.0%

The conclusion is made that the seed-sizes and the number of elements used for Mesh 2 yield acceptable results. Therefore, seed-sizes in Mesh 2 are used in the FE-model for the optimization routine. The seed-size settings for Mesh 2 are presented in Appendix B.

5
Optimization Procedure

The optimization procedure can be distinguished into two parts, GA and a FE-module, as can be seen in Figure 5.1. The parts interact with each other to create a closed loop that generates and evaluates solutions to the given problem. GA is implemented in the software MATLAB as a toolbox and the FE-module is started from MATLAB by calling a Python script. The python script generates an FE-simulation in the software ABAQUS, based on input from GA. The algorithm and the FE-module communicate by writing and reading text files that are continuously updated with present information. The FE-module is based on the structural model described in Chapter 4 and the implementation is further described in Section 5.1. The different parts of GA are presented in Section 5.2. The MATLAB and Python scripts are presented in Appendix C and D.

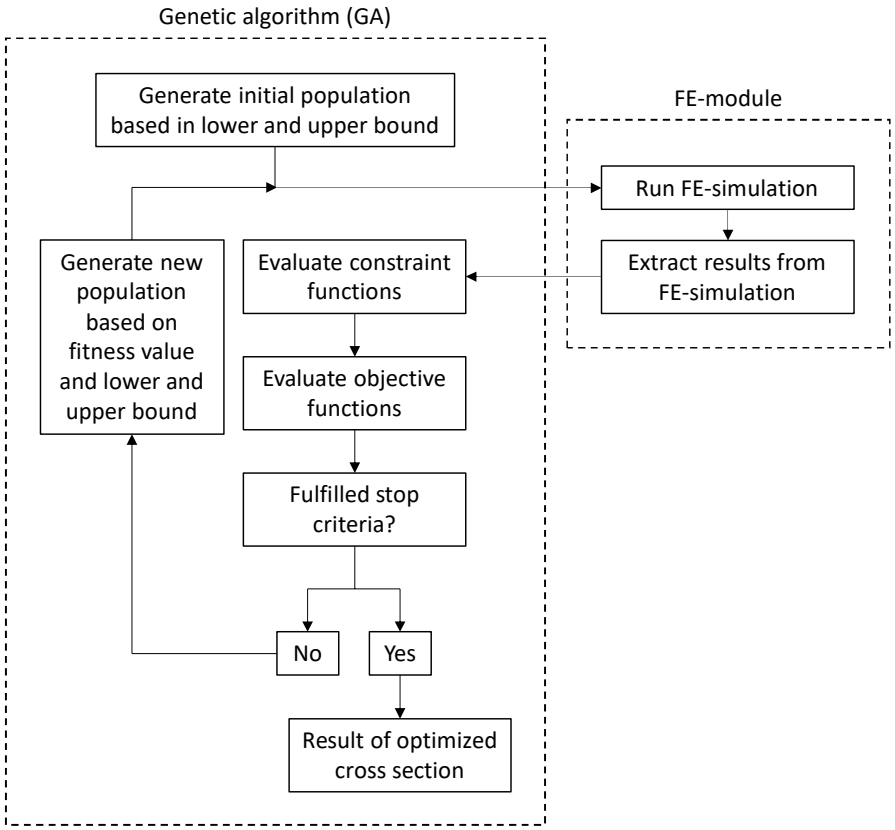


Figure 5.1: *An overview of the optimization procedure using GA and implementation of the FE-module.*

In this thesis, the optimization problem is defined as minimizing the use of material while meeting the required structural performance. This means that the cross-sectional area of the extruded hollow aluminum plate should be minimized while the maximum von Mises stress from the FE-simulation does not exceed the proof strength. The solution of the optimization consists of the final decision variables, representing the most suitable geometry of the cross-section to fulfill the objective. The decision variables are defined in Table 5.1.

Table 5.1: *Definition of decision variables.*

Variable	Definition
cc	Distance between web-cores
r	Radius
h	Total height of profile
t_{pu}	Thickness of upper face plate
t_w	Thickness of web-core
t_{pb}	Thickness of bottom face plate

5.1 FE-module

The FE-module implemented in the optimization is controlled by a Python script. This script is started from MATLAB with the command "abaqus cae noGUI". The command allows the script to run directly in the computer's Command Prompt without a graphical user interface. This is an alternative method to the ABAQUS CAE graphical user interface with a visualized pre and postprocessor. The commands defined in the script are thus sent via a Python interpreter directly to the ABAQUS CAE Kernel. This method enables the creation of an FE-model that can be reused with different inputs, it also saves computational effort which is essential in optimization.

Parameteric design is applied to enable the reuse of the script. It thereby allows free variables and is valid for all combinations of the decision variables. The convergence study shows that the length of the strip should be 3 meters, which corresponds to a minimum length. This is because the code is constructed so that single profile segments are joined together to create the entire plate. All the x- and y-coordinates belonging to the first shell and solid profile segment, respectively, are defined in vectors. The x-coordinates are continuously extended by a measure corresponding to one *cc-distance* at the time until the minimum length of the respective plate segments is reached. These vectors are used throughout the script to handle various settings applied to different parts of the FE-model, described in Chapter 4. The vectors used for the solid part can be seen below together with the numbering and position of the coordinates, see Figure 5.2.

$$x_{solid} = \begin{bmatrix} 0.0 \\ cc/2 - t_w/2 - r \\ cc/2 - t_w/2 \\ cc/2 - t_w/2 \\ cc/2 - t_w/2 - r \\ 0.0 \\ 0.0 \\ cc \\ cc \\ cc/2 + t_w/2 + r \\ cc/2 + t_w/2 \\ cc/2 + t_w/2 \\ cc/2 + t_w/2 + r \\ cc \\ cc \\ 0.0 \end{bmatrix} + cc \cdot n, \qquad y_{solid} = \begin{bmatrix} t_{pb} \\ t_{pb} \\ t_{pb} + r \\ h - t_{pu} - r \\ h - t_{pu} \\ h - t_{pu} \\ h \\ h \\ h - t_{pu} \\ h - t_{pu} \\ h - t_{pu} - r \\ t_{pb} + r \\ t_{pb} \\ t_{pb} \\ 0.0 \\ 0.0 \end{bmatrix}$$

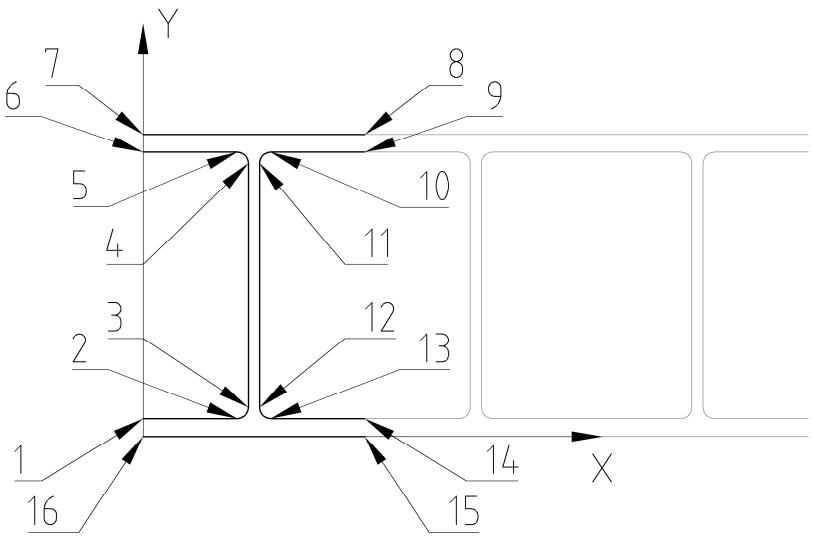


Figure 5.2: *Numbering and position of coordinates for a single profile segment of the solid part.*

5.2 Genetic algorithm settings

The Global Optimization toolbox includes a GA function named **ga**. The function is a solver-based setup, meaning that the algorithm is created to solve a pre-set problem. For **ga**, it is to find the minimum of a function.

When using the algorithm, there are a number of settings that can be used. These are called options and are controlled by the function **optimoptions**. The function is used to set population and reproduction options as well as stopping criteria.

5.2.1 Generate population

The optimization algorithm starts by creating an initial population. A population consists of a number of individuals. One individual is a possible solution to the problem and consists of a combination of the 6 decision variables. The decision variables are given as indexes in the possible range defined by the lower and upper bounds for each decision variable. The initial population is randomly generated within the limits of the lower and upper bound. This results in a different initial population for every new run of the program.

A new population is created with respect to the fitness value related to each individual and with some degree of randomization. However, still within the limits of the lower and upper bounds. For the creation of generations, the option *PopulationSize* is used. *PopulationSize* is the number of individuals in one population. A wide population will decrease the risk of finding a local minimum while a more narrow population is beneficial from a time perspective. Options regarding the creation of populations are listed below, where **nvars** is the number of decision variables for one individual. In this optimization problem, the number of decision variables is 6.

PopulationSize is the number of individuals in a population. The limit is set to $nvars \cdot 10$.

EliteCount, **MutationFcn**, **CrossoverFcn** and **CrossoverFraction** are all reproduction options. *EliteCount* is a number of the best-fitted individuals that will follow the next population. *CrossoverFcn* and *CrossoverFraction* are options that describe how new individuals are created, while *MutationFcn* contribute by adding some degree of randomization to the algorithm. For the reproduction options default settings are used.

5.2.2 Lower and upper bound

The lower and upper bounds define the possible range for each decision variable. The intervals are defined as discrete variables consisting only of integers. The optimization can also be performed with continuous variables, but to save computational time, the number of possible combinations is kept low.

The lower bound limits for thicknesses and radius are based on constraints from the extrusion process and to provide for weldability. While the upper bound limits are based on the dimensions of existing aluminium plate profiles. The limits for cc-distance between web-cores and height are also based on existing profiles, although with the freedom to vary in a larger range. The upper limit of the height is limited by the geometry of the extrusion press. The limit has been set at 200mm, giving a maximum extrusion width of around 400-500mm. A wide extrusion geometry is preferred as it means fewer welds but also fewer extrusions. The smaller cross-section geometries of the profile allow all integers in the domain between the lower and upper bound and the larger geometries allow every fifth or tenth integer in the domain.

The domain of the respective decision variable is presented in Table 5.2. However, due to production limitations, t_w depends on the height of the profile. For a height up to 100mm, the thickness should be 4mm and for a height of 200mm, the thickness should be 6mm. This will be limited in the constraint function.

Table 5.2: *Domain of decision variables, units in [mm].*

Variable	Domain
cc	50, 55 ... 200
r	1, 2 ... 10
h	50, 60 ... 200
t_{pu}	3, 4 ... 8
t_w	4, 5 ... 8
t_{pb}	3, 4 ... 8

5.2.3 Constraint function

Two inequality constraints are defined for the optimization problem. The constraints are constructed such that they should be less than or equal to zero. If the value is close to zero, it implies a high Utilization Rate (UR).

$$\begin{aligned} &Constraint\ 1 \quad \sigma_{mises.max} - f_{0d} \leq 0 \\ &Constraint\ 2 \quad t_{w.min} - t_w \leq 0 \end{aligned}$$

Constraint 1 relates to yield capacity. The maximum von Mises stress must be smaller or equal to the proof strength. Independently of the thickness, the lower of the proof strengths in Table 3.1 is used, which is a conservative choice.

Constraint 2 is set to ensure the production limitation on the ratio of web thickness and height. If the algorithm has generated a thickness smaller than $t_{w.min}$, the individual is not approved. The minimum value of the web-core thickness in relation to total height is presented in Table 5.3.

Table 5.3: *Minimum value of web-core thickness, units in [mm].*

h	$t_{w.min}$
$h \leq 100$	4
$100 < h \leq 150$	5
$150 < h$	6

5.2.4 Objective function

The optimization problem in this thesis has a single objective, which is to minimize the cross-sectional area and thus the weight of the aluminium plate. The objective function is expressed as the total area of the cross-section per meter of the total length of the strip. If the solution fulfills the constraints, the area is calculated as shown below.

$$\frac{Area * n}{cc * n} = \frac{cc * t_{pu} + cc * t_{pb} + (h - t_{pu} - t_{pb}) * t_w + ((2r)^2 - \pi r^2)}{cc} \quad [m^2/m]$$

5.2.5 Stopping criteria

For the optimization algorithm to stop, a number of options can be used. These are called stopping criteria. The user has to define conditions for when the solution has converged, this is made with *MaxStallGenerations* and *FunctionTolerance*. There are also criteria that stop the algorithm without a converged solution, this can either be achieved with time limitations or by limiting the number of iterations the algorithm can run. Listed below are all the options that can be used for the algorithm to terminate.

MaxGenerations is the maximum number of populations the algorithm will generate and run before it stops, even if no solution is found. This limit is set to $nvars \cdot 50$.

MaxTime is the maximum time in seconds the algorithm will run before stopping. No limit is set, the default value of *infinity* is used.

FitnessLimit is a lower boundary for the fitness value, if the limit is reached the algorithm stops. No limit is set, the default value of infinitely small is used.

MaxStallGenerations. If the average relative change over the past number of MaxStallGenerations compared to the best fitness value exceeds FunctionTolerance, the algorithm stops. The limit is set to $nvars \cdot 10/3$.

MaxStallTime. If the fitness value does not improve over the time MaxStallTime in seconds, the algorithm stops. No limit is set, the default value of *infinity* is used.

FunctionTolerance. If the average relative change over the past number of MaxStallGenerations compared to the best fitness value exceeds FunctionTolerance, the algorithm stops. The limit is set to $1 \cdot 10^{-6}$.

5.3 Results of optimization

In the following section results from the optimization algorithm are presented. The optimization routine has been executed twice with the set-up described in the previous part of this section. The cross-sectional geometry for optimization 1 can be seen in Figure 5.3, visualized to scale and resulting dimensions for both optimizations are presented in Table 5.4.

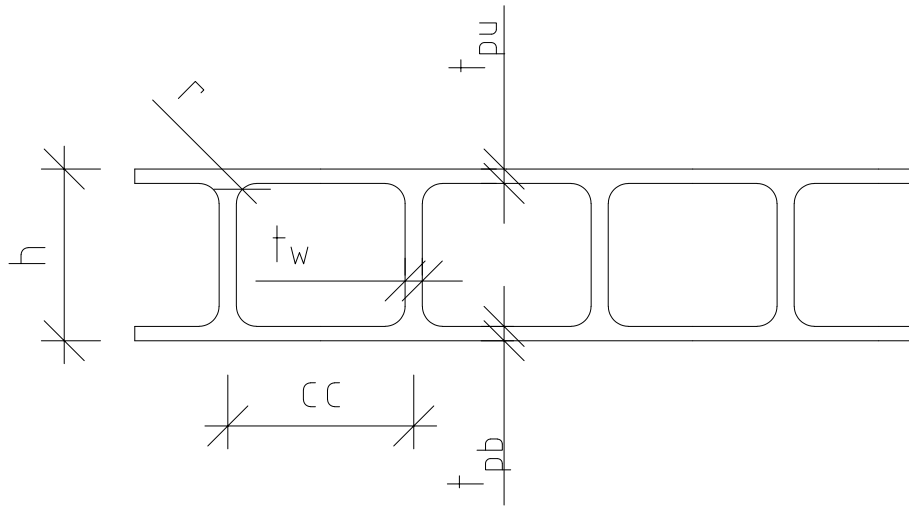


Figure 5.3: *Resulting cross-section from optimization 1 visualized to scale.*

Table 5.4: *Resulting cross-sectional dimensions from the optimizations.*

Variable	**Dimensions [mm]**	
	Optimization 1	Optimization 2
cc	65	55
r	7	8
h	60	50
t_{pu}	5	5
t_w	6	6
t_{pb}	5	5

The evolution of the optimization process for optimization 1 can be followed in Figure 5.4, showing scattered points representing the best area and the mean area from each generation. Each generation contains *60* individuals unless a new best solution is found, then a new generation starts over. In optimization 1 a total of *40* generations and *2345* individuals are created and evaluated and in optimization 2 the number of generations is *44* and *2573* individuals. The criteria that made the optimization algorithm stop in both cases was *MaxStallGenerations* and *FunctionTolerance*, meaning that the average relative change of fitness value over the last 20 generations is smaller than $1 \cdot 10^{-6}$.

The improvement of the minimum area and related geometries can also be read from Figure 5.4 for optimization. The final minimized areas can be seen in Table 5.5.

Table 5.5: *Resulting areas from the optimizations.*

Area [mm^2/mm]	
Optimization 1	15.3
Optimization 2	15.4

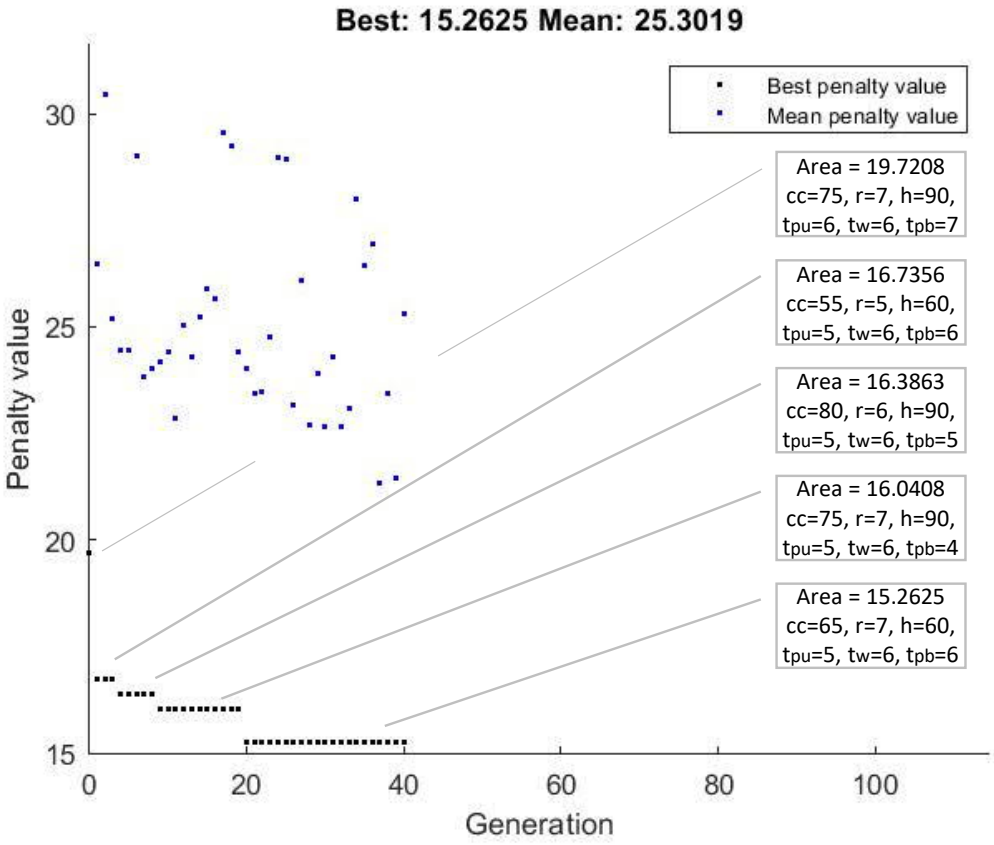


Figure 5.4: *The scattered points show the best area and mean area from each generation. Number of generations on the x-axis and area [mm^2/mm] on the y-axis.*

Maximum von Mises stresses from the two optimizations are presented in Table 5.6. Maximum stress appears on the inside surface of the radius in the area of interest.

Table 5.6: *Resulting maximum von Mises stresses from the optimizations.*

Max von Mises stress [MPa]	
Optimization 1	180
Optimization 2	177

The optimization was made with regard to two constraints. *Constraint 1* restricted von Mises stresses to not exceed the proof strength and *Constraint 2* restricted the

web plate thickness with regard to production limitations. A resulting value close to zero for the constraints is preferable as it implies a high UR. Both constraints are fulfilled for the resulting cross-sections. Only the height resulting from optimization 2 touches the lower bound. Neither of the dimensions in the cross-sections touches the upper bound. The resulting values from the constraint functions are presented in Table 5.7.

Table 5.7: *Resulting constraint values from the optimizations.*

	Constraint 1	**Constraint 2**
Constraint	$\sigma_{mises.max} - f_{0d}$	$t_{w.min} - t_w$
Optimization 1	-1.4	-2
Optimization 2	-4.7	-2

5.4 Structural response of optimized cross-section

In the following section results related to the final geometry of optimization 1 are presented and the structural response is analysed. The results from the two optimizations are close but not the same. The cross-section resulting from optimization 1 has a higher UR, a smaller area and both the height and the cc-distance are greater. All the plate thicknesses are the same. An FE-analysis have been made for the first cross-sectional geometry to look further into the deck behavior. The FE-analysis is based on the FE-model described in Chapter 4 Structural model.

Stresses in both the optimization routine and the FE-analysis have been extracted along the three paths presented in Figure 5.5. The paths are located on the surfaces of the area of interest described in Section 4.2.1. The stress variation along the paths is extracted for load cases 1 to 5, where the wheel pressure from the service vehicle is moved in the transverse direction of the extrusions. The results are presented in graphs, see Figure 5.6, 5.7 and 5.8.

It can clearly be seen that maximum stress is obtained in the top radius of Path 3 for load case 5. From the graphs, it can also be observed stresses at the bottom surface (Path 2) reaches higher stress values compared to the top surface (Path 1). Maximum stresses in the bottom are located close to the locations of the webs, i.e. around the ends of the plot. However, the stresses in the bottom plate field are generally lower compared to stresses in the top plate field for load cases 2 to 4.

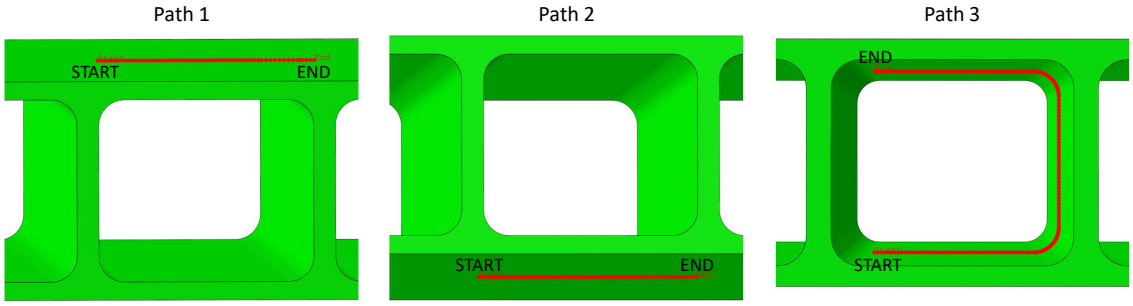


Figure 5.5: *Paths used for extracting von Mises stresses on the solid model.*

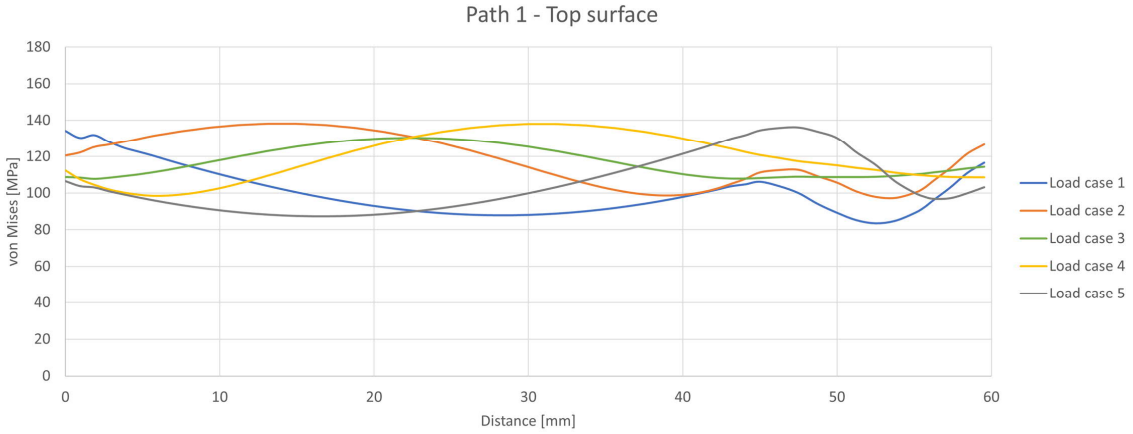


Figure 5.6: *Stress distribution along path 1 for load case 1-5. Distance [mm] on the x-axis and von Mises stresses [MPa] on the y-axis.*

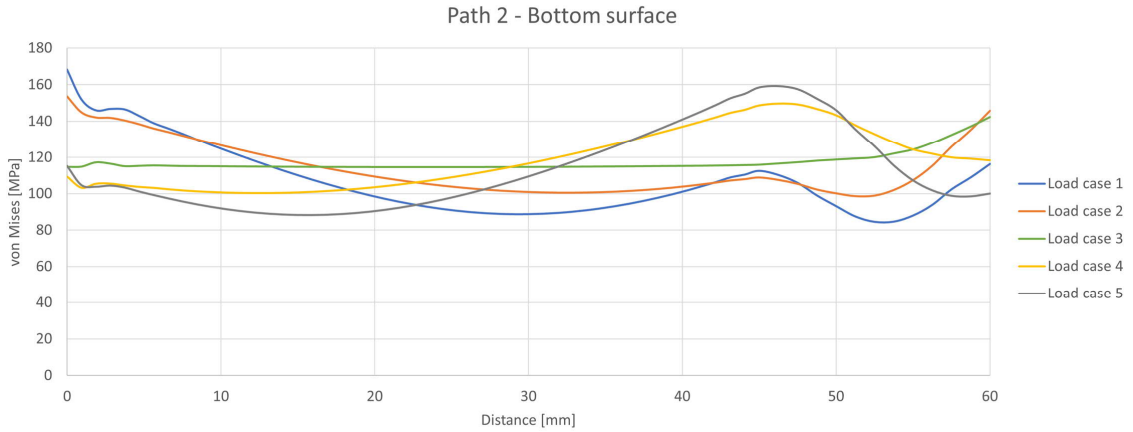


Figure 5.7: *Stress distribution along path 2 for load case 1-5. Distance [mm] on the x-axis and von Mises stresses [MPa] on the y-axis.*

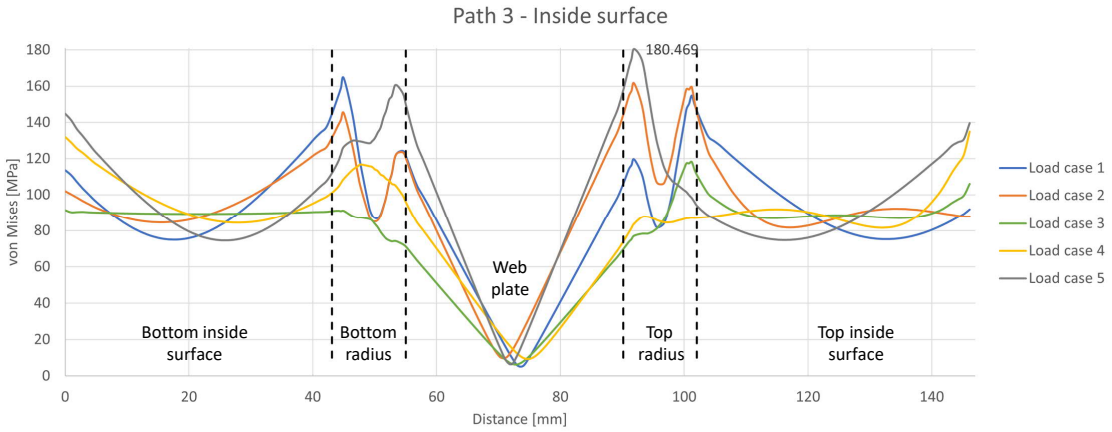


Figure 5.8: *Stress distribution along path 3 for load case 1-5. Distance [mm] on the x-axis and von Mises stresses [MPa] on the y-axis.*

The maximum von Mises stress from Paths 1 to 3 for each of the five load cases are presented in Table 5.8. The maximum stress is obtained along Path 3 for load case 5. This stress is the limiting value for the cross-sectional geometry. Maximum stress along Path 1 is obtained for load cases 2 and 3. This is due to the loading symmetry within the span of the cell. For Path 2, maximum stress is obtained for load case 1. No symmetry can be seen in the edges for Paths 1 and 2 since the paths on the top and bottom surfaces are shifted slightly to the right in relation to the midpoint of the cell of interest.

Table 5.8: *Maximum von Mises stresses in the position of large global bending.*

Load case	**Von Mises stress [MPa]**		
	Path 1	Path 2	Path 3
1	134	**168**	164
2	**138**	154	162
3	130	142	118
4	**138**	150	135
5	136	159	**180**

The stress distribution over the solid model in the position of load case 5 can be seen in Figure 5.9 and 5.10. Figure 5.9 indicates that the maximum stress appears on the inside radius of the unloaded cell of interest, adjacent to the loaded cell. Figure 5.10 shows a cut-through in the middle of the solid profile in the x-direction, indicating that the maximum stresses are located on the surface of the profile.

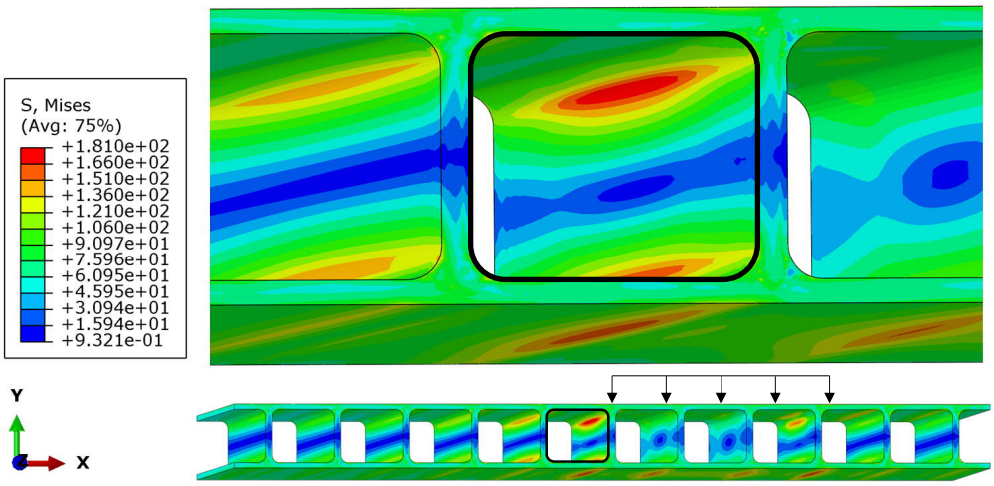


Figure 5.9: *Von Mises stress distribution in the position of large global bending for load case 5, units in [MPa]. Area of interest outlined in black.*

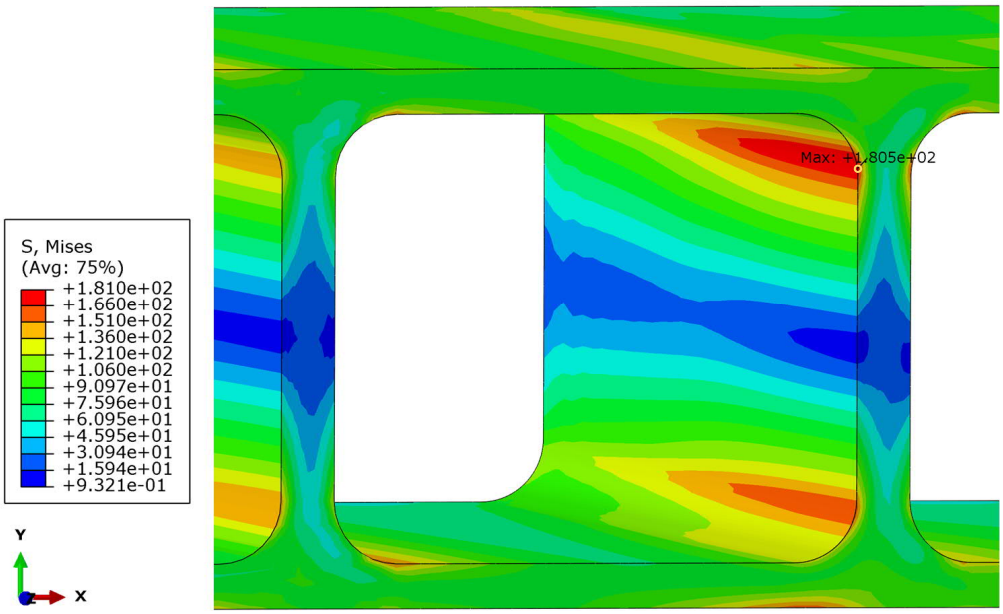


Figure 5.10: *Intersection of the cross section shown in Figure 5.9, presenting stress distribution inside the profile. Units in [MPa].*

In Figure 5.11, the stress distribution over the deck segment is presented. The representative deck segment resulted in a 3.055m long strip. The stress distribution in the weak direction i.e. transverse to the extrusions, shows that stresses are not only carried directly to the support but also propagate in the longitudinal direction of the bridge. This indicates that the relation between the stiffnesses in weak directions has a significant impact. Thus, the stocky cross-sections obtained from the optimizations behaves less orthotropic.

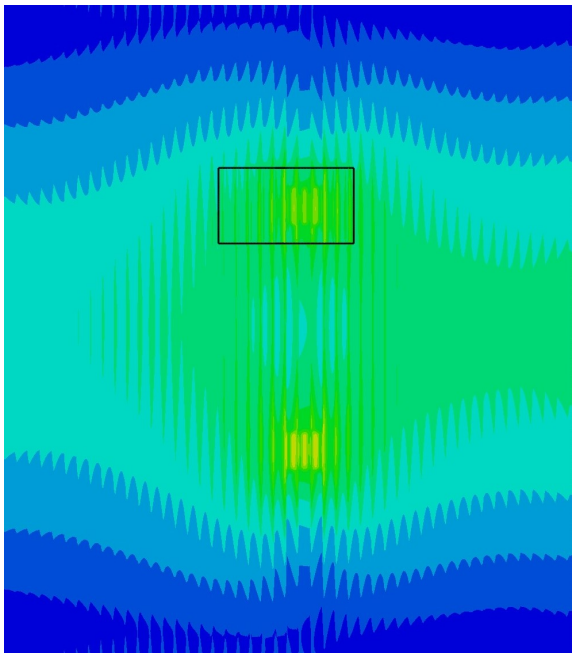


Figure 5.11: *Stress distribution of von Mises stresses in top surface from load case 5 for the optimized bridge deck with a width of 3.055m.*

In Figure 5.12, the distribution of normal stress in x-direction (σ_{xx}) is shown in a deformed shape for load case 5. Deformation scale factor 30 is applied to exaggerate the shape. It can be observed that shear-induced deformation occurs locally in the weak direction i.e. transverse to the extrusion direction. The cells are displaced in relation to each other in y-direction. The webs are also almost vertical in deformed shape.

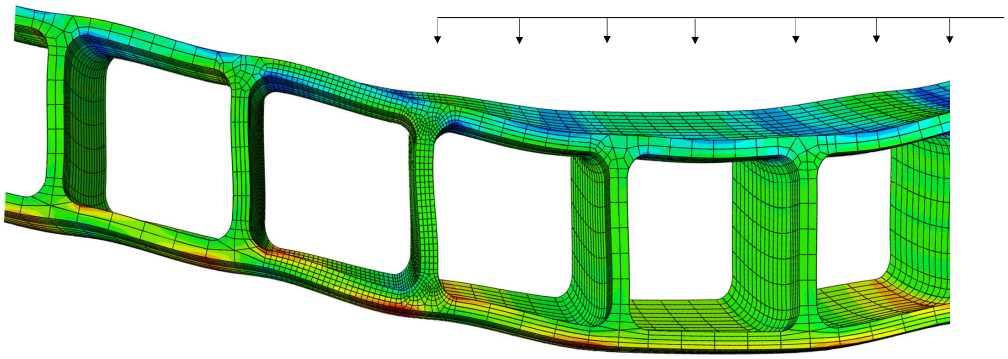


Figure 5.12: *Distribution of normal stress in x-direction (σ_{xx}) for load case 5 with deformed shape. Deformation scale factor 30.*

In the deformed shape of the panel, two effects from loading can be distinguished, shear-induced deformations due to relative displacement between loaded and unloaded cells and local bending under the directly applied wheel pressure on the top plate. These two effects are schematically illustrated in Figure 5.13. As seen in

Figure 5.12, the deformation of the bottom plate is dominated by shear-induced deformations according to the bottom illustration in Figure 5.13. The top plate, on the other hand, does not have the same defined shear-induced deformations. The top plate is affected by a combination of shear-induced deformation and local bending under the directly applied wheel pressure. If the local bending could be isolated, it would deform as illustrated schematically in the top of Figure 5.13 and if the shear-induced deformation could be isolated, it would deform as for the bottom plate. As seen in Figure 5.13 the deformation on the unloaded cell from the two different effects acts in the opposite direction from each other. Thus, the moments counteract each other. This explains why the von Mises stresses presented in 5.8 is larger in the bottom plate compared to the top plate for load case 5.

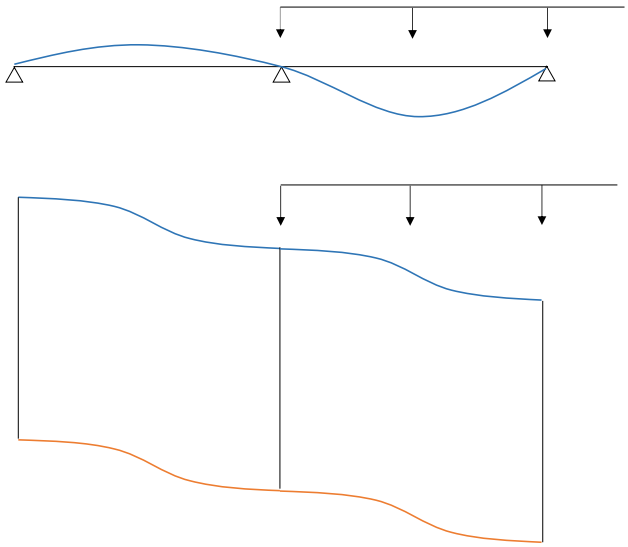


Figure 5.13: *Two principles for the deformation behavior. Effect of directly applied load (top). Effect of shearing of the cells (bottom)*

The normal stress distribution in x-direction (σ_{xx}) along the path on the top surface shown in Figure 5.14 are presented in Figure 5.15. The part of the path located above the web plate including the two radiuses is in Figure 5.15 defined by two vertical black lines. Stresses from load case 5 are used to demonstrate the behavior between loaded and unloaded cells. The left cell of the path is unloaded and the right is loaded. As indicated in Figure 5.15, the stress has a linear distribution on the surface of the unloaded cell and a parabolic distribution on the loaded cell. The shape of the curves reflects the moment diagrams of the loaded and unloaded cell causing the deformation that is illustratively shown in Figure 5.13.

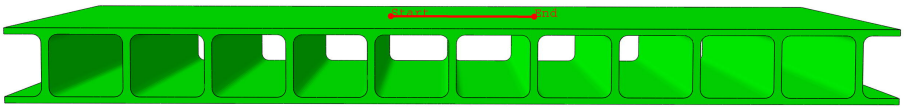


Figure 5.14: *Path on the top surface over unloaded (left) and loaded (right) cell.*

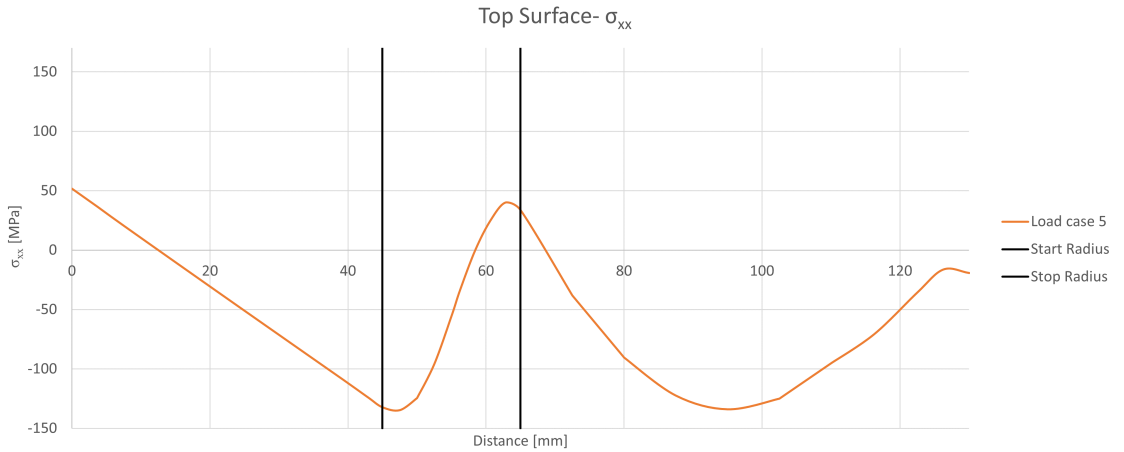


Figure 5.15: *Normal stress distribution on the top plate in x-direction over the unloaded and loaded cell, load case 5. Distance [mm] on the x-axis and stresses [MPa] on the y-axis.*

The normal stress distribution in x-direction (σ_{xx}) along the path on the bottom surface shown in Figure 5.16 are presented in Figure 5.17. As seen in Figure 5.17, the stress distribution on the bottom plate is only affected by shearing of the cells due the wheel pressure and global bending.

Figure 5.16: *Path on the bottom surface under unloaded (left) and loaded (right) cell.*

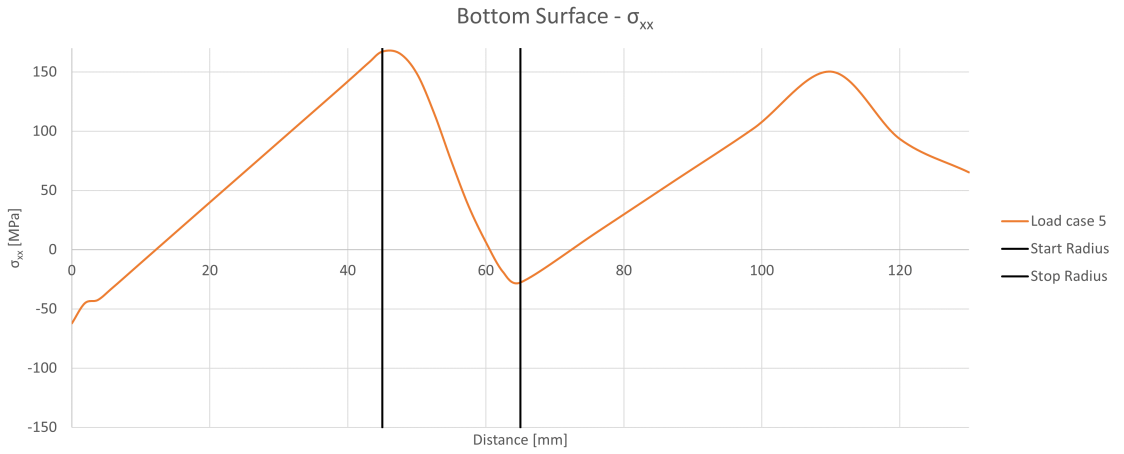


Figure 5.17: *Normal stress distribution on the bottom plate in x-direction over the unloaded and loaded cell, load case 5. Distance [mm] on the x-axis and stresses [MPa] on the y-axis.*

The normal stress distribution in x-direction (σ_{xx}) on the top surface of the entire solid model is shown in Figure 5.18 and the corresponding normal stress distribution in z-direction (σ_{zz}) is shown in Figure 5.19 for load case 5. Black lines in the figures indicate where the wheel pressure is applied. The stresses in x-direction represent local effects and the stresses in y-direction represent global bending. If comparing the two figures, it can be observed that the negative peak stresses outside of the loaded zone are almost of the same size, meaning that stresses from local shear are of the same magnitude as the global bending. This is due to the low panel height, resulting in lower bending stiffness in the strong direction (z-direction).

From Figure 5.18, it can be observed that stresses outside the loaded zone are larger compared to the loaded zone. This implies that the effects of shear are fully developed first outside of the loaded zone. In the center of the loaded zone, only the effects of local wheel pressure and no shear are visible. This is because of no relative displacement occurs for the neighbouring webs in the center of the loaded zone, which also can be seen in Figure 5.12. This also shows that the effect of local bending of the top plate between web plates due to directly applied load is lower compared to the effect of shear. This is a consequence of a relatively small *cc*-distance and large top plate thickness. The short distance between webs leads to a lower local bending moment and the large plate thickness to higher bending stiffness. Outside of the loaded zone, both positive and negative stresses are observed. This is due to the local shear-induced deformation leads to a linear bending moment diagram that changes sign somewhere close to the middle between two webs causing s-shape deformations on the top plate.

In Figure 5.19 it can be seen that the stresses varies in between -50 to -100 MPa except from under the load where the compressive stresses increase. The conclusion made from the Figures is that the oscillation of the stresses comes from the shear-induces stresses outside of the loaded cells and local bending of the top plate in the middle of the loaded cells. The value that the stress distribution oscillates around (the mean value) comes from global bending of the panel. The stress distribution in the the two direction affect each other because of Poisson's effect. This can be observed as the oscillation peaks in the x-direction are approximately three times larger than in the z-direction. The same applies in the other direction, where the stress distribution in the x-direction oscillates around a value one-third of the mean value observed in the z-direction.

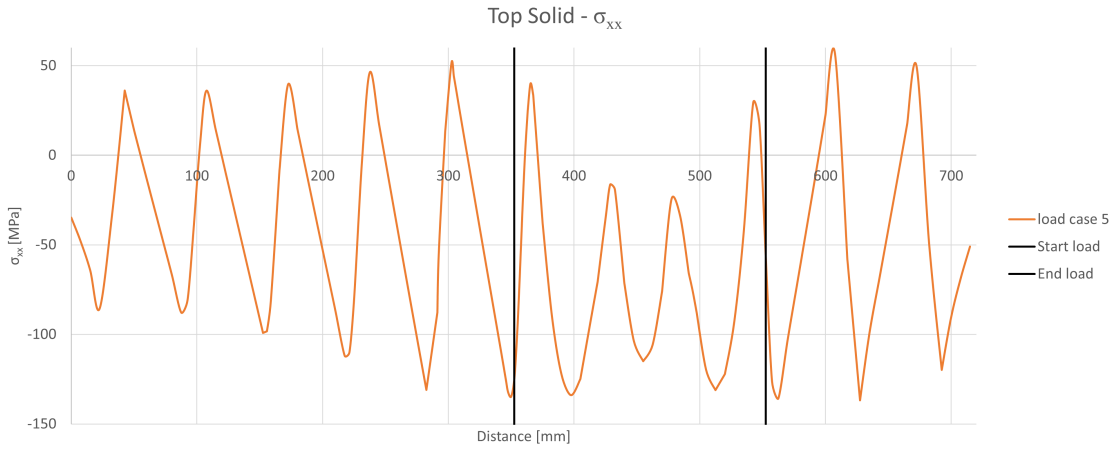


Figure 5.18: *Normal stress distribution in x-direction for the top surface of the solid model, load case 5. Distance [mm] on the x-axis and stresses [MPa] on the y-axis.*

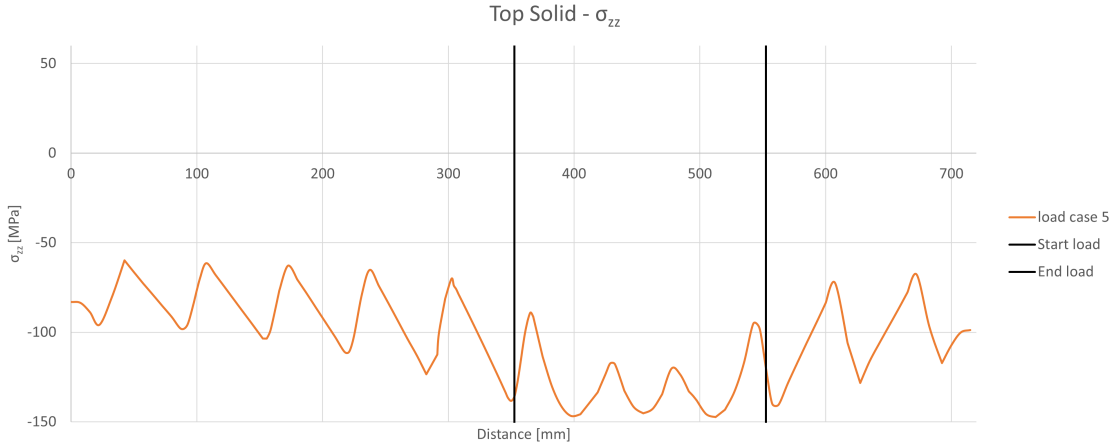


Figure 5.19: *Normal stress distribution in z-direction for the top surface of the solid model, load case 5. Distance [mm] on the x-axis and stresses [MPa] on the y-axis.*

The normal stress distribution in x-direction (σ_{xx}) on the bottom surface of the entire solid model is shown in Figure 5.20. As previously explained the normal stresses in the x-direction mainly show local effects. The normal stress distribution in z-direction (σ_{zz}) is shown in Figure 5.21. Normal stresses in the z-direction mainly come from the global bending of the deck. Since no direct load is applied on the bottom surface, the effect of shear-induced deformation and global bending are more distinguished on the bottom plate compared to the top plate. Black lines in the figures indicate where the wheel pressure is applied. The stress is close to constant over the mid cell as seen in both figures. Besides that, the plate behaves similarly as the top plate but with opposite signs.

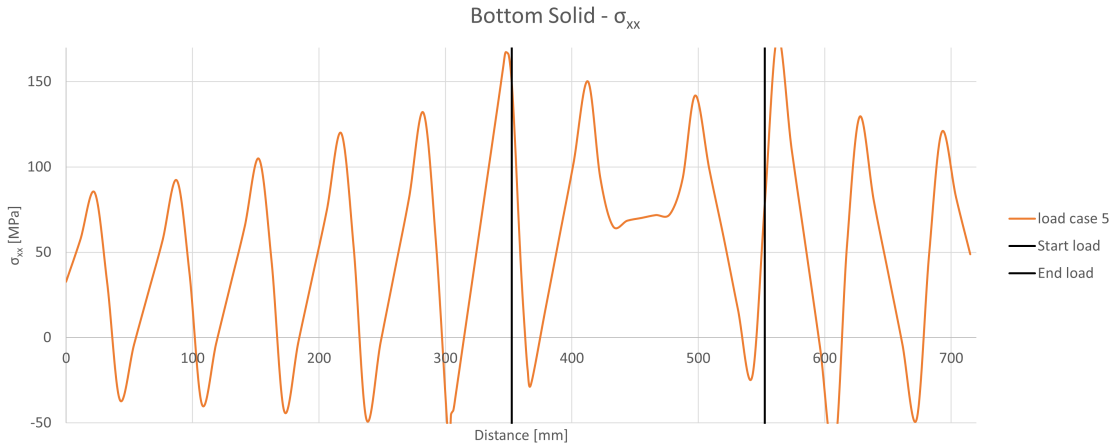


Figure 5.20: *Normal stress distribution in x-direction for the bottom surface of the solid model, load case 5. Distance [mm] on the x-axis and stresses [MPa] on the y-axis.*

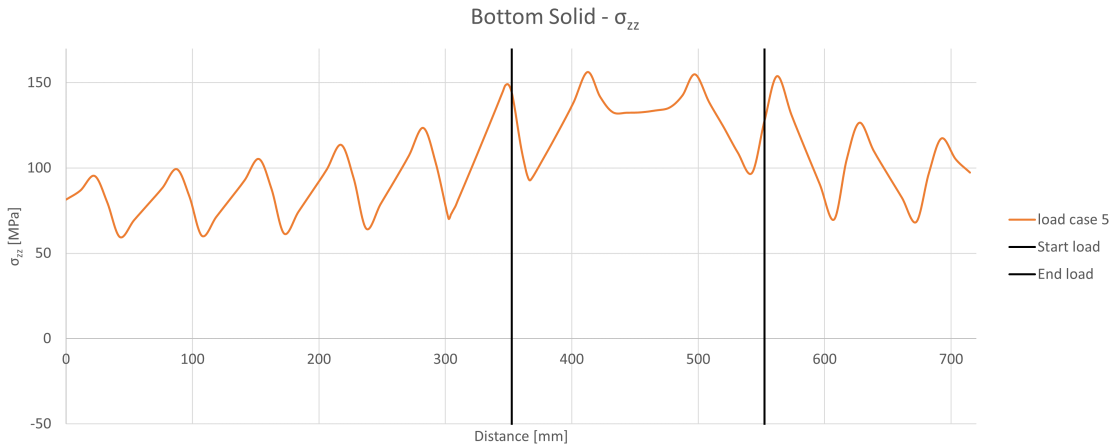


Figure 5.21: *Normal stress distribution in z-direction for the bottom surface of the solid model, load case 5. Distance [mm] on the x-axis and stresses [MPa] on the y-axis.*

5.5 Verification of design assumptions

In the following section, the results of the design verifications are presented. If the design requirements of shear, instability, and local och global deflection are met, this means that the design assumptions made in section 3.3 are valid. The design of the cross-section is thereby governed by stresses due to local load effects directly under the applied wheel pressure. The design verifications are made for the cross-sectional geometry obtained from optimization 1.

5.5.1 Shear

Shear capacity is verified with the yield criterion for von Mises stress in the position of maximum global shear in the stiff direction, presented in Section 3.4.2. To obtain von Mises stresses, the FE-model described in Chapter 4 is modified. The wheel pressure from the service vehicle is moved towards one of the edges of the plate segment.

In Table 5.9 maximum von Mises stresses from load case 1 to 5 in the new load configuration is presented. The maximum stress occurs in the top radius for load case 5 as seen in Figure 5.22. Von Mises stresses in the webs are significantly smaller than the maximum von Mises stress. The maximum von Mises stress for large global shear is less than both the maximum von Mises stress for large global bending and the design proof strength ($f_{0.d}$).

$$\sigma_{mises.max} = 146[MPa] < f_{0d} = 182[MPa]$$

Table 5.9: *Maximum von Mises stresses in the position of large global transverse shear.*

Load case	Von Mises stress [MPa]
1	140
2	145
3	138
4	145
5	**146**

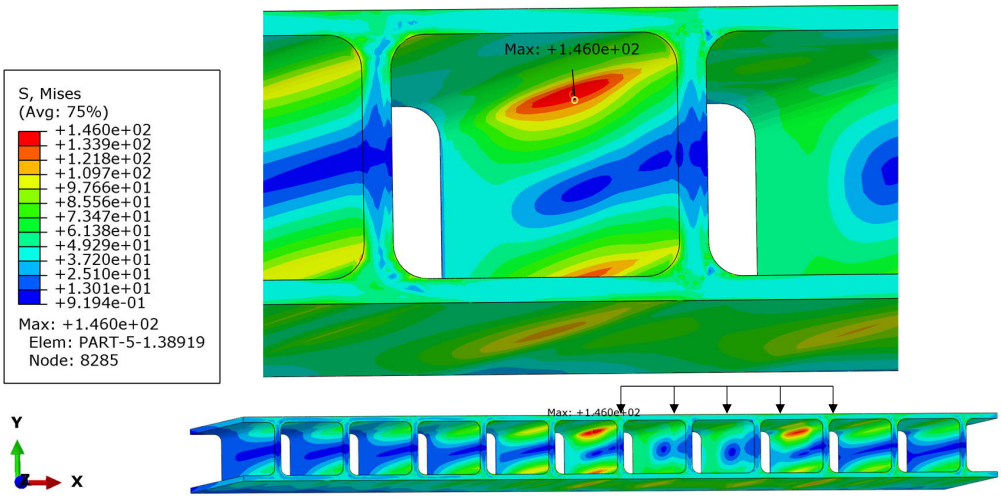


Figure 5.22: *Von Mises stress distribution in the position of large global transverse shear for load case 5. Maximum von Mises stress is highlighted. Units in [MPa].*

The distribution of shear stress τ_{yz} (*through thickness-longitudinal to extrusion*) for load case 5 are shown in Figure 5.23. The shear stresses have a small contribution to the von Mises stresses despite the positioning close to an edge of the service vehicle.

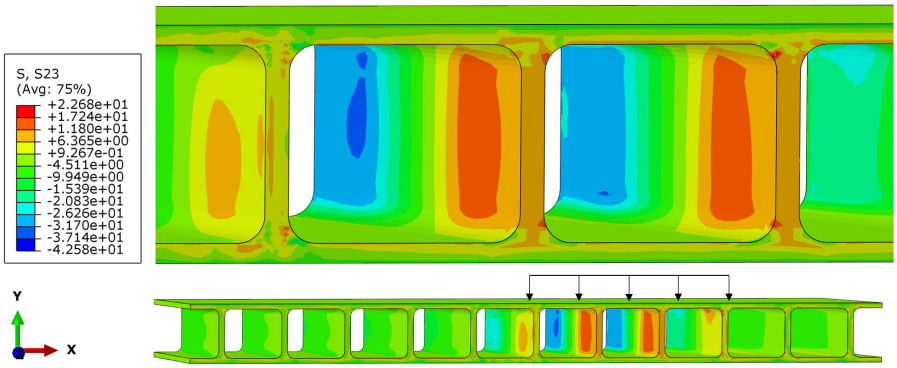


Figure 5.23: *Shear stress (τ_{yz}) distribution for load case 5. Units in [MPa].*

5.5.2 Instability

Verification of local buckling is performed according to Section 3.4.3. The local buckling resistance is sufficient if the resulting top plate thickness and cc-distance between web plates fall on or below the curve in Figure 5.24. The resulting slenderness ratio is 9, implying that the local buckling resistance is sufficient and the assumption that local buckling does not impact the design resistance is valid.

$$\beta = \frac{b}{t} = 9 \quad < \quad \beta_3 = 20.125$$

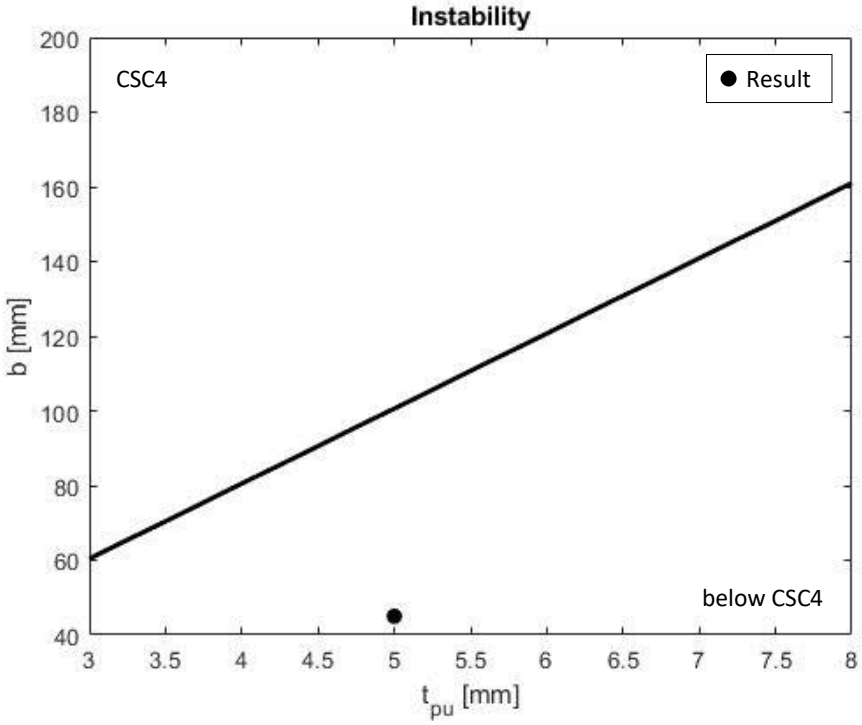


Figure 5.24: *Limiting relationship between top plate thickness and cc-distance for local buckling resistance.*

5.5.3 Global deflection

Global deflections are obtained by running the original FE-model with the resulting cross-sectional dimensions in *ABAQUS* with load combination for SLS and uniformly distributed load (q_{fk}). Load combination for SLS is presented in Table A.3 in Appendix A which is used together with the recommended ψ values according to Table A.1 in Appendix A. For the uniformly distributed load in Equation 3.1, the load is chosen to $5kN$ to be on the conservative side. The global deflection limitation is presented in Section 3.4.5.

A maximum global deflection was obtained in the middle between the supports in the transverse direction of the bridge. The deflection along the width of the deck is shown as a graph in Figure 5.25. Global deflection is not governing the design as the resulting deflection is less than the demand.

$$\delta = 6.5[mm] \quad < \quad \frac{L}{400} = 8.8[mm]$$

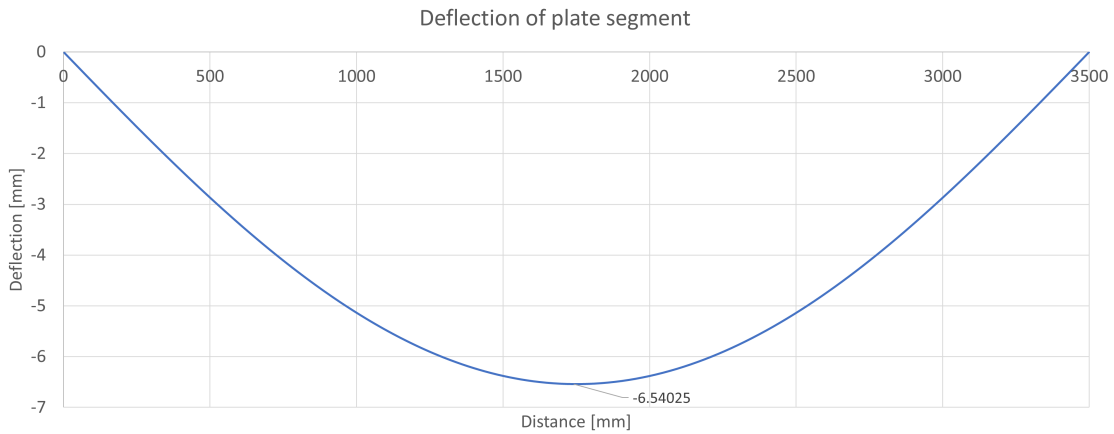


Figure 5.25: *Deflection of the entire plate in the transverse direction of the bridge. Distance [mm] on the x-axis and global deflection [mm] on the y-axis.*

5.5.4 Local deflection

Local deflection is verified according to Section 3.4.4. In Figure 5.26, the relation between the thickness of the top plate and the cc-distance between web plates is plotted as a function of the limiting condition in Equation 3.15. The function is presented in Equation 5.1. The durability of the surface coating is assumed to be fulfilled if the resulting thickness and cc-distance fall on or below the curve.

The maximum allowed cc-distance for a top plate thickness of $5mm$, calculated with Equation 5.1, is $cc_{A.max} = 139mm$ and the obtained cc-distance from the optimization is $cc = 65mm$. The local deflection is sufficiently low.

$$cc_{A.max}(t_{pu}) = \frac{t_{pu} \cdot cc_S}{t_S} \cdot \sqrt[3]{\frac{E_A}{E_S}} \tag{5.1}$$

$$cc_A = 65[mm] \quad < \quad cc_{A.max}(t_{pu}) = 139[mm]$$

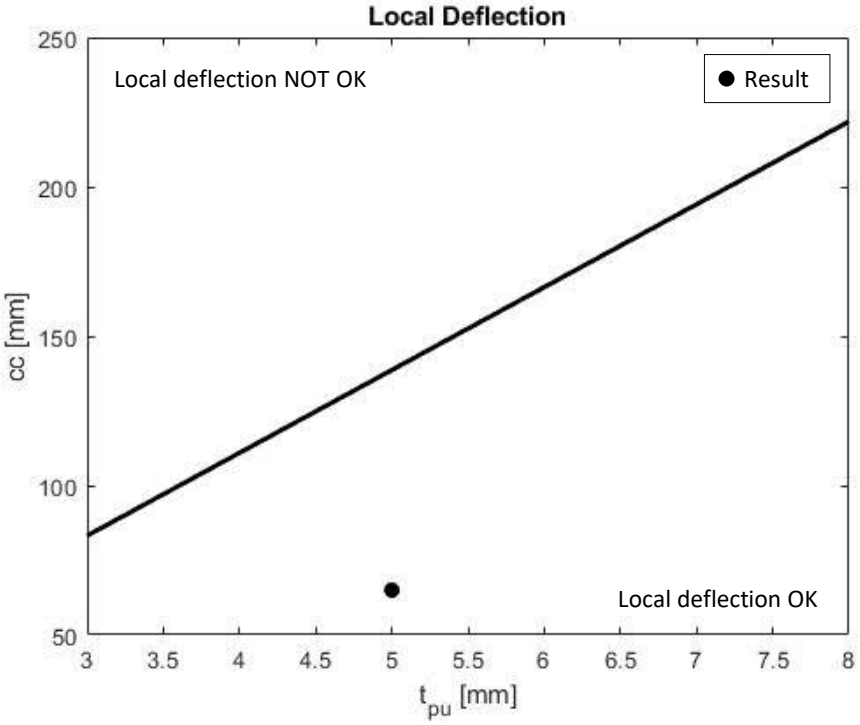


Figure 5.26: *Limiting relationship between top plate thickness and cc-distance for local deflection. cc and t_{pu} in [mm].*

5.5.5 Summary of design verifications

The shear capacity, instability, local deflection, and global deflection all fulfill the requirements for the cross-sectional geometry resulting from optimization 1. In Table 5.10 the verifications are summarized. The limitation of local deflection is however not a requirement according to Eurocode or National standards. Stresses due to directly applied load and large global bending governs the design with a Utilization Rate (UR) of 99%. This means that the design is governed by locally applied wheel pressure and the design assumptions presented in Section 3.3 hold.

Table 5.10: *Summary of design verifications from optimization 1.*

Verification	**Result**	**Limitation**	**UR**
Global bending	$\sigma_{mises.max} = 180$ [MPa]	$f_0/\gamma_{M1} = 182$ [MPa]	99 %
Global shear	$\sigma_{mises.max} = 14$ [MPa]	$f_0/\gamma_{M1} = 182$ [MPa]	80 %
Instability	$\beta = 9$	$\beta_3 = 20.125$	Not in CSC 4
Global deflections	$\delta = 6.5$ [mm]	$L/400 = 8.8$ [mm]	75 %
Local deflection	$cc_A = 65$ [mm]	$cc_{A.max} = 139$ [mm]	47 %

6
Cost Comparison

The production cost for the optimized aluminium deck is compared with conventional steel and stainless steel decks. The comparison is made mainly to give an indication of whether an aluminum bridge deck could be a competitive alternative to steel. No detailed design calculations for the steel decks are conducted. The design of connections to the main load-carrying structure is not included. The production cost is limited to the cost of the material, manufacturing, and corrosion protection. For the cost comparison, a 30m long and 3.5m wide pedestrian bridge is considered.

The cost of production, for both the aluminium deck and the steel decks, is based on the ongoing research project *Lightspan* (VINNOVA, to be published). In the research project, data from steel and aluminium workshops producing bridges are collected through interviews. The data contains information on the amount of time needed for each stage of manufacturing. The total cost for manufacturing is calculated by multiplying the time needed by the cost of one working person. The cost for one working person is estimated to be $500 SEK/h$.

6.1 Extrusion width and estimation of welding positions

In order to estimate the production cost for the aluminium bridge deck, a feasible production method is proposed with regard to Section 2.4. It is desirable to manufacture as large profiles as possible in order to limit the amount of welding. In the next step, it is desirable to manufacture as large segments with FSW-joined profiles as possible to limit the amount of MIG welding. The size of the profiles is limited by the capacity of the extrusion press and the size of the FSW-joined segments is limited by the width of the FSW-rig. Furthermore, to make a fair comparison, consideration of backing and edge preparation for the two different welding techniques is taken. As MIG and FSW require different preparations, three different profiles are proposed and therefore three different extrusion dies are needed.

Within the context of the research project VINNOVA (to be published), the three profiles presented in Figure 6.1, 6.2 and 6.3 have been suggested based on the optimized geometry. The profiles are designed with respect to the limitations of the extrusion press and the requirements of preparation for MIG and FSW. According to the producers, the common width of the FSW-rig is 3.0-3.5m. It is thereby assumed

that the maximum segment width is 3.5m.

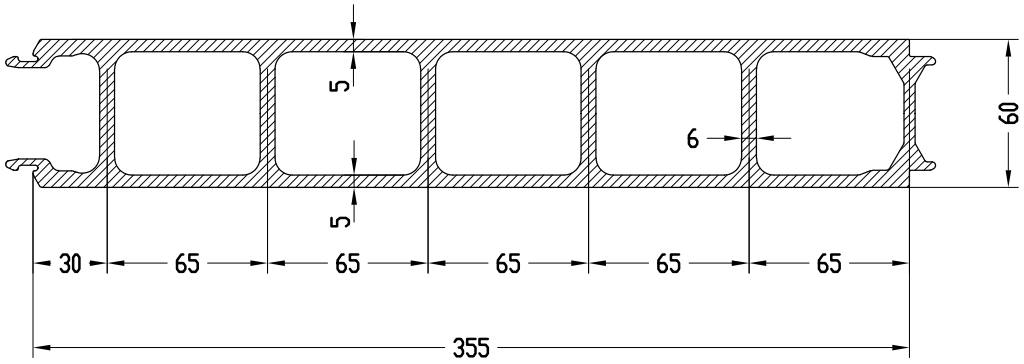


Figure 6.1: *Profile type 1: Male connection preparation for MIG-welding to the left.*

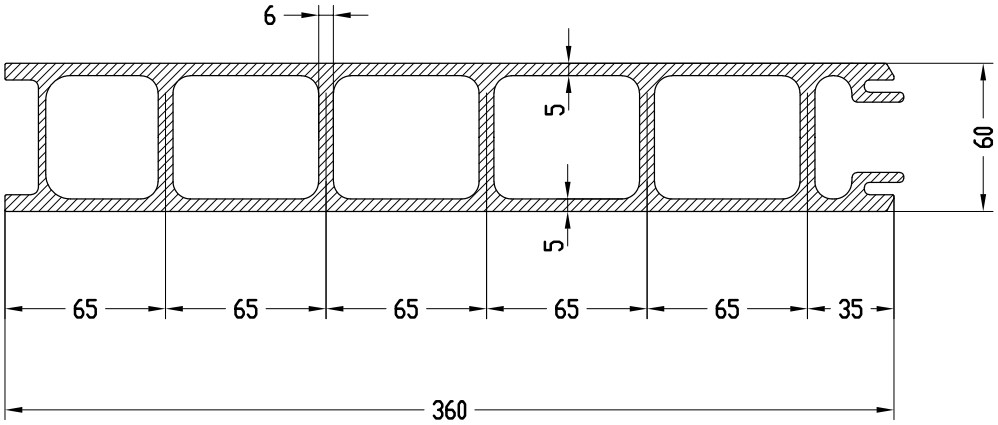


Figure 6.2: *Profile type 2: Female connection preparation for MIG-welding to the right.*

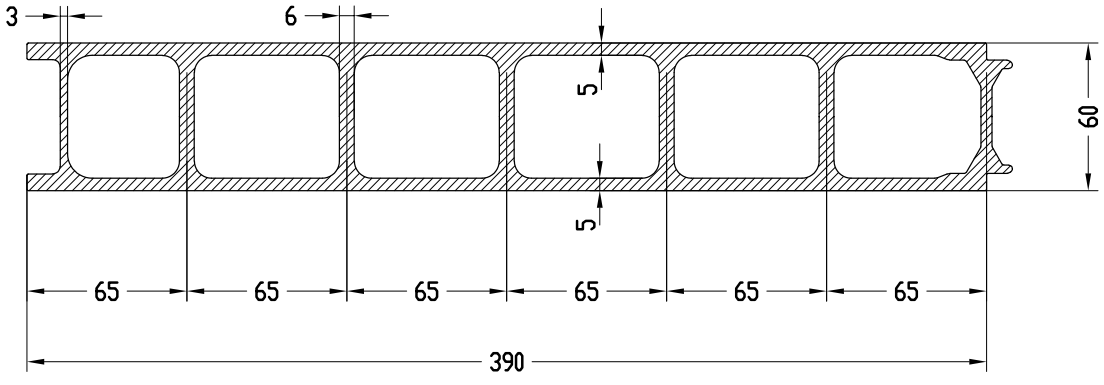


Figure 6.3: *Profile type 3: FSW connection preparations.*

Each deck segment is manufactured by joining six units of *Type 3* profiles and an additional one of each *Type 1* and *Type 2* profile at the ends by FSW. This creates 3055mm wide deck segments, see Figure 6.4. As seen in the figure, the web thickness is increased and double webs are utilized at the location of FSW connections. This can also be indicated from the profile design in Figure 6.1 (right) and Figure 6.2 (left). This is made to withstand the high pressure applied during the FSW. The individual deck segments are joined by MIG welding with the male-female connection as backing. To produce the 30m span of the reference bridge, 10 segments are needed. Resulting in a bridge span of $30.55m$ in total.

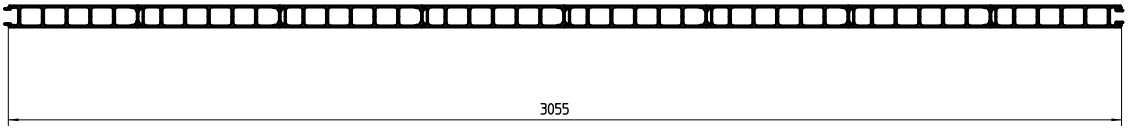


Figure 6.4: *FSW welded deck segment with length 3.055m.*

6.2 Cost estimation of aluminum deck

The production cost of the aluminium deck includes material and manufacturing. The material cost is estimated to be 37.5 SEK/kg incl. premium, based on data from *London Metal Exchange* (2023) and the currency conversion from USD to SEK. Manufacturing of the 30m long reference bridge consists of the following parameters:

- Cost of extruding 80 profiles (10 Type 1, 10 Type 2, and 60 Type 3).
- Cost of 140 FSW-welds (length 3.5m each)
- Cost of 18 MIG-welds (length 3.5m each)

The cost of extrusion, FSW, and MIG welding is calculated based on data collected within VINNOVA (to be published). Manufacturing of extrusions and welding includes aspects such as energy consumption, employees, renting of premises, logistics, equipment, Destructive Testing (DT) and Non-Destructive Testing (NDT). The cost of extrusion also includes producing three different extrusion dies. This cost has been measured as a price per kilogram, with an estimate of 2 SEK/kg. The total cost of a die is normally a one-off payment and depends on the geometry and the alloy to be extruded. Each die can be used to extrude about 75 *tonnes* of aluminium. Hence, the assumption of applying a fixed rate for the cost of the extrusion dies implies that multiple bridges are built using the same profiles. The resulting production cost of the optimized aluminium deck alternative is presented in Table 6.1.

Table 6.1: *Cost of producing the optimized aluminium deck for a bridge of 30m.*

Category	Unit	Aluminium deck
Material	$[k \cdot SEK]$	175.2
Extrusion	$[k \cdot SEK]$	170.0
FSW	$[k \cdot SEK]$	102.5
MIG	$[k \cdot SEK]$	25.2
Total	$[k \cdot SEK]$	**472.9**

6.3 Cost estimation of steel decks

Production costs are calculated for four conventional deck alternatives with different steel materials for a bridge deck of 30m. The four alternatives are presented in Table 6.2. The main difference between the two c-MN steel classes is their strength. The two stainless steel materials in Deck 3 and Deck 4 are assumed to have identical strength and stiffness properties. Deck 4 with material 1.4462 has higher corrosion resistance compared to Deck 3 with material 1.4162. According to *Krav brobyggande* Trafikverket (2019), material 1.4462 can be used in a corrosivity category up to C5, while material 1.4162 should only be used up to C4. This means that Deck 4 (1.4462) can be used in harsh marine environments without corrosion protection.

Table 6.2: *Steel and stainless steel used in the cost comparison. Properties of the materials are taken from SS-EN 1993-1-1 (2023). Material cost (VINNOVA, to be published).*

	Ordinary steel		Stainless steel	
Deck	1	2	3	4
Material	S355J2	S460N	1.4162	1.4462
Cross-section	1	2	2	2
f_y [MPa]	355	460	460	460
E [GPa]	210	210	200	200
Cost [SEK/kg]	12	14	30	40

The conventional steel decks are designed with a top plate and trapezoidal plate stiffeners, see Figure 6.5. The top plate consists of individual steel plates that are welded together with butt welds. The deck is then formed by welding the stiffeners onto the bottom of the top plate.

Two simplified deck configurations are obtained from within the research project VINNOVA (to be published), where both are designed to resist the maximum field moment for pedestrian bridge traffic according to Eurocode. Two different cross-section is needed due to the strength difference of the steel materials. The dimensions are presented in Table 6.3. It is the width and height of the stiffener and cc-distance that differs between the two cross-sections. The same plate thicknesses were used for both cross-sections to comply with the criteria of minimal thicknesses of the top plate and the stiffeners. The requirement is a minimum thickness of 10 mm for the

top plate and a minimum thickness of 6 mm for the stiffeners (SS-EN 1993-2, 2006).

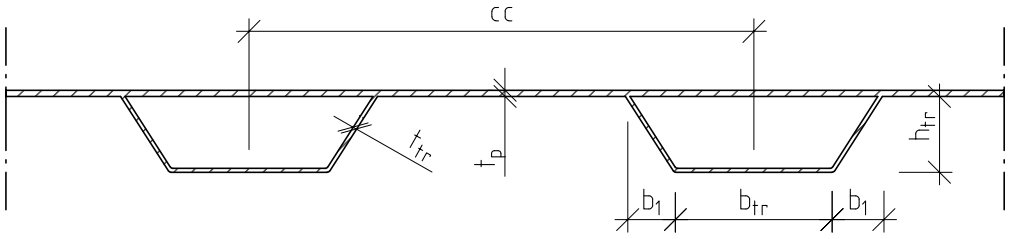


Figure 6.5: *Configuration of steel deck.*

Table 6.3: *Cross-sectional dimensions. All units in [mm].*

Cross-section:	1	2
t_p	10	10
t_{tr}	6	6
cc	625	800
b_{tr}	150	250
b_1	75	75
h_{tr}	160	120

The cost of material is calculated by multiplying the amount of steel used for the different decks with the cost presented in Table 6.2. When calculating the amount of steel, spillage due to the cutting of the steel sheets is assumed to 20% for the trapezoidal plate stiffeners and 10% for the plates.

The cost is calculated by multiplying the number of hours needed for each deck depending on the geometry, amount of welding, and cutting, with the cost of one factory hour (500 SEK/h). The following parameters are included in the calculation of manufacturing costs:

- Cutting of steel sheets for the plate and the trapezoidal plate stiffeners.
- Bending of the steel sheet for the trapezoidal plate stiffeners.
- Edge preparation of top plate for butt welds.
- Assembly and fixating of individual top plate panels with manual point welds.
- Semi-automatic welding of top plates with welding tractor.
- Assembly and fixating of trapezoidal plate stiffeners to the top plate with manual point weld.
- Semi-automatic welding of the trapezoidal plate stiffeners to the top plate with a welding tractor.

The C-Mn steel decks require painting to protect against corrosion. It is assumed that only the bottom surface will require painting as the top surface is protected by an acrylate surface coating. The cost of the protective painting is estimated to be 1000 SEK/m^2 (VINNOVA, to be published). The total cost of the painting is thereby calculated by multiplying the cost per square meter with the total surface area of the bottom of the deck. The stainless steel materials are assumed not to need any corrosion protection. However, the cost of pickling for Deck 3 and Deck 4 is assumed to be 10% of the painting cost of Deck 2. The cost of painting and pickling is included in the category of surface protection.

In addition, a cost for NDT is included. The cost of NDT is assumed to be 1/3 of the total NDT-cost for a 30m long pedestrian bridge (VINNOVA, to be published). The resulting production cost for each of the four deck alternatives is presented in Table 6.4.

Table 6.4: *Cost of producing Deck 1 to 4 in steel for a bridge of 30m.*

Category	Unit	Deck 1	Deck 2	Deck 3	Deck 4
Material	$[k \cdot SEK]$	165	184	394	525
Manufacturing	$[k \cdot SEK]$	111	97	97	97
Surface protection	$[k \cdot SEK]$	137	123	12.3	12.3
NDT	$[k \cdot SEK]$	10	10	10	10
Total	$[k \cdot SEK]$	**423**	**414**	**513**	**645**

6.4 Results of cost comparison

Production costs for the 5 deck alternatives for the reference bridge with a length of 30m are compared to each other. As stated in the previous Sections 6.2 and 6.3, all alternatives include a cost for NDT. It is assumed that the cost of NDT is the same for aluminium as for steel, therefore, 10 $k \cdot SEK$ is deducted from all alternatives.

The cost of each deck alternative, as well as comparison measurements such as cost per square meter, cost per weight, and weight per square meter of the 30m long bridge deck, is presented in Table 6.5.

Table 6.5: *Cost of all deck alternatives and comparable measures for the 30m long bridge deck.*

Category	Unit	Deck 1	Deck 2	Deck 3	Deck 4	Deck 5
Material	-	S355J2	S460N	1.4162	1.4462	AW-6005A
Bridge Surface	m^2	105	105	105	105	107
Material (spillage incl.)	*tonne*	13.9	13.1	13.1	13.1	4.7
Material (spillage excl.)	*tonne*	12.2	11.6	11.6	11.6	4.7
Total Cost (NDT excl.)	$k \cdot SEK$	413	404	503	635	463
Total Cost/m^2	$k \cdot SEK/m^2$	3.9	3.8	4.8	6.0	4.3
Total Cost/kg	SEK/kg	29.8	30.9	38.4	48.5	99.1
Weight/m^2	kg/m^2	116.2	110.5	110.5	110.5	43.7

7
Discussion

The developed optimization routine is made for one set-up of boundary conditions and loading conditions. The code is constructed such that the width of the bridge can be changed and the material properties can be substituted so that different aluminium alloys can be evaluated. However, the optimization routine has areas of development in terms of the settings of the GA and in the FE-model, both of which can affect the final results. The cost comparison is made to evaluate the competitiveness of aluminium as a material for bridge deck application, compared to steel. Only the bridge decks are considered, connections to the longitudinal load-carrying structure and Life-Cycle-Costs (LCC) are outside the scope of this thesis.

7.1 Optimization routine

The optimization routine is very time-consuming. One individual takes approximately 180 seconds to run, which is about 3 hours for one population containing 60 individuals. There are two reasons for that. First is that a license must be sought to start ABAQUS for each iteration. The second is that the FE-model created in ABAQUS is large and thus takes a long time to finalize. This limited both the number of executed optimizations and the possibility to do a proper sensitivity analysis. In order for the procedure to take a reasonable amount of time, it would be beneficial to run the entire job on a computer cluster, where individuals can be evaluated simultaneously. A more time-efficient optimization allowing for further simulations would have increased the amount of results.

A proper sensitivity study should be carried out to establish the GA settings, but due to the reasons mentioned above this could not be conducted. Rigorous requirements were therefore used. The number of best-fitted individuals (*EliteCount*) that followed to the next generation was only 5 % of the generation size, which gave the algorithm more room to try new solutions. The stopping criteria for convergence were also strict. A small tolerance over many generations was used, increasing the possibility of testing more solutions before stopping. When the algorithm is able to test many solutions, the risk of finding a local minimum is reduced. The optimization routine was executed twice with the settings described in this chapter. Due to the time frame of this thesis, there was not enough time to run it several times and with different settings. As an example, the lower and upper bounds were defined as discrete variables consisting only of integers. If the lower and upper bounds could have been narrowed and allowed for smaller steps, there is a probability that a

more optimized cross-section could have been found. For example, compared to the result from optimization 1, a decrease of 0.5mm of the top and bottom plate thicknesses would result in 6% less material for a bridge with length 30m and width 3.5m.

Only the bridge deck is included in the superstructure and the bridge's longitudinal load-carrying structure is disregarded. This means that even connections are disregarded as well as the effects of welding, their location, and preparations. To obtain a more comprehensive model, other boundary conditions could be included. Depending on the connection between the longitudinal load-carrying structure and the deck, it may provide some degree of restraint. This could, unlike simply supported, cause a negative global moment towards the edges. The results of the optimization gave a symmetrical cross-section and thus the UR would probably not be exceeded as the maximum global bending would be smaller for constant local effects. Interaction between materials in the structure could also be included in the model, such as thermal expansion coefficient differences between aluminium and, for example, steel, leading to restraint forces.

The design verifications made for the optimized cross-section are generally simplifications and could be performed with hand calculations except from the verification regarding global shear. Therefore, these could have been included as constraints in the optimization routine.

7.2 Optimization results

The optimization routine was executed twice with small deviations in the result. Optimization 1 and 2 have the same plate thickness, however the radius, cc-distance, and height have minor variations. The area obtained from optimization 2 is 0.66% higher compared to optimization 1 and the maximum stress is 1.85% lower compared to optimization 1. Optimization 2 therefore provided a cross-section with a lower UR with regard to the yield criterion for the position of large global bending. The results demonstrate the stochastic nature of a GA, where the algorithm is based on some degree of randomness and the initial population differs for each run of the optimization. All the design verifications are made for the cross-sectional geometry resulting from optimization 1. As the cc-distance between the web are larger in the first optimization, either instability or local deflection will be a problem for the second geometry. For the von Mises stresses in the position of large global transverse shear, it is unlikely that the designing proof strength is exceeded for geometry 2, since the UR for geometry 1 is 80%. Further, geometry 2 has both a larger area and the UR for large global bending is lower.

A disadvantage of optimization in combination with the implementation of a FE-model is that not all situations can be predicted in advance. All convergence studies have been carried out for a reference geometry within the range of the upper and lower bound. In Figure 5.11 it can be seen that the stress distribution at the outer edges is considerably larger in the longitudinal direction of the modeled segment than predicted by the strip length convergence study, compare with Figure 4.12.

The purpose of the convergence study was to find a minimum length where the stresses at the outer edges of the model in the longitudinal direction would be small and the stress distribution would have small deviations for increased model length. From the results of the convergence study, it can be seen that an increasing length means more spread of the load and generally lower stresses. Thus, the stresses obtained in the optimizations have been overestimated. The results are therefore on the conservative side. The accuracy of the results could be increased if a number of convergence studies had been carried out for several geometries within the upper and lower limits of the cross-sectional dimensions. The FE-model could then be more dynamically adapted. However, in some cases this means that the segment length has to be increased, which can lead to two consequences. The model size will be increased and therefore require longer computational time. Secondly, if the segment length is increased the second axle-pair can have to be included, demanding the plate length to increase further.

As described in Section 3.3, it was assumed that the design is governed by stresses directly under the applied wheel pressure. Maximum stress is obtained on the surface of the top radius. In the top surface, maximum stresses are lower than in the bottom plate as can be seen if comparing the results presented in Figure 5.6 and 5.7. This applies to the resulting cross-sectional geometry and is not a general statement for this kind of structure. The higher stresses in the bottom face plate compared to the top face plate were unanticipated. The reason for this has been explained in Section 5.4. However, stresses due to locally applied wheel pressure coincident global bending governs the design, and the assumptions hold true. The UR of all the design verifications made for the optimized cross-section is lower (see Section 5.5 and Table 5.10), which is in line with the design assumptions.

The transverse shear stiffness of a plate with web-core is generally low. This can be observed in Figure 5.12, where the plate deforms locally in the weak direction due to the applied wheel pressure. The webs are also almost vertical in the deformed shape and the sections are no longer normal to the mid-plane of the deck, implying a rotation of the sections. Transverse force can therefore not be neglected as mentioned in Section 2.3.2.

The final result indicates that a stocky cross-section with a lower height and thicker plates can be more efficient than a slender cross-section with a higher web and thinner plate thickness. The initial guess of the design was that the result would give a greater height than cc-distance. As indicated in Figure 5.4, the first four cross-sections had a greater height. This would give a larger distance to the neutral axis for the face plates which normally increases the bending stiffness. Since it was assumed that the stresses would be higher in the top plate than in the bottom plate, it was thought that the bottom plate thickness would be thinner compared to the upper one, possibly resulting in a smaller cross-sectional area. However, a stocky cross-section can be advantageous for production. A lower height allows for wider profiles to be extruded which means fewer welds and a lower cost. No general conclusions can be given for the efficiency of the cross-section before several opti-

mizations have been executed. This for greater tolerances for the step sizes of the design variables, and the GA settings have been established with a proper sensitivity study.

7.3 Cost

The result of the cost comparison presented in Table 6.5 shows that the total cost and cost per square meter of the aluminium deck (Deck 5) is greater than for the steel decks (Deck 1 and Deck 2). The total cost is however less than for the stainless steel decks (Deck 3 and Deck 4). The aluminium deck presented in this thesis can therefore be seen as a possible alternative as the cost is within the range of the presented steel decks. Although the initial cost is important for many bridge authorities in Europe, the LCC is a useful indicator. The LCC is influenced by many factors, including maintenance costs. During the lifetime of steel decks, anti-corrosion protective painting will need to be carried out continuously. If this cost is included in the comparison there is a possibility that the aluminum deck will become the most cost-effective among the presented alternatives.

To make the comparison on equal terms with regard to durability properties, the aluminium deck (Deck 5) can be compared to Deck 3 where no protective coating is required up to a moderate marine environment. Savings of 40 $k \cdot SEK$ can then be made on the total initial cost. If the aluminium deck would be designed with proper drainage and sealing of the end sections perpendicular to the extrusion in such a way that water doesn't stagnate, the durability class of the aluminium deck would probably also be comparable to Deck 4. Savings could then be made with up to 172 $k \cdot SEK$. However, it is important to remember that the costs are estimated for the bridge decks only. The limitations of this thesis are generous and a full analysis of the deck has not been conducted.

The aluminium deck (Deck 5) is significantly lighter than all other deck alternatives. The weight per square meter of the bridge deck is 60% lighter than the lightest steel alternative. Generally the self-weight of the deck has a small impact on the design of the rest of the superstructure and the sub-structure compared to the traffic load. A lighter weight can however be advantageous during transportation and assembly on-site. Other core configurations and allowing for smaller steps of the upper and lower bound in the optimization algorithm may further reduce the weight, and thus increase the competitiveness of aluminum in bridge design.

The cost for producing the extrusion dies is as mentioned normally a one-off payment but for the comparison, this is assumed to be a cost per kilogram of extruded material. This simplification is based on that multiple bridge decks are made. If only one bridge deck would be built, the initial total cost would be significantly higher compared to the steel alternatives presented. As each die can be used to extrude 75000kg of aluminium, about 16 bridges can be produced without changing the dies. This assumes that 3 different dies are needed and that all are worn out equally. In general, it can therefore be assumed that several bridge decks must be

manufactured with the same dies in order for aluminum to be an economically justified alternative in terms of initial costs.

The production cost of the extruded aluminium deck is greatly dependent on the aluminium alloy used and the cross-sectional geometry as it affects both the amount of welds and extrudability. Slender sections with sharp edges are generally more expensive to extrude. An optimization focusing on the production cost instead of just material consumption would therefore be interesting in order to minimize the investment cost and make aluminum a more competitive choice.

8

Conclusion

The aim of this thesis was to develop an optimization routine for pedestrian bridge decks constructed of hollow aluminium extrusions. The optimization routine was developed to minimize the material consumption of the cross-section while meeting the required structural performance according to Eurocode. The optimization procedure was built up using a GA function in the software MATLAB and with the implementation of a FE-module, consisting of parameterized Python scripting to create a FE-model and execute an analysis in the software ABAQUS for each iteration. The results of the optimization were used to compare the production cost between the optimized cross-section and four conventional bridge decks in steel with different properties. Discussions and conclusions made are based on what is presented in this thesis. The following concluding remarks can be made for the optimization methodology:

- The optimization is developed with an interaction between a GA and FE-analysis. The method proved to be promising as the procedure has returned reasonable and similar results.

- The procedure is very time-consuming and in order to do the necessary number of iterations and establish proper GA settings the optimization needs to be executed on a computer cluster.

- Basing the model settings on only one reference geometry is not sufficient as the load distribution varies for different geometries.

From the results of the optimization, the following can be concluded:

- The highest UR is 99%, evaluated with von Mises stresses from locally applied load placed in the position of large global bending.

- The design assumptions presented in Section 3.3 are valid for the optimized cross-section. Stresses developed in the vicinity of the locally applied wheel pressure from the service vehicle governs the design.

- A stocky cross-section seems to be more efficient compared to a slender based on the two optimizations.

Finally, from the cost comparison the following conclusions can be made:

- The results reported in this thesis indicate that bridge decks from extruded aluminium profiles can be an economically competitive solution compared to conventional orthotropic steel decks.
- The resulting optimized cross-section presented in this thesis is shown to reduce cost by 8% and 27% compared to duplex stainless steel alternatives.

- Several bridge decks must be produced with the same extrusion die in order for the aluminium deck to be an economic alternative.

9
Future studies

To capture the accurate response of an aluminium deck with varying cross-sectional geometry, a suggestion for future studies is to include dynamic settings in the FE-model. Both for the mesh in the critical section and the length of the modeled segment. An important factor for further optimizations using the method presented in this thesis is to make the optimization routine faster. Either by using a computer cluster or optimizing the efficiency of the code.

Other loading cases that would be interesting to investigate are when the deck is connected to a longitudinal load-carrying structure. Both to evaluate the response of restraint forces coming from potential differences in thermal expansion coefficients and to study the effect of fixed boundary conditions. To further investigate the competitiveness of aluminum in bridge deck construction other core geometries and different aluminium alloys need to be investigated.

Furthermore, future studies may change the objective from material consumption to investment cost. Thus, include the presented cost estimations within the optimization.

References

Agarwal, B. D., Broutman, L. J., & Chandrashekhara, K. (2018). *Analysis and performance of fibre composites*. Wiley.

Allen, H. G. (1969). *Analysis and design of sandwich panels*. Pergamon Oxford.

Arrien, P., Bastien, J., & Beaulieu, D. (2001, 12). Rehabilitation of bridges using aluminum decks. *Canadian Journal of Civil Engineering*, *28*(6), 992–1002. doi: 10.1139/cjce-28-6-992

Avén, S., Lantz, H., & Lorentsen, M. (1983). *Handboken Bygg Byggtabeller*.

Bartolozzi, G., Pierini, M., Orrenius, U., & Baldanzini, N. (2013, 6). An equivalent material formulation for sinusoidal corrugated cores of structural sandwich panels. *Composite Structures*, *100*, 173–185. doi: 10.1016/j.compstruct.2012.12.042

Cusson, B. (2022). *Designing and Detailing a GMAW Welded Aluminium Deck on Steel Girders for CSA S6:19 Traffic Loads* (Tech. Rep.). Toronto: WSP.

Djedid, A., Guillot, M., Desjardins, V., Annan, C.-D., & Fafard, M. (2020). Design and fabrication of bridge decks made from extruded aluminium. doi: 10.11283/jlwa.58.138s

Dobmeier, J. M., Barton, F. W., Gomez, J. P., Massarelli, P. J., & McKeel Jr., W. T. (2001). Failure study of an aluminum bridge deck panel. *Journal of Performance of Constructed Facilities*, *15*(2), 68 – 76. doi: 10.1061/(ASCE)0887-3828(2001)15:2(68)

EN 1990 Annex A2. (2004). *Eurocode - Basis of structural design* (Tech. Rep.).

Holland, J. H. (1992). Genetic algorithms. *Scientific American*, *267*(1), 66–73. doi: 10.2307/24939139

Hydro Building Systems Sweden AB. (2022, 2). *GOLVSYSTEM ALUMINIUMPROFILER FÖR INDUSTRI OCH OFFENTLIG MILJÖ*. Retrieved from `www.sapabuildingsystem.se`

Höglund, T., & Nilsson, L. (2006). Aluminium in bridge decks and in a new military bridge in sweden. *Structural Engineering International: Journal of the International Association for Bridge and Structural Engineering (IABSE)*, *16*(4), 348 – 351. doi: 10.2749/101686606778995100

Libove, C., & Batdorf, S. B. (1948). A general small-deflection theory for flat sandwich plates.

Libove, C., & Hubka, R. E. (1951). Elastic constants for corrugated-core sandwich plates.

Lindqvist, J., & Svensson, A. (2022). *Laser Welded Steel Sandwich Panels for Pedestrian Bridge Application A Case Study* . Chalmers University of Technology.

Lok, T.-S., & Cheng, Q.-H. (2000). Elastic stiffness properties and behavior of

truss-core sandwich panel. *Journal of structural engineering New York, N.Y.*, *126*(5), 552 – 559. doi: 10.1061/(ASCE)0733-9445(2000)126:5(552)

London Metal Exchange. (2023, 5). Retrieved from `https://www.lme.com/en/Metals/Non-ferrous/LME-Aluminium#Trading+day+summary`

Massicotte, B., El-Hage, B., Lagier, F., & Fafard, M. (2020). ANALYSE DE LA RÉPARTITION DES CHARGES ROUTIÈRES POUR LES TABLIERS DE TYPE PLATELAGE EN ALUMINIUM SUR POUTRES EN ACIER.

Mathers, G. (2002). *The welding of aluminium and its alloys*. CRC Press.

Mazzolani, F. M. (2003). *Aluminium structural design*. Springer Science & Business Media.

Mazzolani, F. M. (2004, 10). Competing issues for aluminium alloys in structural engineering. *Progress in Structural Engineering and Materials*, *6*(4), 185–196. doi: 10.1002/pse.178

Misch, P. C., Graduate Research Assistant Furman Barton, J. W., Faculty Research Scientist Jose Gomez, P. P., Senior Research Scientist Peter Massarelli, P. J., McKeel, W. T., & Research Manager, P. (1999). *FINAL REPORT EXPERIMENTAL AND ANALYTICAL EVALUATION OF AN ALUMINUM DECK BRIDGE* (Tech. Rep.).

Mishra, R. S., & Ma, Z. Y. (2005, 8). Friction stir welding and processing. *Materials Science and Engineering R: Reports*, *50*(1), 1-78. doi: 10.1016/j.mser.2005.07.001

Nazemi, N., & Ghrib, F. (2019). Strength characteristics of heat-affected zones in welded aluminum connections. *Journal of Engineering Mechanics*, *145*(12). doi: 10.1061/(ASCE)EM.1943-7889.0001647

Nilsson, P. (2015). *Steel-sandwich elements in bridge applications* (Tech. Rep.). Chalmers University of Technology.

Nilsson, P., Al-Emrani, M., & Atashipour, S. R. (2017, 8). Transverse shear stiffness of corrugated core steel sandwich panels with dual weld lines. *Thin-Walled Structures*, *117*, 98–112. doi: 10.1016/j.tws.2017.04.008

Nilsson Strand, P. (2020). *Laser-welded corrugated core steel sandwich bridge decks*. Gothenburg.

Norsk Hydro ASA. (2023). *Design manual by Hydro.* Retrieved from `https://www.hydroaluminiumdesign.com/en`

Nyström, H. (n.d.). *We are aluminium* (Tech. Rep.).

Okura, I., Osawa, S., Takeno, M., Hagisawa, N., & Ishikawa, T. (2006). Development of aluminum decks for highway bridges. In *Iabse symposium report* (Vol. 92, pp. 26–33).

PANTURA. (2011). *D5.3 Needs for maintenance and refurbishment of bridges in urban environments* (Tech. Rep.). Chalmers University of Technology.

Raknes Brekke, C. A. (2017). *Concept Development of an Aluminum Pedestrian Bridge* . Norwegian University of Science and Technology.

Reddy, J. (2004). *Mechanics of laminated composite plates and shells –Theory and analysis* (2nd ed. ed.).

Romanoff, J. (2007). *Bending Response of Laser-Welded Web-Core Sandwich Plates* (Tech. Rep.). Retrieved from `https://www.researchgate.net/publication/27516700`

Romanoff, J., & Varsta, P. (2005, 6). Bending response of web-core sandwich beams. *Composite Structures*, *73*(4), 478–487. doi: 10.1016/j.compstruct.2005.02.018

Saha, P. K. (2000). *Aluminum Extrusion Technology.* Asm International.

Saleem, M. A., Mirmiran, A., Xia, J., & Mackie, K. (2012, 1). Experimental Evaluation of Aluminum Bridge Deck System. *Journal of Bridge Engineering*, *17*(1). doi: 10.1061/(asce)be.1943-5592.0000204

Sivanandam, S. N., & Deepa, S. N. (2008). *Introduction to Genetic Algorithms.* Springer-Verlag, Berlin Heidelbergs.

Siwowski, T. (2006). Aluminium bridges–past, present and future. *Structural engineering international*, *16*(4), 286–293.

Siwowski, T. W. (2009). Structural behaviour of aluminium bridge deck panels. *Engineering Structures*, *31*(7), 1349 – 1353. doi: 10.1016/j.engstruct.2009.02.002

SS-EN 1011-4. (2001). *Welding-Recommendations for welding of metallic materials - Part 4: Arc welding of aluminium and aluminium alloys* (Tech. Rep.). Retrieved from `www.sis.se`

SS-EN 1090-3. (2019). *Execution of steel structures and aluminium structures - Part 3: Technical requirements for aluminium structures* (Tech. Rep.). Retrieved from `www.sis.se`

SS-EN 1990. (2002). *Eurocode - Basis of structural design* (Tech. Rep.). Retrieved from `www.sis.se`

SS-EN 1991-2. (2003). *Eurocode 1: Action on structures - Part 2: Traffic loads on bridges* (Tech. Rep.). Retrieved from `www.eurokoder.se,`

SS-EN 1993-1-1. (2023). *Eurokod 3-Dimensionering av stålkonstruktioner-Del 1-1: Allmänna regler och regler för byggnader Eurocode 3-Design of steel structures-Part 1-1: General rules and rules for buildings* (Tech. Rep.). Retrieved from `www.sis.se`

SS-EN 1993-2. (2006). *Eurocode 3: Design of steel structures - Part 2: Steel bridges* (Tech. Rep.). Retrieved from `www.eurokoder.se,`

SS-EN 1999-1-1. (2007). *Eurocode 9: Design of aluminium structures - Part 1-1: General structural rules* (Tech. Rep.). Retrieved from `www.sis.se`

SS-EN 1999-1-2. (2007). *Eurocode 9: Design of aluminium structures - Part 1-2: Structural fire design* (Tech. Rep.). Retrieved from `www.sis.se`

SS-EN-573-1. (2005). *Aluminium and aluminium alloys-Chemical composition and form of wrought products-Part 1: Numerical designation system* (Tech. Rep.). Retrieved from `www.sis.se`

Threadgilll, P. L., Leonard, A. J., Shercliff, H. R., & Withers, P. J. (2009, 03). Friction stir welding of aluminium alloys. *International Materials Reviews*, *54* , 49-93. doi: 10.1179/174328009X411136

Trafikverket. (2019). Krav Brobyggande (TDOK 2016:0204).

VINNOVA. (to be published). *Lightweight structures for robotised production of large span bridges and buildings - LIGHTSPAN.* Retrieved from `https://www.vinnova.se/p/lattviktslosningar-for-robotiserad-produktion-av-33broar-och-byggnadsverk-med-stora-spannvidder---lightspan/`

Vinson, J. R. (2001). *Sandwich structures* (Tech. Rep.). Retrieved from `http://asmedigitalcollection.asme.org/appliedmechanicsreviews/`

article-pdf/54/3/201/5438082/201_1.pdf

Walbridge, S., & de la Chevrotiére, A. (2019). Opportunities for the use of Aluminum in Vehicular Bridge Construction. *Aluminium Association of Canada Report, MAADI Group, Montreal*.

Yang, X. S. (2010). *Engineering Optimization: An Introduction with Metaheuristic Applications*. John Wiley & Sons. doi: 10.1002/9780470640425

Zenkert, D. (1995). *An introduction to sandwich structures*. Engineering Materials Advisory Services.

Zenkert, D. (1997). *The handbook of Sandwich Construction*. EMAS Publishing.

Zhang, Y., Qiu, J., & Bai, H. (2012). Application and research progress of aluminum alloy bridge decks. In *Key engineering materials* (Vol. 517, pp. 763–770). Trans Tech Publications Ltd. doi: 10.4028/www.scientific.net/KEM.517.763

A

Load combinations in ULS and SLS

Table A.1: *Recommended values of ψ for footbridges (EN 1990 Annex A2, 2004)*

Action	**Symbol**	ψ_0	ψ_1	ψ_2
Traffic loads	gr1	0.40	0.40	0
	Q_{fwk}	0	0	0
	gr2	0	0	0

Design values for actions in Ultimate Limit State and Serviceability Limit State:

Table A.2: *STR, ultimate limit state (EN 1990 Annex A2, 2004)*

	Permanent actions		**Variable action**	**Accompanying variable actions**	
	Unfavourable	Favourable	Leading	Main	Other
Eq. 3.2	1.35 $G_{k,j.sup}$	1.0 $G_{k,j.inf}$		1.5 $\psi_{0,1}$ $Q_{k,1}$	1.5 $\psi_{0,i}$ $Q_{k,i}$
Eq. 3.3	0.85 x 1.35 $G_{k,j.sup}$	1.0 $G_{k,j.inf}$	1.5 $Q_{k,1}$		1.5 $\psi_{0,i}$ $Q_{k,i}$

Table A.3: *Frequent load combination, serviceability limit state (EN 1990 Annex A2, 2004)*

	Permanent actions		**Variable actions**	
	Unfavourable	Favourable	Leading	Other
Eq. 3.4	1.0 $G_{k,j.sup}$	1.0 $G_{k,j.inf}$	1.0 ψ_1 $Q_{k,1}$	1.0 ψ_2 $Q_{k,i}$

B

Seed-sizes of FE-model

The final seed-sizes used in the model are presented below. For cross-section parameters see 4.3. Seed-sizes for the shell model are presented in Figure B.1 and Table B.1. Seed-sizes for the solid model are presented in Figure B.2 and Table B.2. The dimensions obtained from the formulas are the approximate element size in the indicated direction.

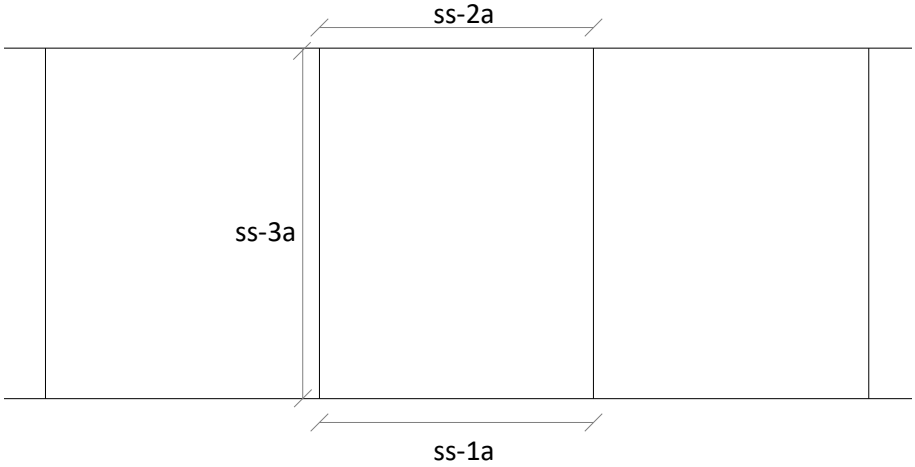


Figure B.1: *Index of edges for seeding - shell model.*

Table B.1: *Seed-sizes of the shell model. ss-Za.1 and ss-Za.2 are used in z-direction close to the solid model and further away from it, respectively.*

Index	Formula
ss-1a	$cc/4$
ss-2a	$cc/6$
ss-3a	$h/5$
ss-Za.1	2· ss-2a
ss-Za.2	3· ss-2a

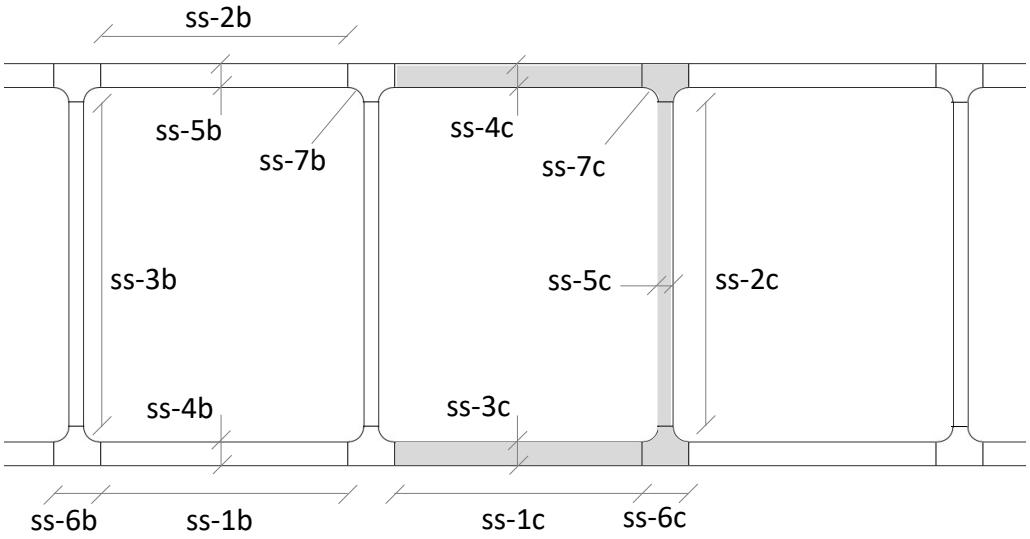


Figure B.2: *Index of edges for seeding - solid model. The gray area represents the finer mesh area*

Table B.2: *Seed-sizes of the solid model. ss-Zb and ss-Zc are the seed-sizes in z-direction.*

Coarser		**Finer**	
Index	Formula	Index	Formula
ss-1b	$(cc - t_w - 2r)/4$	ss-1c	$(cc - t_w - 2r)/24$
ss-2b	$(cc - t_w - 2r)/6$	ss-2c	$(h - t_{pu} - t_{pb} - 2r)/20$
ss-3b	$(h - t_{pu} - t_{pb} - 2r)/5$	ss-3c	$t_{pb}/4$
ss-4b	$t_{pb}/2$	ss-4c	$t_{pu}/4$
ss-5b	$t_{pu}/2$	ss-5c	$t_w/4$
ss-6b	$(t_w + 2r)/2$	ss-6c	$(t_w + 2r)/8$
ss-7b	$(2\pi r/4)/4$	ss-7c	$(2\pi r/4)/12$
ss-Zb	2· ss-2b	ss-Zc	50/10

C

MATLAB script

Appendix C contains the MATLAB scripts that represent the genetic algorithm part of the developed optimization routine. It includes one main script and three functions. The genetic algorithm function from MATLAB's Global Optimization Toolbox is implemented in the main script.

- Main MATLAB script
- Objective function
- Constraint function
- Function to implement the FE-module

```
%--------------------------------------------------------------------------
% Master's Thesis
%
% Extruded aluminium decks for pedestrian bridges
% – Design optimization using genetic algorithm
%
% Emelie Falkenberg and Markus Fredriksson
% 2023
%
%
% Main script
%
%--------------------------------------------------------------------------

clear all
clear figure
clc

%------------------------- Input and output -------------------------------
% Global material parameters
global fy
global sig_vM_max

% Yield limit (proof strength)
gamma_M1 = 1.1;
fy=200/gamma_M1;
```

```
27
28 % ------------------ Settings for optimization problem -----------------
29 % Number of variables
30 nvars=6;
31
32 % Global geometry parameters
33 global CC
34 global R
35 global H
36 global TFU
37 global TW
38 global TFB
39
40 CC = 50:5:200;
41 R = 1:1:10;
42 H = 50:10:200;
43 TFU = 3:1:8;
44 TW = 4:1:8;
45 TFB = 3:1:8;
46
47 % Lower bound
48 LB=ones(1,nvars);
49
50 % Upper bound
51 UB=zeros(1,nvars);
52 UB(1) = numel(CC);
53 UB(2) = numel(R);
54 UB(3) = numel(H);
55 UB(4) = numel(TFU);
56 UB(5) = numel(TW);
57 UB(6) = numel(TFB);
58
59
60 % Initial guesses
61 for i = 1:nvars
62   X0(i) = 1;
63 end
64
65 % Add options to control GA settings
66 options = optimoptions(@ga, ...
67                        'PopulationSize', nvars*10, ... % (Population
                           size)
68                        'MaxGenerations', nvars*50, ... % (Will stop at
                           300 even if no solution is found)
69                        'EliteCount', 3, ...            % (5% of
                           population size)
```

```
70                      'MaxStallGenerations', 20, ...  % (50 as default)
71                      'FunctionTolerance', 1e-6, ...
72                      'PlotFcn', @gaplotbestf); ...
73                      InitialPopulationMatrix = X0;
74
75
76 % Call objective function and constraints function
77 ObjFcn= @Fitness_Func_plate;
78 ConsFcn= @constraints_plate;
79
80 %------------------------ Run optimization problem ---------------------
81 [x,fval,exitflag]= ga (ObjFcn, nvars, [], [], [], [], LB, UB, ConsFcn
       , 1:nvars, options);
82 % x = Vector containg resulting indexes  in the possible range
       defined by the lower and upper bounds
83 % fval = Resulting value of the fittness function - Area
84 % existflag = Integer corresponding to exit message
85
86 %---------------------------- Display results ---------------------------
87 [cc r H tfu tw tfb ];
88 [CC(x(1)) R(x(2)) H(x(3)) TFU(x(4)) TW(x(5)) TFB(x(6))];
89 Area = fval;
90 max_stress = sig_vM_max;
91
92 %---------------------- Write results to text file ---------------------
93 GAresults = [max_stress Area CC(x(1)) R(x(2)) H(x(3)) TFU(x(4)) TW(x
       (5)) TFB(x(6)) exitflag];
94 fileID = fopen('ResultGAdata.txt','w');
95 fprintf(fileID,'%8.7f\n',GAresults);
96 fclose(fileID);
```

```
function Area = Fitness_Func_plate(x)
%--------------------------------------------------------------------------
% Master's Thesis
%
% Extruded aluminium decks for pedestrian bridges
% - Design optimization using genetic algorithm
%
% Emelie Falkenberg and Markus Fredriksson
% 2023
%
%
% Objective function
%
%--------------------------------------------------------------------------

% Global geometry parameters
global CC
global R
global H
global TFU
global TW
global TFB

cc = CC(x(1));
r = R(x(2));
h = H(x(3));
tfu = TFU(x(4));
tw = TW(x(5));
tfb = TFB(x(6));

% Calculate area of cross-section (Area*nprofiles/cc*nprofiles = Area
    /m)
Area = (cc*tfu + cc*tfb + (h-tfu-tfb)*tw + (((2*r)^2) - (pi*r^2)))/cc

end
```

```
function [c, ceq] = constraints_plate(x)
% ------------------------------------------------------------------------
% Master's Thesis
%
% Extruded aluminium decks for pedestrian bridges
% - Design optimization using genetic algorithm
%
% Emelie Falkenberg and Markus Fredriksson
% 2023
%
%
% Constraint function
%
% ------------------------------------------------------------------------

% Global input and output parameters
global fy
global sig_vM_max
global CC
global R
global H
global TFU
global TW
global TFB


% ------------------- Call function to run FE-module -------------------
[sig_vM_max] = run_abaqus_python_plate(x)


% ------- Writes information from each iteration to a txt file -------
cc = CC(x(1));
r = R(x(2));
h = H(x(3));
tfu = TFU(x(4));
tw = TW(x(5));
tfb = TFB(x(6));

Invividual_area = (cc*tfu + cc*tfb + (h-tfu-tfb)*tw + (((2*r)^2) - (
    pi*r^2)))/cc;
Invividual_GA_data = [sig_vM_max Invividual_area cc r h tfu tw tfb];
fileID = fopen('AllGAdata.txt','a');
fprintf(fileID,'%0.4f ',Invividual_GA_data);
fprintf(fileID,'\n');
fclose(fileID);
```

```
45
46 % ------------------------------ Constraints ------------------------------
47 % Constraint 1 - yielding
48 C1 = sig_vM_max - fy;
49
50 % Constraint 2 - tw to h ratio
51 if h <= 100.0
52     tw_min = 4.0;
53 elseif h > 100.0 && h <= 150.0
54     tw_min = 5.0;
55 else
56     tw_min = 6.0;
57 end
58
59 C2 = tw_min - tw;
60
61 ceq = [];
62 c = [C1 C2];
63
64 end
```

```
function [sig_vM_max] = run_abaqus_python_plate(x)
% ------------------------------------------------------------------------
% Master's Thesis
%
% Extruded aluminium decks for pedestrian bridges
% – Design optimization using genetic algorithm
%
% Emelie Falkenberg and Markus Fredriksson
% 2023
%
%
% Run FE-module
%
% ------------------------------------------------------------------------

% Global geometry parameters
global CC
global R
global H
global TFU
global TW
global TFB

variables = [CC(x(1)) R(x(2)) H(x(3)) TFU(x(4)) TW(x(5)) TFB(x(6))];

% ------------------- Save variables to a text file -------------------
% 6 = stops to write on column 6
% .x x number of decimals
fileID = fopen('variables.txt','w');
fprintf(fileID,'%6.1f\n',variables);
fclose(fileID);

% ------------------ Run python script in command promt ----------------
system(['abaqus cae noGUI=shell_solid_plate_path2.py']);

% ----- Import Abaqus Von Mises stresses (output) from FE-module -----
fileID = fopen('results_mises.txt','r');
mises_stresses = fscanf(fileID, '%f');
fclose(fileID);

% Von Mises stress
sig_vM_max = max(mises_stresses);

end
```

D

PYTHON script

Appendix D contains the PYTHON script that represent the FE module of the developed optimization routine.

Note: The *#Set Directory* has been removed in the code (line 31) due to a interfering bug with the LaTeX-compiler and the code. This line is however only of necessity when a directory needs to be specified.

```
1  # ===========================================================
2  # Master's Thesis
3  #
4  # Extruded aluminium decks for pedestrian bridges
5  # – Design optimization using genetic algorithm
6  #
7  # Emelie Falkenberg and Markus Fredriksson
8  # 2023
9  #
10 # ===========================================================
11 # ==================== IMPORT MODULES =======================
12 import subprocess
13 import os
14 import math
15 import numpy as np
16 import sys
17 from types import IntType
18
19 from abaqus import *
20 from abaqusConstants import *
21 from caeModules import *
22 from odbAccess import *
23
24
25
26 # ======================== SETTINGS =========================
27 # Number of cores used for running the FE simulation
28 n_cpus = 8
29
30 # Set directory
```

```
31
32
33
34 # ================== IMPORT VARIABLES ====================
35 # Reading independent variables from text-file generated in
       matlab
36 text_file = open('variables.txt', 'r')
37 lines = text_file.readlines()
38
39 cc = float(lines[0])      # cc distance
40 r = float(lines[1])       # Radius
41 H = float(lines[2])       # Total height
42 tfu = float(lines[3])     # Thickness upper flange
43 tw = float(lines[4])      # Thickness web
44 tfb = float(lines[5])     # Thickness bottom flange
45
46
47
48 # ========== FIXED VARIABLES (material property) ==========
49 density = 2.7e-09         # Density of
50 youngs_modulus = 70000.0 # Youngs modulus
51 poissons_ratio = 0.3      # Poissons ratio
52
53
54
55 # ============== FIXED VARIABLES (geometry) ===============
56 length = 3500.0                  # Length of extrusion
57 Bmin = 3000.0                    # Min width of plate
58 minSolid = 700.0                 # Min width of cut out
59 solid_width = 400.0              # Min length of cut out
60 axle_dist = 1300.0               # Distance between axles
61 z_centre_axle_loads = length/2  # Centre position between
       axle loads
62
63 # Numer of profiles for the entire plate
64 if Bmin % cc == 0:
65     nprofiles = Bmin / cc
66 else:
67     nprofiles = math.ceil(Bmin / cc)
68
69 # Numer of profiles for cut out / solid part
70 if minSolid % cc == 0:
71     solidprofiles = minSolid / cc
72 else:
73     solidprofiles = math.ceil(minSolid / cc)
74
```

```
75
76 # X coordinates for the cut out / solid part
77 if nprofiles % 2 == 0:        # Even
78     if solidprofiles % 2 == 0: # Even (cc*solidprofiles/2)
79         xpoint1 = cc * nprofiles/2 - cc*solidprofiles/2
80         xpoint2 = cc * nprofiles/2 + cc*solidprofiles/2
81     else:                      # Uneven (cc*solidprofiles/2
       +- cc/2)
82         xpoint1 = cc * nprofiles/2 - cc*solidprofiles/2 -
           cc/2
83         xpoint2 = cc * nprofiles/2 + cc*solidprofiles/2 +
           cc/2
84         solidprofiles = solidprofiles + 1
85 else:                          # Uneven
86     if solidprofiles % 2 == 0: # Even (cc*solidprofiles/2
       +- cc/2)
87         xpoint1 = cc * nprofiles/2 - cc*solidprofiles/2 -
           cc/2
88         xpoint2 = cc * nprofiles/2 + cc*solidprofiles/2 +
           cc/2
89         solidprofiles = solidprofiles + 1
90     else:                      # Uneven (cc*solidprofiles/2)
91         xpoint1 = cc * nprofiles/2 - cc*solidprofiles/2
92         xpoint2 = cc * nprofiles/2 + cc*solidprofiles/2
93
94 # Z coordinates for the cut out / solid part (fixed width)
95 zpoint1 = z_centre_axle_loads - axle_dist/2 - solid_width/2
96 zpoint2 = z_centre_axle_loads - axle_dist/2 + solid_width/2
97
98
99 # Coordinates (local) on solid
100 zpositions_solid_partition = np.array([100.0, solid_width/2
      - 25.0, solid_width/2 + 25.0, 300.0])
101 zpositions_solid_top = np.array([0.0, 100.0, solid_width/2
    - 25.0, solid_width/2 + 25.0, 300.0, solid_width])
102 zpositions_solid = np.array([0.0, solid_width/2 - 25.0,
    solid_width/2 + 25.0, solid_width])
103 ypositions_solid = np.array([0.0, tfb, H-tfu, H])
104 ypositions_solid_web = np.array([tfb+r, H-tfu-r])
105 ypositions_solid_z = np.array([0, tfb, tfb+r, H-tfu-r, H-
    tfu, H])
106
107
108
109 # ================ FIXED VARIABLES (load) =================
110 gravity = 0.00982                                        #
```

```
    Gravity constant (mm/s2)
111 coating = 0.00022                                    #
    Distributed load from surface coating (10mm akrylat)
112 axle_load = 1.0                                      # Load
     from service vehicle (80/(2*0.2*0.2))
113 factor_self_weight = 0.85*1.35                       #
    Partial factor for load combinations - self weight
114 factor_coating = 0.85*1.35                           #
    Partial factor for load combinations - surface coating
115 factor_traffic_load = 1.5                            #
    Partial factor for load combinations - Service vehicle
116
117 load_length = 200.0                                  #
    Dimensions of wheel pressure 200x200
118 load_step = 5                                        #
    Number of loading steps
119 load_length_tot = load_length + cc                   # Tot
    length for load to move in X
120 step_length = load_length_tot/(load_step-1)          # Step
     length in X
121 load_start = math.floor(solidprofiles/2)*cc - cc/2   #
    First loading pos in local x
122
123 # Load positions from (0,0,0) local.
124 # - Used for partition of solid part.
125 zload1_local = solid_width/2 - load_length/2          # (
    before moving part)
126 zload2_local = solid_width/2 + load_length/2          # (
    before moving part)
127
128
129 # Load positions in global Z.
130 # - 3-4 Used for partition on shell part.
131 # - 1-4 Used for creating surfaces for load from service
    vehicle.
132 zload1 = z_centre_axle_loads - axle_dist/2 - load_length/2
     # = zload1_local + zpoint1 (after moving part)
133 zload2 = z_centre_axle_loads - axle_dist/2 + load_length/2
     # = zload2_local + zpoint2 (after moving part)
134 zload3 = z_centre_axle_loads + axle_dist/2 - load_length/2
135 zload4 = z_centre_axle_loads + axle_dist/2 + load_length/2
136
137
138
139 # ================ FIXED VARIABLES (mesh) =================
140 # Coordinates (global) used for partition lines on shell
```

```
    around solid part
# - Used when seed edge
X_limits_solid = np.array([xpoint1, xpoint2])          #
    Global position of solid part
Z_limits_solid = np.array([zpoint1, zpoint2])          #
    Global position of solid part
Z_solid_local = np.array([0.0, solid_width])           #
    Local position of solid part
Y_limits = np.array([tfb/2, H-tfu/2])                  #
    Boundaries for shell in Y
Z_limits = np.array([0.0, length])                     #
    Boundaries for shell in Z

# Z positions (global) partition of shell-part, used for
    seeding
zpartition1 = zpoint1-300
zpartition2 = zpoint2+300


# Shell: seed size
seed_size_x = cc/6                                     # Top
    plate flanges in X
seed_size_x_bot = cc/4                                 #
    Bottom plate flanges in X
seed_size_y = H/5                                      # Web
    in Y
seed_size_global = seed_size_x                         #
    Global seed for rest of the edges

seed_size_initial_z1 = 3*seed_size_x                   #
    Width in Z - Coarser mesh Finer mesh
if 100.0 % seed_size_initial_z1 == 0:
    seed_size_z1 = 3*seed_size_x
else:
    seed_size_z1 = 100.0 / math.ceil(100.0/
        seed_size_initial_z1)

seed_size_initial_z2 = 2*seed_size_x                   #
    Width in Z - Finer mesh
if 100-0 % seed_size_initial_z2 == 0:
    seed_size_z2 = 2*seed_size_x
else:
    seed_size_z2 = 100.0 / math.ceil(100.0/
        seed_size_initial_z2)


```

```
172 # Solid: seed size
173 seed_size_solid_flange_top_x = (cc-tw-2*r)/6          # Top
       plate flanges in X
174 seed_size_solid_flange_bot_x = (cc-tw-2*r)/4          #
       Bottom plate flanges in X
175 seed_size_solid_web_top = (tw + 2*r)/2               # Part
        of face plate above web in face plate in X
176 seed_size_solid_web = (H-tfu-tfb-2*r)/5              # Web
       in Y
177 seed_size_solid_top_y = tfu/2                        # Top
       face plate through thickness in Y
178 seed_size_solid_bot_y = tfb/2                        #
       Bottom face plate through thickness in Y
179 seed_size_solid_radius = (2*math.pi*r/4)/4          #
       Length of corner/x
180 seed_size_solid_global = seed_size_solid_flange_top_x #
       Global seed for rest of the edges
181
182 seed_size_solid_initial_z = 2*seed_size_solid_flange_top_x
       # Width in Z
183 if 25.0 % seed_size_solid_initial_z == 0:
184     seed_size_solid_z = 2*seed_size_solid_flange_top_x
185 else:
186     seed_size_solid_z = 25.0 / math.ceil(25.0/
          seed_size_solid_initial_z)
187
188
189 # Solid FINER: seed size
190 seed_size_solid_flange_top_x_finer = (cc-tw-2*r)/24  # Top
       and bottom plate flanges in X
191 seed_size_solid_web_top_finer = (tw + 2*r)/8         # Part
        of face plate above web in face plate in X
192 seed_size_solid_web_thickness_finer = tw/4           # Web
       thickness in X
193 seed_size_solid_web_finer = (H-tfu-tfb-2*r)/20       # Web
       in Y
194 seed_size_solid_thickness_top_finer = tfu/4          # Top
       face plate through thickness in Y
195 seed_size_solid_thickness_bot_finer = tfb/4          #
       Bottom face plate through thickness in Y
196 seed_size_solid_radius_finer = (2*math.pi*r/4)/12    #
       Length of corner/x
197 seed_size_solid_z_finer = 50.0/10.0                  #
       Width in Z
198
199
```

```
200
201 # ============ COORDINATES - ONE SHELL PROFILE ============
202 # Coordinates for the first I-profile (straight lines)
203
204 # Straight lines - Length 6
205 xshell = np.array([0.0, cc, cc/2, cc/2, 0.0, cc])
206 yshell = np.array([tfb/2, tfb/2, tfb/2, H-tfu/2, H-tfu/2, H
       -tfu/2])
207 #  5--------4--------6
208 #           |
209 #           |
210 #           |
211 #  1--------3--------2
212
213 # X positions on each part of the flange
214 xedge = np.array([cc/4, 3*cc/4])
215
216
217
218 # ============ COORDINATES - ONE SOLID PROFILE ============
219 # Coordinates for the first I-profile (straight lines,
       corners, vertical end lines)
220
221 # Straight lines - Length 16
222 xlines = np.array([0.0, cc/2-tw/2-r, cc/2-tw/2, cc/2-tw/2,
       cc/2-tw/2-r, 0.0, 0.0, cc,
223                    cc, cc/2+tw/2+r, cc/2+tw/2, cc/2+tw/2,
                          cc/2+tw/2+r, cc, cc, 0.0 ])
224 ylines = np.array([tfb, tfb, tfb+r, H-tfu-r, H-tfu, H-tfu,
       H, H, H-tfu, H-tfu, H-tfu-r, tfb+r, tfb, tfb, 0.0, 0.0])
225
226 # Corners (centre, start, end) - Length 12
227 xradius = np.array([cc/2-tw/2-r, cc/2-tw/2-r, cc/2-tw/2, cc
       /2-tw/2-r, cc/2-tw/2, cc/2-tw/2-r,
228                     cc/2+tw/2+r, cc/2+tw/2+r, cc/2+tw/2, cc
                        /2+tw/2+r, cc/2+tw/2, cc/2+tw/2+r])
229 yradius = np.array([tfb+r, tfb, tfb+r, H-tfu-r, H-tfu-r, H-
       tfu, H-tfu-r, H-tfu, H-tfu-r, tfb+r, tfb+r, tfb])
230
231 # Vertical end lines - Length 4
232 xstart = np.array([0.0, 0.0, 0.0, 0.0])
233 ystart = np.array([0.0, tfb, H-tfu, H])
234 xstop = np.array(xstart + cc*solidprofiles)
235 ystop = ystart
236
237 #      Straight lines              Corners
```

```
#  6,7 ---------------- 8,9        ----------------
#         4,5 | 10,11              4,5,6 | 7,8,9
#             |                           |
#         2,3 | 12,13              1,2,3 | 10,11,12
# 1,16 ---------------- 14,15      ----------------
# Vertical end lines
#  3,4 ----------------
#             |
#             |
#             |
#  1,2 ----------------



# ==================== MODULES IN ABAQUS ====================
# ====================== PART (shell) ======================
# Create shell model
Mdb()

# PART 1: Bottom face plate
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)

for n in range(int(nprofiles)):
    xshell1 = np.array(xshell + cc*n)
    s.Line(point1=(xshell1[0], yshell[0]), point2=(xshell1
        [1], yshell[1]))

p = mdb.models['Model-1'].Part(name='Part-1',
    dimensionality=THREE_D,
    type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['Part-1']
p.BaseShellExtrude(sketch=s, depth=length)
s.unsetPrimaryObject()
del mdb.models['Model-1'].sketches['__profile__']


# PART 2: Web-core
s = mdb.models['Model-1'].ConstrainedSketch(name='
    __profile__',
    sheetSize=200.0)
```

```
279 g, v, d, c = s.geometry, s.vertices, s.dimensions, s.
    constraints
280 s.setPrimaryObject(option=STANDALONE)
281
282 for n in range(int(nprofiles)):
283     xshell1 = np.array(xshell + cc*n)
284     s.Line(point1=(xshell1[2], yshell[2]), point2=(xshell1
        [3], yshell[3]))
285
286 p = mdb.models['Model-1'].Part(name='Part-2',
    dimensionality=THREE_D,
287     type=DEFORMABLE_BODY)
288 p = mdb.models['Model-1'].parts['Part-2']
289 p.BaseShellExtrude(sketch=s, depth=length)
290 s.unsetPrimaryObject()
291 del mdb.models['Model-1'].sketches['__profile__']
292
293
294 # PART 3: Top face plate
295 s = mdb.models['Model-1'].ConstrainedSketch(name='
    __profile__',
296     sheetSize=200.0)
297 g, v, d, c = s.geometry, s.vertices, s.dimensions, s.
    constraints
298 s.setPrimaryObject(option=STANDALONE)
299
300 for n in range(int(nprofiles)):
301     xshell1 = np.array(xshell + cc*n)
302     s.Line(point1=(xshell1[4], yshell[4]), point2=(xshell1
        [5], yshell[5]))
303
304 p = mdb.models['Model-1'].Part(name='Part-3',
    dimensionality=THREE_D,
305     type=DEFORMABLE_BODY)
306 p = mdb.models['Model-1'].parts['Part-3']
307 p.BaseShellExtrude(sketch=s, depth=length)
308 s.unsetPrimaryObject()
309 del mdb.models['Model-1'].sketches['__profile__']
310
311
312
313 # ================== PROPERTY (shell) ====================
314 # Define material
315 mdb.models['Model-1'].Material(name='Alu')
316 mdb.models['Model-1'].materials['Alu'].Density(table=((
    density, ), ))
```

```
317 mdb.models['Model-1'].materials['Alu'].Elastic(table=((
       youngs_modulus, poissons_ratio), ))
318
319
320 # Create section (assign material, thickness)
321 mdb.models['Model-1'].HomogeneousShellSection(name='Section
       -1',
322     preIntegrate=OFF, material='Alu', thicknessType=UNIFORM
          , thickness=tfb,
323     thicknessField='', nodalThicknessField='', idealization
          =NO_IDEALIZATION,
324     poissonDefinition=DEFAULT, thicknessModulus=None,
          temperature=GRADIENT,
325     useDensity=OFF, integrationRule=SIMPSON, numIntPts=5) #
          5 as default for Simpson integration rule
326
327 mdb.models['Model-1'].HomogeneousShellSection(name='Section
       -2',
328     preIntegrate=OFF, material='Alu', thicknessType=UNIFORM
          , thickness=tw,
329     thicknessField='', nodalThicknessField='', idealization
          =NO_IDEALIZATION,
330     poissonDefinition=DEFAULT, thicknessModulus=None,
          temperature=GRADIENT,
331     useDensity=OFF, integrationRule=SIMPSON, numIntPts=5) #
          5
332
333 mdb.models['Model-1'].HomogeneousShellSection(name='Section
       -3',
334     preIntegrate=OFF, material='Alu', thicknessType=UNIFORM
          , thickness=tfu,
335     thicknessField='', nodalThicknessField='', idealization
          =NO_IDEALIZATION,
336     poissonDefinition=DEFAULT, thicknessModulus=None,
          temperature=GRADIENT,
337     useDensity=OFF, integrationRule=SIMPSON, numIntPts=5) #
          5
338
339
340 # Create set and assign section
341 # Part 1 / Set-1 / Section-1: Bottom face plate
342 faces = []
343 p = mdb.models['Model-1'].parts['Part-1']
344 f = p.faces
345 for n in range(int(nprofiles)):
346     faces.append(f.findAt(((cc/4+cc*n, tfb/2, length/4), ))
```

```
        )
347
348 region = p.Set(faces=faces, name='Set-1')
349 p.SectionAssignment(region=region, sectionName='Section-1',
        offset=0.0,
350     offsetType=MIDDLE_SURFACE, offsetField='',
351     thicknessAssignment=FROM_SECTION)
352
353 # Part 2 / Set-2 / Section-2: Web-core
354 faces = []
355 p = mdb.models['Model-1'].parts['Part-2']
356 f = p.faces
357 for n in range(int(nprofiles)):
358     faces.append(f.findAt(((cc/2+cc*n, H/2, length/4), )))
359
360 region = p.Set(faces=faces, name='Set-2')
361 p.SectionAssignment(region=region, sectionName='Section-2',
        offset=0.0,
362     offsetType=MIDDLE_SURFACE, offsetField='',
363     thicknessAssignment=FROM_SECTION)
364
365 # Part 3 / Set-3 / Section-3: Top face plate
366 faces = []
367 p = mdb.models['Model-1'].parts['Part-3']
368 f = p.faces
369 for n in range(int(nprofiles)):
370     faces.append(f.findAt(((cc/4+cc*n, H-tfu/2, length/4),
          )))
371
372 region = p.Set(faces=faces, name='Set-3')
373 p.SectionAssignment(region=region, sectionName='Section-3',
        offset=0.0,
374     offsetType=MIDDLE_SURFACE, offsetField='',
375     thicknessAssignment=FROM_SECTION)
376
377
378
379 # ==================== ASSEMBLY (shell) ====================
380 # Assembly of instances to PART-4
381 a = mdb.models['Model-1'].rootAssembly
382 a.DatumCsysByDefault(CARTESIAN)
383
384 p = mdb.models['Model-1'].parts['Part-1']
385 a.Instance(name='Part-1-1', part=p, dependent=OFF)
386
387 p = mdb.models['Model-1'].parts['Part-2']
```

```
388 a.Instance(name='Part-2-1', part=p, dependent=OFF)
389
390 p = mdb.models['Model-1'].parts['Part-3']
391 a.Instance(name='Part-3-1', part=p, dependent=OFF)
392
393 a.InstanceFromBooleanMerge(name='Part-4', instances=(a.
      instances['Part-1-1'],
394     a.instances['Part-2-1'], a.instances['Part-3-1'], ),
          originalInstances=SUPPRESS, domain=GEOMETRY)
395
396
397
398 # =============== PART (shell - partition) ================
399 # Help lines to smoother create sets/surfaces/mesh etc.
400 p = mdb.models['Model-1'].parts['Part-4']
401 v1, e, f, d = p.vertices, p.edges, p.faces, p.datums
402
403
404 # ------ Partition parallell to extrusion for seed edges
      around cut ------
405 for n in range(len(X_limits_solid)):
406     for m in range(len(Y_limits)):
407         pickedFaces1 = f.findAt(((X_limits_solid[n],
              Y_limits[m], cc), ))
408         p.PartitionFaceByShortestPath(point1=v1.findAt(
              coordinates=(X_limits_solid[n], Y_limits[m],
              Z_limits[0])),
409                                       point2=v1.findAt(
                                                 coordinates=(
                                                 X_limits_solid[n],
                                                  Y_limits[m],
                                                 Z_limits[1])),
410                                       faces=pickedFaces1)
411
412
413 # ------ Partition perpendicular to extrusion for seed edges
      around cut ------
414 # Datum planes for partition
415 p.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset
      =zpartition1)
416 p.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset
      =zpartition2)
417
418 # Execute partition
419 datumplane_ID_shell = p.datums.keys()
420 # Execute partition horizontal (First two index)
```

```
421 for n in datumplane_ID_shell[0:1+1]:
422     pickedFaces = f.getByBoundingBox(0.0, tfb/2, 0.0, cc*
        nprofiles, H-tfu/2, length)
423     p.PartitionFaceByDatumPlane(datumPlane=d[n], faces=
        pickedFaces)
424
425
426 # ------ Partition for creating load surfaces for service
    vehicle ------
427 # Create orientation for partition
428 t = p.MakeSketchTransform(sketchPlane=f.findAt(coordinates
    =(cc/4, H-tfu/2, length/4),
429     normal=(0.0, -1.0, 0.0)), sketchUpEdge=e.findAt(
        coordinates=(0.0, H-tfu/2, length/4)),
430     sketchPlaneSide=SIDE1, sketchOrientation=LEFT, origin
        =(0.0, 0.0, 0.0))
431
432
433 # Create geometry for partition
434 s = mdb.models['Model-1'].ConstrainedSketch(name='
    __profile__',
435     sheetSize=7207.01, gridSpacing=180.17, transform=t)
436 g, v, d, c = s.geometry, s.vertices, s.dimensions, s.
    constraints
437 s.setPrimaryObject(option=SUPERIMPOSE)
438 p.projectReferencesOntoSketch(sketch=s, filter=
    COPLANAR_EDGES)
439
440 # NOTE1! that the coordinates order when drawing is:
441 # LOCAL (x,y) = GLOBAL (X,Z), drawing from bottom surface
442
443 # Perpendicular to extrusion (vertical, constant x = X)
444 s.Line(point1=(xpoint1, zload3), point2=(xpoint2, zload3))
445 s.Line(point1=(xpoint1, zload4), point2=(xpoint2, zload4))
446
447 # Parallell to extrusions (horizontal, constant y = Z)
448 for n in range(load_step):
449     load_pos_start = xpoint1 + load_start + n*step_length
450     load_pos_stop = load_pos_start - load_length
451     s.Line(point1=(load_pos_start, zload3), point2=(
        load_pos_start, zload4)) # Start
452     s.Line(point1=(load_pos_stop, zload3), point2=(
        load_pos_stop, zload4))   # Stop
453
454
455 # Execute partition
```

```
456 pickedFaces = f.getByBoundingBox(xpoint1, H-tfu/2, 0.0,
       xpoint2, H-tfu/2, length)
457 p.PartitionFaceBySketch(sketchUpEdge=e.findAt(coordinates
       =(0.0, H-tfu/2, length/4)),
458     faces=pickedFaces, sketchOrientation=LEFT, sketch=s)
459 s.unsetPrimaryObject()
460 del mdb.models['Model-1'].sketches['__profile__']
461
462
463 a.regenerate()
464
465
466
467 # ================ PART (Cut shell model) =================
468 # Create orientation for cut out
469 p = mdb.models['Model-1'].parts['Part-4']
470 f, e = p.faces, p.edges
471
472 t = p.MakeSketchTransform(sketchPlane=f.findAt(coordinates
       =(cc/4, H-tfu/2, length/4),
473     normal=(0.0, -1.0, 0.0)), sketchUpEdge=e.findAt(
           coordinates=(0.0, H-tfu/2, length/4)),
474     sketchPlaneSide=SIDE1, sketchOrientation=LEFT, origin
           =(0.0, 0.0, 0.0))
475
476
477 # Create geometry for cut out
478 s = mdb.models['Model-1'].ConstrainedSketch(name='_profile_
       ',
479     sheetSize=7011.41, gridSpacing=175.28, transform=t)
480 g, v, d, c = s.geometry, s.vertices, s.dimensions, s.
       constraints
481 s.setPrimaryObject(option=SUPERIMPOSE)
482 p.projectReferencesOntoSketch(sketch=s, filter=
       COPLANAR_EDGES)
483
484 # NOTE1! that the coordinates order when drawing is:
485 # LOCAL (x,y) = GLOBAL (X,Z), drawing from bottom surface
486 s.rectangle(point1=(xpoint1, zpoint1), point2=(xpoint2,
       zpoint2))
487
488
489 # Execute cut out
490 p.CutExtrude(sketchPlane=f.findAt(coordinates=(cc/4, H-tfu
       /2, length/4),
491     normal=(0.0, -1.0, 0.0)), sketchUpEdge=e.findAt(
```

```
        coordinates=(0.0, H-tfu/2, length/4)),
492     sketchPlaneSide=SIDE1, sketchOrientation=LEFT, sketch=s
        , flipExtrudeDirection=ON)
493 s.unsetPrimaryObject()
494 del mdb.models['Model-1'].sketches['_profile_']
495
496
497 a.regenerate()
498
499
500
501 # ====================== PART (solid) ======================
502 # Create solid model - PART-5
503 s1 = mdb.models['Model-1'].ConstrainedSketch(name='
    __profile__', sheetSize=200.0)
504 g, v, d, c = s1.geometry, s1.vertices, s1.dimensions, s1.
    constraints
505 s1.setPrimaryObject(option=STANDALONE)
506
507 # Create part - Loop over all lines to create the geometry
    of n number of profiles
508 for n in range(int(solidprofiles)):
509     xline = np.array(xlines + cc*n)
510     xrad = np.array(xradius + cc*n)
511     for i in range(0,len(xlines),2):
512         s1.Line(point1=(xline[i], ylines[i]), point2=(xline
            [i+1], ylines[i+1]))
513
514     for i in range(0,len(xradius),3):
515         s1.ArcByCenterEnds(center=(xrad[i], yradius[i]),
            point1=(xrad[i+1], yradius[i+1]),
516         point2=(xrad[i+2], yradius[i+2]), direction=
            COUNTERCLOCKWISE)
517
518 for i in range(0,len(xstart),2):
519     s1.Line(point1=(xstart[i], ystart[i]), point2=(xstart[i
        +1], ystart[i+1]))
520     s1.Line(point1=(xstop[i], ystop[i]), point2=(xstop[i
        +1], ystop[i+1]))
521
522 # Define part and type of elements
523 p = mdb.models['Model-1'].Part(name='Part-5',
    dimensionality=THREE_D, type=DEFORMABLE_BODY)
524 p = mdb.models['Model-1'].parts['Part-5']
525 p.BaseSolidExtrude(sketch=s1, depth=solid_width)
526 s1.unsetPrimaryObject()
```

```
527 del mdb.models['Model-1'].sketches['__profile__']
528
529
530
531 # ================ PART (solid - partition) ================
532 p = mdb.models['Model-1'].parts['Part-5']
533 f, e, c, d = p.faces, p.edges, p.cells, p.datums
534
535
536 # ------ Partition for creating load surfaces for service
       vehicle ------
537 # Create orientation for partition
538 t = p.MakeSketchTransform(sketchPlane=f.findAt(coordinates
       =(cc/4, H, solid_width/2)),
539     sketchUpEdge=e.findAt(coordinates=(cc/4, H, solid_width
          )), sketchPlaneSide=SIDE1,
540     origin=(0.0, 0.0, 0.0))
541
542 # Create geometry for partition
543 s1 = mdb.models['Model-1'].ConstrainedSketch(name='
       __profile__', sheetSize=1510.19, gridSpacing=37.75,
       transform=t)
544 s1.setPrimaryObject(option=SUPERIMPOSE)
545 p.projectReferencesOntoSketch(sketch=s1, filter=
       COPLANAR_EDGES)
546
547 # NOTE1! that the coordinates order when drawing is:
548 # LOCAL (x,y) = GLOBAL (Z,X), drawing on top surface
549 # NOTE2! Not yet moved, drawing from (0,0,0)
550
551 # Perpendicular to extrusion (vertical, constant x = Z )
552 s1.Line(point1=(zload1_local, 0.0), point2=(zload1_local,
       cc*solidprofiles))
553 s1.Line(point1=(zload2_local, 0.0), point2=(zload2_local,
       cc*solidprofiles))
554
555 # Parallell to extrusions (horizontal, constant y = X)
556 for n in range(load_step):
557     load_pos_start = load_start + n*step_length
558     load_pos_stop = load_pos_start - load_length
559     s1.Line(point1=(zload1_local, load_pos_start), point2=(
          zload2_local, load_pos_start)) # Start
560     s1.Line(point1=(zload1_local, load_pos_stop), point2=(
          zload2_local, load_pos_stop))   # Stop
561
562 # Execute partition
```

```
563 pickedFaces = f.findAt(((cc/4, H, solid_width/2), ))
564 p.PartitionFaceBySketch(sketchUpEdge=e.findAt(coordinates=(
       cc/4, H, solid_width)), faces=pickedFaces, sketch=s1)
565 s1.unsetPrimaryObject()
566 del mdb.models['Model-1'].sketches['__profile__']
567
568
569 # ----------------- Partition for seeding edges -----------------
570 # Datum planes for partition horizontal
571 p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset
       =tfb+r)
572 p.DatumPlaneByPrincipalPlane(principalPlane=XZPLANE, offset
       =H-tfu-r)
573
574 # Datum plane perpendicular to extrusion
575 p.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset
       =solid_width/2 - 25.0)
576 p.DatumPlaneByPrincipalPlane(principalPlane=XYPLANE, offset
       =solid_width/2 + 25.0)
577
578 # Datum planes for partition vertical
579 for n in range(int(solidprofiles)):
580     p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE,
          offset=(cc-tw)/2 - r +cc*n)
581     p.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE,
          offset=(cc+tw)/2 + r + cc*n)
582
583
584 # Execute partition
585 datumplane_ID = p.datums.keys()
586
587 # Execute partition horizontal (First two index)
588 for n in datumplane_ID[0:1+1]:
589     pickedCells = c.findAt(((cc/2, H, solid_width/2), ))
590     p.PartitionCellByDatumPlane(datumPlane=d[n], cells=
          pickedCells)
591
592 # Execute partition perpendicular (Second two index)
593 for n in datumplane_ID[1+1:1+3]:
594     pickedCells = c.getByBoundingBox(0.0, 0.0, 0.0, cc*
          solidprofiles, H, solid_width)
595     p.PartitionCellByDatumPlane(datumPlane=d[n], cells=
          pickedCells)
596
597 # Execute partition vertical (From index 5 and forth)
598 for n in datumplane_ID[1+3:]:
```

```
599     pickedCells = c.getByBoundingBox(0.0, 0.0, 0.0, cc*
            solidprofiles, H, solid_width)
600     p.PartitionCellByDatumPane(datumPlane=d[n], cells=
            pickedCells)
601
602
603
604 # ==================== PROPERTY (solid) ====================
605 # Create section (assign material)
606 mdb.models['Model-1'].HomogeneousSolidSection(name='Section
        -4', material='Alu', thickness=None)
607
608 # Create set
609 p = mdb.models['Model-1'].parts['Part-5']
610 c = p.cells
611 cells = c.getByBoundingBox(0.0, 0.0, 0.0, cc*solidprofiles,
        H, solid_width)
612 region = p.Set(cells=cells, name='Set-1')
613
614 # Assign section
615 p.SectionAssignment(region=region, sectionName='Section-4',
        offset=0.0,
616     offsetType=MIDDLE_SURFACE, offsetField='',
            thicknessAssignment=FROM_SECTION)
617
618
619
620 # ==================== ASSEMBLY (solid) ====================
621 # Assembly of solid to Part-5-1
622 a = mdb.models['Model-1'].rootAssembly
623 p = mdb.models['Model-1'].parts['Part-5']
624 a.Instance(name='Part-5-1', part=p, dependent=ON) # OFF for
        independent
625
626
627
628 # ================ ASSEMBLY (shell + solid) ================
629 # Move solid part into position
630 a = mdb.models['Model-1'].rootAssembly
631 a.translate(instanceList=('Part-5-1', ), vector=(xpoint1,
        0.0, zpoint1))
632
633
634
635 # ============== INTERACTION (shell + solid) ==============
636 # Tie constraint
```

```
637 a = mdb.models['Model-1'].rootAssembly
638
639 # NODES OF SHELL AT CUT ZONE (main=region1)
640 e1 = a.instances['Part-4-1'].edges
641 side1Edges1 = e1.getByBoundingBox(xpoint1, tfb/2, zpoint1,
       xpoint2, H-tfu/2, zpoint2)
642 region1 = a.Surface(side1Edges=side1Edges1, name='Shell-
       surf-cut')
643
644 # SURFACES OF SOLID PART (secondary=region2)
645 surface1 = []
646 s1 = a.instances['Part-5-1'].faces
647
648 # Surfaces perpendicular to extrusion
649 for n in range(int(solidprofiles)):
650     xedges1 = xpoint1 + cc/2 + cc*n
651     xedges2 = xpoint1 + xedge + cc*n
652     for m in range(len(Z_limits_solid)):
653         surface1.append(s1.findAt(((xedges1, tfb/2,
              Z_limits_solid[m]), ), ((xedges1, H/2,
              Z_limits_solid[m]), ),
654                                   ((xedges1, H-tfu/2,
                                        Z_limits_solid[m]), )
                                        ))
655     for o in range(len(xedge)):
656         surface1.append(s1.findAt(((xedges2[o], tfb/2,
              Z_limits_solid[0]), ), ((xedges2[o], H-tfu/2,
              Z_limits_solid[0]), ),
657                                   ((xedges2[o], tfb/2,
                                        Z_limits_solid[1]), ),
                                         ((xedges2[o], H-tfu
                                        /2, Z_limits_solid[1])
                                        , )))
658
659 # Surfaces parallell to extrusion
660 z_tie = np.array([zpoint1+solid_width/4, zpoint1+
       solid_width/2, zpoint1+3*solid_width/4])
661 for n in range(len(z_tie)):
662     surface1.append(s1.findAt(((xpoint1, tfb/4, z_tie[n]),
          ), ((xpoint1, H-tfu/4, z_tie[n]), )))
663     surface1.append(s1.findAt(((xpoint2, tfb/4, z_tie[n]),
          ), ((xpoint2, H-tfu/4, z_tie[n]), )))
664 region2 = a.Surface(side1Faces=surface1, name='Solid-surf-
       cut')
665
666
```

```
667 # TIE
668
669 mdb.models['Model-1'].Tie(name='Constraint-1', master=
       region1, slave=region2,
670     positionToleranceMethod=COMPUTED, adjust=OFF,
           tieRotations=OFF,
671     thickness=ON)
672
673
674 # ========================== STEP ==========================
675 # Create step: default settings - (1:number of load_step)
676 # Creates Field output and history output
677 mdb.models['Model-1'].StaticLinearPerturbationStep(name='
       Step-1', previous='Initial')
678
679
680
681 # ====================== LOAD (load) =======================
682 # Load from: Service vehicle, self-weight and surface
       coating
683 a = mdb.models['Model-1'].rootAssembly
684 s1 = a.instances['Part-5-1'].faces
685 s2 = a.instances['Part-4-1'].faces
686
687
688 # ---------------------- Service vehicle ----------------------
689 # Point load from each axle distributed over a square
       surface 200x200
690 # Create surface from partition for every load step - (0:
       number of load_step)
691 for n in range(load_step):
692     load_pos_start = xpoint1 + load_start + n*step_length
693     load_pos_end = load_pos_start - load_length
694     side1Faces1 = s1.getByBoundingBox(load_pos_end, H,
           zload1, load_pos_start, H, zload2)             #
           Solid
695     side2Faces2 = s2.getByBoundingBox(load_pos_end, H-tfu
           /2, zload3, load_pos_start, H-tfu/2, zload4) # Shell
696     a.Surface(side1Faces=side1Faces1, side2Faces=
           side2Faces2, name=LoadSurf-vehicle-+str(n))
697
698 # Add load on surface - (0:number of load_step)
699 for n in range(load_step):
700     region = a.surfaces[LoadSurf-vehicle-+str(n)]
701     mdb.models['Model-1'].Pressure(name=Load-vehicle-+str(n)
           , createStepName='Step-1', region=region,
```

```
702         distributionType=UNIFORM, field='', magnitude=
               axle_load, amplitude=UNSET)
703
704
705 # ------------------------ Gravity load ------------------------
706 mdb.models['Model-1'].Gravity(name='Load-gravity',
       createStepName='Step-1',
707         comp2=-gravity, distributionType=UNIFORM, field='')
708
709
710 # ----------------------- Surface coating -----------------------
711 # Create a surface of entire top face plate
712 side1Faces1 = s1.getByBoundingBox(xpoint1, H, zpoint1,
       xpoint2, H, zpoint2)          # Solid
713 side2Faces2 = s2.getByBoundingBox(0.0, H-tfu/2, 0.0, cc*
       nprofiles, H-tfu/2, length) # Shell
714 a.Surface(side1Faces=side1Faces1, side2Faces=side2Faces2,
       name='LoadSurf-coating')
715
716 # Add load on top face plate
717 region = a.surfaces['LoadSurf-coating']
718 mdb.models['Model-1'].Pressure(name='Load-coating',
       createStepName='Step-1',
719         region=region, distributionType=UNIFORM, field='',
               magnitude=coating)
720
721
722 # ======================== LOAD (BC) ========================
723 # SIMPLY SUPPORTED BC (perpendicular to extrusions)
724 # NOTE1! BC propagates from previous step, only adding on
       first step.
725 a = mdb.models['Model-1'].rootAssembly
726 e1 = a.instances['Part-4-1'].edges
727 v1 = a.instances['Part-4-1'].vertices
728
729 # Pick EDGES and assign BC (bottom/top, Z=0.0 and length) -
       BC fixed in Y = 0
730 edges1 = []
731 for n in range(int(nprofiles)):
732     xedge1 = xedge + cc*n
733     xedge2 = cc/2 + cc*n
734     edges1.append(e1.findAt(((xedge2, H/2, 0.0), ), ((
           xedge2, H/2, length), )))
735     for m in range(len(xedge)):
736         edges1.append(e1.findAt(((xedge1[m], tfb/2, 0.0), )
               , ((xedge1[m], H-tfu/2, 0.0), ),
```

```
737                                     ((xedge1[m], tfb/2, length)
                                           , ), ((xedge1[m], H-tfu
                                           /2, length), )))
738
739 region1 = a.Set(edges=edges1, name='BC-Set-1')
740 mdb.models['Model-1'].DisplacementBC(name='BC-y',
       createStepName='Step-1',
741     region=region1, u1=UNSET, u2=0.0, u3=UNSET, ur1=UNSET,
           ur2=UNSET, ur3=UNSET,
742     amplitude=UNSET, fixed=OFF, distributionType=UNIFORM,
           fieldName='', localCsys=None)
743
744 # Pick POINTS in x and assign BC - BC fixed rotation in z
745 verts1 = v1.findAt(((cc/2, tfb/2, 0.0), ), ((cc*nprofiles-
       cc/2, tfb/2, 0.0), ))
746 region = a.Set(vertices=verts1, name='BC-Set-2')
747 mdb.models['Model-1'].DisplacementBC(name='BC-point1',
       createStepName='Step-1',
748     region=region, u1=UNSET, u2=UNSET, u3=UNSET, ur1=UNSET,
            ur2=UNSET, ur3=0.0,
749     amplitude=UNSET, fixed=OFF, distributionType=UNIFORM,
           fieldName='', localCsys=None)
750
751 # Pick POINTS in z and assign BC - BC fixed in x
752 verts1 = v1.findAt(((cc/2, tfb/2, 0.0), ), ((cc/2, tfb/2,
       length), ))
753 region = a.Set(vertices=verts1, name='BC-Set-3')
754 mdb.models['Model-1'].DisplacementBC(name='BC-point2',
       createStepName='Step-1',
755     region=region, u1=0.0, u2=UNSET, u3=UNSET, ur1=UNSET,
           ur2=UNSET, ur3=UNSET,
756     amplitude=UNSET, fixed=OFF, distributionType=UNIFORM,
           fieldName='', localCsys=None)
757
758
759
760 # ==================== LOAD (Load case) ====================
761 for n in range(load_step):
762     mdb.models['Model-1'].steps['Step-1'].LoadCase(name=
           LoadCase-+str(n+1),
763     loads=(('Load-coating', factor_coating), ('Load-gravity
           ', factor_self_weight), (Load-vehicle-+str(n),
           factor_traffic_load)),
764     boundaryConditions=(('BC-point1', 1.0), ('BC-point2',
           1.0), ('BC-y', 1.0)),
765     includeActiveBaseStateBC=OFF)
```

```
766
767
768
769 # ==================== MESH (shell) ====================
770 p = mdb.models['Model-1'].parts['Part-4']
771 e = p.edges
772 f = p.faces
773
774 # Mesh control (All faces)
775 pickedRegions = f.getByBoundingBox(0.0, tfb/2, 0.0, cc*
       nprofiles, H-tfu/2, length)
776 p.setMeshControls(regions=pickedRegions, elemShape=QUAD,
       technique=STRUCTURED)
777
778 # Assign element type (All faces)
779 # S8R:  An 8-node doubly curved thick shell, reduced
       integration.
780 elemType1 = mesh.ElemType(elemCode=S8R, elemLibrary=
       STANDARD)
781 elemType2 = mesh.ElemType(elemCode=STRI65, elemLibrary=
       STANDARD)
782
783 pickedRegions1 =(pickedRegions, )
784 p.setElementType(regions=pickedRegions1, elemTypes=(
       elemType1, elemType2))
785
786
787 # SEED EDGES
788 # ------------------------ X-direction ------------------------
789 # Seed: TOP short edge FLANGES (Z=0.0 and length) / (Z=
       zpoint1 and zpoint2)
790 for n in range(int(nprofiles)):
791     xedge1 = xedge + cc*n
792     for m in range(len(xedge)):
793         pickedEdges1 = e.findAt(((xedge1[m], H-tfu/2, 0.0),
                ), ((xedge1[m], H-tfu/2, zpartition1), ),
794                                 ((xedge1[m], H-tfu/2,
                                    zpartition2), ), ((
                                    xedge1[m], H-tfu/2,
                                    length), ))
795         p.seedEdgeBySize(edges=pickedEdges1, size=
               seed_size_x, deviationFactor=0.1, constraint=
               FINER) #FINER
796
797 for n in range(int(solidprofiles)):
798     xedge1 = xedge + xpoint1 + cc*n
```

```
799     for m in range(len(xedge)):
800         pickedEdges1 = e.findAt(((xedge1[m], H-tfu/2,
               zpoint1), ), ((xedge1[m], H-tfu/2, zpoint2), ))
801         p.seedEdgeBySize(edges=pickedEdges1, size=
               seed_size_x, deviationFactor=0.1, constraint=
               FINER) #FINER
802
803 # Seed: BOTTOM short edge FLANGES (Z=0.0 and length) / (Z=
       zpoint1 and zpoint2)
804 for n in range(int(nprofiles)):
805     xedge1 = xedge + cc*n
806     for m in range(len(xedge)):
807         pickedEdges1 = e.findAt(((xedge1[m], tfb/2, 0.0), )
               , ((xedge1[m], tfb/2, zpartition1), ),
808                                 ((xedge1[m], tfb/2,
                                       zpartition2), ), ((
                                       xedge1[m], tfb/2, length
                                       ), ))
809         p.seedEdgeBySize(edges=pickedEdges1, size=
               seed_size_x_bot, deviationFactor=0.1, constraint
               =FINER) #FINER
810
811 for n in range(int(solidprofiles)):
812     xedge1 = xedge + xpoint1 + cc*n
813     for m in range(len(xedge)):
814         pickedEdges1 = e.findAt(((xedge1[m], tfb/2, zpoint1
               ), ), ((xedge1[m], tfb/2, zpoint2), ))
815         p.seedEdgeBySize(edges=pickedEdges1, size=
               seed_size_x_bot, deviationFactor=0.1, constraint
               =FINER) #FINER
816
817
818 # ------------------------ Y-direction ------------------------
819 # Seed: Short edge WEB (Z=0.0 and length) / (Z=zpoint1 and
       zpoint2)
820 for n in range(int(nprofiles)):
821     pickedEdges1 = e.findAt(((cc/2 + cc*n, H/2, 0.0), ), ((
           cc/2 + cc*n, H/2, zpartition1), ),
822                             ((cc/2 + cc*n, H/2, zpartition2
                                   ), ), ((cc/2 + cc*n, H/2,
                                   length), ))
823     p.seedEdgeBySize(edges=pickedEdges1, size=seed_size_y,
           deviationFactor=0.1, constraint=FINER) #FINER
824
825 for n in range(int(solidprofiles)):
826     pickedEdges2 = e.findAt(((cc/2 + xpoint1 + cc*n, H/2,
```

```
        zpoint1), ), ((cc/2 + xpoint1 + cc*n, H/2, zpoint2),
         ))
827     p.seedEdgeBySize(edges=pickedEdges2, size=seed_size_y,
        deviationFactor=0.1, constraint=FINER) #FINER
828
829
830
831 # ------------------------- Z-direction -------------------------
832 # Seed: LONG edge above/below web
833 # All bottom/top over nprofiles and solidprofiles away from
    cut zone - Coarser mesh
834 for n in range(int(nprofiles)):
835     pickedEdges1 = e.findAt(((cc/2 + cc*n, tfb/2, 10.0), ),
        ((cc/2 + cc*n, H-tfu/2, 10.0), ),
836                             ((cc/2 + cc*n, tfb/2,
                                zpartition2+10), ), ((cc/2 +
                                 cc*n, H-tfu/2, zpartition2
                                +10), ))
837     p.seedEdgeBySize(edges=pickedEdges1, size=seed_size_z1,
        deviationFactor=0.1, constraint=FINER)
838
839 for n in range(int(solidprofiles)):
840     pickedEdges2 = e.findAt(((cc/2 + xpoint1 + cc*n,
        Y_limits[1], zload3+10.0), ),
841                             ((cc/2 + xpoint1 + cc*n,
                                Y_limits[1], length-10.0), )
                                )
842     p.seedEdgeBySize(edges=pickedEdges2, size=seed_size_z1,
        deviationFactor=0.1, constraint=FINER)
843
844 # All bottom/top over nprofiles and solidprofiles around
    cut zone - Finer mesh
845 for n in range(int(nprofiles)):
846     pickedEdges1 = e.findAt(((cc/2 + cc*n, tfb/2,
        zpartition1+10), ), ((cc/2 + cc*n, H-tfu/2,
        zpartition1+10), ))
847     p.seedEdgeBySize(edges=pickedEdges1, size=seed_size_z2,
        deviationFactor=0.1, constraint=FINER)
848
849 for n in range(int(solidprofiles)):
850     for m in range(len(Y_limits)):
851         pickedEdges2 = e.findAt(((cc/2 + xpoint1 + cc*n,
            Y_limits[m], zpartition2-10), ))
852         p.seedEdgeBySize(edges=pickedEdges2, size=
            seed_size_z2, deviationFactor=0.1, constraint=
            FINER)
```

```
853
854
855 # Seed: Long END edges
856 # - Coarser mesh
857 for m in range(len(Y_limits)):
858     pickedEdges4 = e.findAt(((0.0, Y_limits[m], 10.0), ),
           ((cc*nprofiles, Y_limits[m], 10.0),),
859                             ((0.0, Y_limits[m], zpartition2
                                  +10), ), ((cc*nprofiles,
                                  Y_limits[m], zpartition2+10)
                                  ,))
860     p.seedEdgeBySize(edges=pickedEdges4, size=seed_size_z1,
            deviationFactor=0.1, constraint=FINER)
861
862 # - Finer mesh
863 for m in range(len(Y_limits)):
864     pickedEdges4 = e.findAt(((0.0, Y_limits[m], zpartition1
           +10), ), ((cc*nprofiles, Y_limits[m], zpartition1
           +10),))
865     p.seedEdgeBySize(edges=pickedEdges4, size=seed_size_z2,
            deviationFactor=0.1, constraint=FINER)
866
867
868 # Seed: Long partition lines parallell to extrusion
869 # - Coarser mesh
870 for n in range(len(X_limits_solid)):
871     for m in range(len(Y_limits)):
872         pickedEdges5 = e.findAt(((X_limits_solid[n],
               Y_limits[m], 10.0), ), ((X_limits_solid[n],
               Y_limits[m], zpartition2+10), ),
873                                 ((X_limits_solid[n],
                                    Y_limits[m], zload3+10),
                                     ), ((X_limits_solid[n],
                                     Y_limits[m], length-10)
                                    , ))
874         p.seedEdgeBySize(edges=pickedEdges5, size=
               seed_size_z1, deviationFactor=0.1, constraint=
               FINER)
875
876 # - Finer mesh
877 for n in range(len(X_limits_solid)):
878     for m in range(len(Y_limits)):
879         pickedEdges5 = e.findAt(((X_limits_solid[n],
               Y_limits[m], zpartition1+10), ), ((
               X_limits_solid[n], Y_limits[m], zpoint1+10), ),
880                                 ((X_limits_solid[n],
```

```
                                        Y_limits[m], zpartition2
                                        -10), ))
881         p.seedEdgeBySize(edges=pickedEdges5, size=
               seed_size_z2, deviationFactor=0.1, constraint=
               FINER)
882
883
884 # ------------------------------ Global ------------------------------
885 p.seedPart(size=seed_size_global, deviationFactor=0.1,
       minSizeFactor=0.1)
886
887
888 # Generate mesh
889 p.generateMesh()
890 a.regenerate()
891
892
893
894 # ====================== MESH (solid) ======================
895 p = mdb.models['Model-1'].parts['Part-5']
896 c, e, d = p.cells, p.edges, p.datums
897
898 # ------------------------- Element type -------------------------
899 # Assign element type (All faces)
900 # C3D20R:  A 20-node quadratic brick, reduced integration.
901 elemType1 = mesh.ElemType(elemCode=C3D20R, elemLibrary=
       STANDARD)
902 elemType2 = mesh.ElemType(elemCode=C3D15, elemLibrary=
       STANDARD)
903 elemType3 = mesh.ElemType(elemCode=C3D10, elemLibrary=
       STANDARD)
904
905 cells = c.getByBoundingBox(0.0, 0.0, 0.0, cc*solidprofiles,
        H, solid_width)
906 pickedRegions =(cells, )
907 p.setElementType(regions=pickedRegions, elemTypes=(
       elemType1, elemType2, elemType3))
908
909
910 # ------------------------- Mesh control -------------------------
911 # Mesh control - elemShape=HEX, technique=AS IS (as default
       )
912 pickedRegions = c.findAt(((load_start+cc, tfb/2,
       solid_width/8), ), ((load_start+cc, tfb/2, solid_width
       /2), ), ((load_start+cc, tfb/2, 7*solid_width/8), ))
913 p.setMeshControls(regions=pickedRegions, technique=SWEEP,
```

```
    algorithm=ADVANCING_FRONT)
914
915
916 # SEED EDGES
917 # ------------------------ X-direction -------------------------
918 # Part of face plate above web - element size: half
    thickness
919 for n in range(int(solidprofiles)):
920     xedges1 = cc/2 + cc*n
921     for m in range(len(Z_solid_local)):
922         pickedEdges = e.findAt(((xedges1, 0.0,
              Z_solid_local[m]), ), ((xedges1, H,
              Z_solid_local[m]), ) )
923         p.seedEdgeBySize(edges=pickedEdges, size=
              seed_size_solid_web_top, deviationFactor=0.1,
              constraint=FINER)
924
925 # Seed: BOTTOM plate edges in X (0.0 and tfb)
926 for n in range(int(solidprofiles)):
927     xedge1 = xedge + cc*n
928     for m in range(len(xedge)):
929         for z in zpositions_solid:
930             pickedEdges1 = e.findAt(((xedge1[m], 0.0, z), )
                  , ((xedge1[m], tfb, z), ))
931             p.seedEdgeBySize(edges=pickedEdges1, size=
                  seed_size_solid_flange_bot_x,
                  deviationFactor=0.1, constraint=FINER)
932
933 # Seed: TOP plate edges in X (H and H-tfu) and top plate
    bottom skin in X (H-tfu))
934 for n in range(int(solidprofiles)):
935     xedge1 = xedge + cc*n
936     for m in range(len(xedge)):
937         pickedEdges1 = e.findAt(((xedge1[m], H-tfu, 0.0), )
              , ((xedge1[m], H, 0.0), ),
938                                 ((xedge1[m], H-tfu, solid_width), )
                                      , ((xedge1[m], H, solid_width),
                                      ),
939                                 ((xedge1[m], H-tfu, solid_width
                                      /2-25.0), ), ((xedge1[m], H-tfu,
                                       solid_width/2+25.0), ))
940         p.seedEdgeBySize(edges=pickedEdges1, size=
              seed_size_solid_flange_top_x, deviationFactor
              =0.1, constraint=FINER)
941
942 # Seed: TOP plate - face partition in X (H)
```

```
943 x_edges_coarse_seed_solid = []
944 x_edges_fine_seed_solid = []
945 for edge in p.edges:
946     for n in range(int(solidprofiles-1)):
947         for z in zpositions_solid_partition:
948             if edge.pointOn[0][2] == z and edge.pointOn
                   [0][1] == H and (cc*n + (cc/2+tw/2+r)) <
                   edge.pointOn[0][0] < ( cc*n + (3*cc/2-tw/2-r
                   )):
949                 x_edges_coarse_seed_solid.append(edge.
                       pointOn[0])
950             elif edge.pointOn[0][2] == z and edge.pointOn
                   [0][1] == H and 0.0 <= edge.pointOn[0][0] <
                   (cc/2-tw/2-r):
951                 x_edges_coarse_seed_solid.append(edge.
                       pointOn[0])
952             elif edge.pointOn[0][2] == z and edge.pointOn
                   [0][1] == H and (solidprofiles*cc-cc/2+tw/2+
                   r) <= edge.pointOn[0][0] < (cc*solidprofiles
                   ):
953                 x_edges_coarse_seed_solid.append(edge.
                       pointOn[0])
954     for z in zpositions_solid_top:
955         for y in ypositions_solid:
956             if edge.pointOn[0][2] == z and edge.pointOn
                   [0][1] == y and (load_start+tw/2+r) < edge.
                   pointOn[0][0] < (load_start+cc-tw/2-r):
957                 x_edges_fine_seed_solid.append(edge.
                       pointOn[0])
958
959
960 #p.Set(edges=p.edges.findAt(coordinates=
       x_edges_coarse_seed2), name='x_edges_coarse_seed-SET2')
961 p.seedEdgeBySize(edges=p.edges.findAt(coordinates=
       x_edges_coarse_seed_solid), size=
       seed_size_solid_flange_top_x, deviationFactor=0.1,
       constraint=FINER)
962
963
964
965 # ------------------------ Y-direction ------------------------
966 # Face plate - through thickness
967 # Bottom
968 for n in range(int(solidprofiles)):
969     xline1 = (cc-tw)/2 - r + cc*n
970     xline2 = (cc+tw)/2 + r + cc*n
```

```
971     for m in range(len(Z_solid_local)):
972         pickedEdges = e.findAt(((xline1, tfb/2,
               Z_solid_local[m]), ), ((xline2, tfb/2,
               Z_solid_local[m]), ))
973         p.seedEdgeBySize(edges=pickedEdges, size=
               seed_size_solid_bot_y, deviationFactor=0.1,
               constraint=FINER)
974
975 # Bottom - ends
976 pickedEdges = e.findAt(((0.0, tfb/2, 0.0), ), ((0.0, tfb/2,
        solid_width), ),
977                        ((cc*solidprofiles, tfb/2, 0.0), ),
                              ((cc*solidprofiles, tfb/2,
                              solid_width), ))
978 p.seedEdgeBySize(edges=pickedEdges, size=
       seed_size_solid_bot_y, deviationFactor=0.1, constraint=
       FINER)
979
980 # Top
981 for n in range(int(solidprofiles)):
982     xline1 = (cc-tw)/2 - r + cc*n
983     xline2 = (cc+tw)/2 + r + cc*n
984     for m in range(len(Z_solid_local)):
985         pickedEdges = e.findAt(((xline1, H-tfu/2,
               Z_solid_local[m]), ), ((xline2, H-tfu/2,
               Z_solid_local[m]), ))
986         p.seedEdgeBySize(edges=pickedEdges, size=
               seed_size_solid_top_y, deviationFactor=0.1,
               constraint=FINER)
987
988 # Top - ends
989 pickedEdges = e.findAt(((0.0, H-tfu/2, 0.0), ), ((0.0, H-
       tfu/2, solid_width), ),
990                        ((cc*solidprofiles, H-tfu/2, 0.0), )
                              , ((cc*solidprofiles, H-tfu/2,
                              solid_width), ))
991 p.seedEdgeBySize(edges=pickedEdges, size=
       seed_size_solid_top_y, deviationFactor=0.1, constraint=
       FINER)
992
993
994 # Web
995 for n in range(int(solidprofiles)):
996     xline1 = (cc-tw)/2 + cc*n
997     xline2 = (cc+tw)/2 + cc*n
998     for z in zpositions_solid:
```

```
999         pickedEdges = e.findAt(((xline1, H/2, z), ), ((
               xline2, H/2, z), ))
1000        p.seedEdgeBySize(edges=pickedEdges, size=
               seed_size_solid_web, deviationFactor=0.1,
               constraint=FINER)
1001
1002
1003 # ------------------------- Z-direction -------------------------
1004 # All lines in Z-direction (excl surface partition for
       service vehicle and ends)
1005 z_edges_coarse_seed_solid = []
1006 z_edges_fine_seed_solid = []
1007 for edge in p.edges:
1008     for y in ypositions_solid_z:
1009
1010         if 0.0 < edge.pointOn[0][2] < 100.0 and edge.
               pointOn[0][1] == y:
1011             z_edges_coarse_seed_solid.append(edge.pointOn
                   [0])
1012         elif 100.0 < edge.pointOn[0][2] < 175.0 and edge.
               pointOn[0][1] == y:
1013             z_edges_coarse_seed_solid.append(edge.pointOn
                   [0])
1014         elif 225.0 < edge.pointOn[0][2] < 300.0 and edge.
               pointOn[0][1] == y:
1015             z_edges_coarse_seed_solid.append(edge.pointOn
                   [0])
1016         elif 300.0 < edge.pointOn[0][2] < 400.0 and edge.
               pointOn[0][1] == y:
1017             z_edges_coarse_seed_solid.append(edge.pointOn
                   [0])
1018         elif 175.0 < edge.pointOn[0][2] < 225.0 and edge.
               pointOn[0][1] == y:
1019             z_edges_fine_seed_solid.append(edge.pointOn[0])
1020
1021 p.seedEdgeBySize(edges=p.edges.findAt(coordinates=
       z_edges_coarse_seed_solid), size=seed_size_solid_z,
       deviationFactor=0.1, constraint=FINER)
1022
1023
1024 # ------------------------- Radius -------------------------
1025 x_radius = (r-r*math.cos(math.pi/4))
1026 y_radius = (r-r*math.sin(math.pi/4))
1027
1028 for n in range(int(solidprofiles)):
1029     xline1 = (cc-tw)/2 - x_radius + cc*n
```

```
    xline2 = (cc+tw)/2 + x_radius + cc*n
    for z in zpositions_solid:
        pickedEdges = e.findAt(((xline1, tfb+y_radius, z),
           ), ((xline1, H-tfu-y_radius, z), ),
                               ((xline2, tfb+y_radius, z),
                                  ), ((xline2, H-tfu-
                                  y_radius, z), ))
        p.seedEdgeBySize(edges=pickedEdges, size=
           seed_size_solid_radius, deviationFactor=0.1,
           constraint=FINER)



# ---------------------------- FINE MESH ----------------------------
# X-DIRECTION
# Part of face plate above/below web
pickedEdges = e.findAt(((load_start+cc, 0.0, 0.0), ), ((
   load_start+cc, H, 0.0), ),
                       ((load_start+cc, 0.0, solid_width),
                          ), ((load_start+cc, H,
                          solid_width), ))
p.seedEdgeBySize(edges=pickedEdges, size=
   seed_size_solid_web_top_finer, deviationFactor=0.1,
   constraint=FINER)

# Plate edges + face partition in X
p.seedEdgeBySize(edges=p.edges.findAt(coordinates=
   x_edges_fine_seed_solid), size=
   seed_size_solid_flange_top_x_finer, deviationFactor=0.1,
    constraint=FINER)

# Top and bottom of web
pickedEdges = e.findAt(((load_start+cc, tfb+r, 0.0), ), ((
   load_start+cc, H-tfu-r, 0.0), ),
                       ((load_start+cc, tfb+r, solid_width)
                          , ), ((load_start+cc, H-tfu-r,
                          solid_width), ))
p.seedEdgeBySize(edges=pickedEdges, size=
   seed_size_solid_web_thickness_finer, deviationFactor
   =0.1, constraint=FINER)


# Y-DIRECTION
# Web
for n in range(len(zpositions_solid)):
    pickedEdges = e.findAt(((load_start+cc-tw/2, H/2,
```

```
        zpositions_solid[n]), ), ((load_start+cc+tw/2, H/2,
        zpositions_solid[n]), ))
1058    p.seedEdgeBySize(edges=pickedEdges, size=
        seed_size_solid_web_finer, deviationFactor=0.1,
        constraint=FINER)
1059
1060 # Face plate - through thickness
1061 pickedEdges = e.findAt(((load_start+cc+tw/2+r, tfb/2, 0.0),
    ), ((load_start+cc+tw/2+r, tfb/2, solid_width), ),
1062                        ((load_start+cc-tw/2-r, tfb/2, 0.0),
                            ), ((load_start+cc-tw/2-r, tfb
                           /2, solid_width), ))
1063 p.seedEdgeBySize(edges=pickedEdges, size=
    seed_size_solid_thickness_bot_finer, deviationFactor
    =0.1, constraint=FINER)
1064
1065 pickedEdges = e.findAt(((load_start+cc+tw/2+r, H-tfu/2,
    0.0), ), ((load_start+cc+tw/2+r, H-tfu/2, solid_width),
    ),
1066                        ((load_start+cc-tw/2-r, H-tfu/2,
                            0.0), ), ((load_start+cc-tw/2-r,
                            H-tfu/2, solid_width), ))
1067 p.seedEdgeBySize(edges=pickedEdges, size=
    seed_size_solid_thickness_top_finer, deviationFactor
    =0.1, constraint=FINER)
1068
1069
1070 # Z-DIRECTION
1071 # Everyting in z on finer part
1072 p.seedEdgeBySize(edges=p.edges.findAt(coordinates=
    z_edges_fine_seed_solid), size=seed_size_solid_z_finer,
    deviationFactor=0.1, constraint=FINER)
1073
1074
1075 # RADIUS
1076 for z in zpositions_solid:
1077     pickedEdges = e.findAt(((load_start+cc-tw/2-x_radius,
        tfb+y_radius, z), ), ((load_start+cc-tw/2-x_radius,
        H-tfu-y_radius, z), ),
1078                            ((load_start+cc+tw/2+x_radius,
                                tfb+y_radius, z), ), ((
                                load_start+cc+tw/2+x_radius,
                                H-tfu-y_radius, z), ))
1079     p.seedEdgeBySize(edges=pickedEdges, size=
        seed_size_solid_radius_finer, deviationFactor=0.1,
        constraint=FINER)
```

```
1080
1081
1082 # ------------------------ Generate mesh ------------------------
1083 p.generateMesh()
1084 a.regenerate()
1085
1086
1087
1088 # ====================== STEP (output) ======================
1089 # Adjusting field output requests with regard to set
1090 mdb.models['Model-1'].fieldOutputRequests['F-Output-1'].
        setValues(variables=('MISES', ),
1091      region=MODEL, exteriorOnly=OFF, sectionPoints=DEFAULT,
           rebar=EXCLUDE)
1092
1093 # Delete history outputs
1094 del mdb.models['Model-1'].historyOutputRequests['H-Output-1
        ']
1095
1096
1097
1098 # ========================== JOB ==========================
1099 # Create and submit job
1100 mdb.Job(name='Job-1', model='Model-1', description='', type
        =ANALYSIS,
1101      atTime=None, waitMinutes=0, waitHours=0, queue=None,
           memory=90,
1102      memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
1103      explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE,
           echoPrint=OFF,
1104      modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
           userSubroutine='',
1105      scratch='', resultsFormat=ODB, multiprocessingMode=
           DEFAULT, numCpus=n_cpus,
1106      numDomains=n_cpus, numGPUs=0)
1107
1108
1109 mdb.jobs['Job-1'].submit(consistencyChecking=OFF)
1110 mdb.jobs['Job-1'].waitForCompletion()
1111
1112
1113
1114 # =============== CREATE PATHS AND XY-DATA ===============
1115 # Create node-sets to be used for paths
1116 assembly = mdb.models['Model-1'].rootAssembly
1117 shell = assembly.instances['Part-4-1']
```

```
1118 solid = assembly.instances['Part-5-1']
1119
1120
1121 # ---------------------- Create node-lists ----------------------
1122 # Create list of node-coordinates for creating paths
1123 top_mid_solid = []
1124 bottom_mid_solid = []
1125 web_coord = []
1126 web_coordtop = []
1127 web_coordweb = []
1128 web_coordbot = []
1129 web_coordrad_bot = []
1130 web_coordrad_top = []
1131
1132 tol_coord = 5e-5
1133 for node in solid.nodes:
1134     # Top solid (1 cell)
1135     if (H-tol_coord) <= node.coordinates[1] <= (H+tol_coord
        ) and node.coordinates[2] == zpoint1+(zpoint2-
        zpoint1)/2 and (xpoint1+load_start+tw/2+r-tol_coord)
         <= node.coordinates[0] <= (xpoint1+load_start+cc+2*
        seed_size_solid_web_top_finer+tol_coord):
1136         top_mid_solid.append(node.coordinates)
1137     # Bottom solid (1 cell)
1138     elif (0.0-tol_coord) <= node.coordinates[1] <= (0.0+
        tol_coord) and node.coordinates[2] == zpoint1+(
        zpoint2-zpoint1)/2 and (xpoint1+load_start+tw/2+r-
        tol_coord) <= node.coordinates[0] <= (xpoint1+
        load_start+cc+2*seed_size_solid_web_top_finer+
        tol_coord):
1139         bottom_mid_solid.append(node.coordinates)
1140     # Web (inside solid) - (bottom, top, web, bottom radius
        , top radius)
1141     elif (tfb-tol_coord) <= node.coordinates[1] <= (tfb+
        tol_coord) and (xpoint1+load_start+tw/2+r) < node.
        coordinates[0] <= (xpoint1+load_start+cc-tw/2-r+
        tol_coord) and node.coordinates[2] == (zpoint1+(
        zpoint2-zpoint1)/2):
1142         web_coordbot.append(node.coordinates)
1143     elif (H-tfu-tol_coord) <= node.coordinates[1] <= (H-tfu
        +tol_coord) and (xpoint1+load_start+tw/2+r) < node.
        coordinates[0] <= (xpoint1+load_start+cc-tw/2-r+
        tol_coord) and node.coordinates[2] == (zpoint1+(
        zpoint2-zpoint1)/2):
1144         web_coordtop.append(node.coordinates)
1145     elif (tfb+r-tol_coord) <= node.coordinates[1] <= (H-tfu
```

```
        -r+tol_coord) and (xpoint1+load_start+cc-tw/2-
        tol_coord) <= node.coordinates[0] <= (xpoint1+
        load_start+cc-tw/2+tol_coord) and node.coordinates
        [2] == (zpoint1+(zpoint2-zpoint1)/2):
            web_coordweb.append(node.coordinates)
    elif (tfb) < node.coordinates[1] < (tfb+r) and (xpoint1
        +load_start+cc-tw/2-r-tol_coord) < node.coordinates
        [0] < (xpoint1+load_start+cc-tw/2+tol_coord) and
        node.coordinates[2] == (zpoint1+(zpoint2-zpoint1)/2)
        :
            web_coordrad_bot.append(node.coordinates)
    elif (H-tfu-r) < node.coordinates[1] < (H-tfu) and (
        xpoint1+load_start+cc-tw/2-r-tol_coord) < node.
        coordinates[0] < (xpoint1+load_start+cc-tw/2+
        tol_coord) and node.coordinates[2] == (zpoint1+(
        zpoint2-zpoint1)/2):
            web_coordrad_top.append(node.coordinates)


# ---------------------- Sort node-lists ------------------------
# Sort the list of coordinates
# Sorting key for paths on top and bottom surface of solid:
    xmin to xmax
top_mid_solid = sorted(top_mid_solid)
bottom_mid_solid = sorted(bottom_mid_solid)


# Sorting key for path Web-Radius-inside plate surface of
   solid: ymin to ymax / xmax to xmin
def sorting_order(coord):
    return coord[1], -coord[0]

web_coordbot = sorted(web_coordbot)
web_coordtop = sorted(web_coordtop, key=sorting_order)
web_coordweb = sorted(web_coordweb)
web_coordrad_bot = sorted(web_coordrad_bot)
web_coordrad_top = sorted(web_coordrad_top, key=
   sorting_order)

web_coordrad_bot_sorted = []
y_old = web_coordrad_bot[0][1]
for y in web_coordrad_bot:
    if y[1] >= y_old:
        y_old = y[1]
        web_coordrad_bot_sorted.append(y)
```

```
1177 web_coordrad_top_sorted = []
1178 x_old = web_coordrad_top[0][0]
1179 for x in web_coordrad_top:
1180     if x[0] <= x_old:
1181         x_old = x[0]
1182         web_coordrad_top_sorted.append(x)
1183
1184 web_coord = web_coordbot + web_coordrad_bot_sorted +
        web_coordweb + web_coordrad_top_sorted + web_coordtop
1185
1186
1187 # ------------------------- Create path -------------------------
1188 # open odb and create paths
1189 odb = session.openOdb('Job-1.odb')
1190
1191 session.viewports['Viewport: 1'].setValues(displayedObject=
        odb)
1192 session.viewports['Viewport: 1'].makeCurrent()
1193
1194 shell = odb.rootAssembly.instances['PART-4-1']
1195 solid = odb.rootAssembly.instances['PART-5-1']
1196
1197 session.Path(name='Solid-Top', type=POINT_LIST, expression=
        top_mid_solid)
1198 session.Path(name='Solid-Bottom', type=POINT_LIST,
        expression=bottom_mid_solid)
1199 session.Path(name='Web', type=POINT_LIST, expression=
        web_coord)
1200
1201
1202 # ------------------- Von Mises stresses -------------------
1203 # Find maximum Von Mises stress
1204 max_mises = []
1205 max_top = 0.0
1206 max_bot = 0.0
1207 max_web = 0.0
1208 for i in range(1, 6):
1209     session.viewports['Viewport: 1'].odbDisplay.setFrame(
            step=0, frame=i )
1210     # Top solid
1211     session.XYDataFromPath(name='Top_Solid'+str(i), path=
            session.paths['Solid-Top'],
1212         includeIntersections=False, projectOntoMesh=False,
                pathStyle=PATH_POINTS,
1213         numIntervals=10, projectionTolerance=0, shape=
                UNDEFORMED,
```

```
1214          labelType=TRUE_DISTANCE_X, removeDuplicateXYPairs=
                 True,
1215          includeAllElements=True)
1216      # Bottom solid
1217      session.XYDataFromPath(name='Bottom_Solid'+str(i), path
             =session.paths['Solid-Bottom'],
1218          includeIntersections=False, projectOntoMesh=False,
                 pathStyle=PATH_POINTS,
1219          numIntervals=10, projectionTolerance=0, shape=
                 UNDEFORMED,
1220          labelType=TRUE_DISTANCE_X, removeDuplicateXYPairs=
                 True,
1221          includeAllElements=True)
1222      # Web (inside solid)
1223      session.XYDataFromPath(name='Web'+str(i), path=session.
             paths['Web'],
1224          includeIntersections=False, projectOntoMesh=False,
                 pathStyle=PATH_POINTS,
1225          numIntervals=10, projectionTolerance=0, shape=
                 UNDEFORMED,
1226          labelType=TRUE_DISTANCE, removeDuplicateXYPairs=
                 True,
1227          includeAllElements=True)
1228      for xy_top in session.xyDataObjects['Top_Solid'+str(i)
             ].data:
1229          if xy_top[1] > max_top:
1230              max_top = xy_top[1]
1231      for xy_bot in session.xyDataObjects['Bottom_Solid'+str(
             i)].data:
1232          if xy_bot[1] > max_bot:
1233              max_bot = xy_bot[1]
1234      for xy_web in session.xyDataObjects['Web'+str(i)].data:
1235          if xy_web[1] > max_web:
1236              max_web = xy_web[1]
1237
1238  max_mises = [max([max_top, max_bot, max_web])]
1239
1240
1241  #session.viewports['Viewport: 1'].setValues(displayedObject
        =odb)
1242  #session.viewports['Viewport: 1'].makeCurrent()
1243
1244  # ------------------------------ .txt --------------------------------
1245  # Write maximum Von Mises stress to text file
1246  with open('results_mises.txt', 'w') as f:
1247          np.savetxt(f,max_mises)
```

```
1248
1249 odb.close()
```