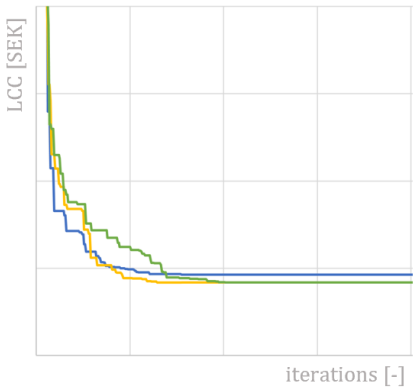
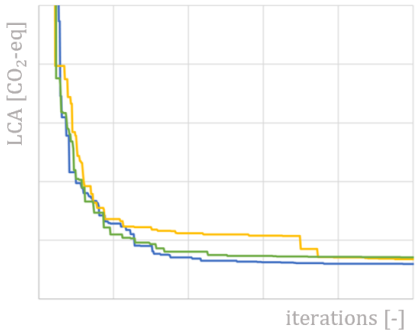
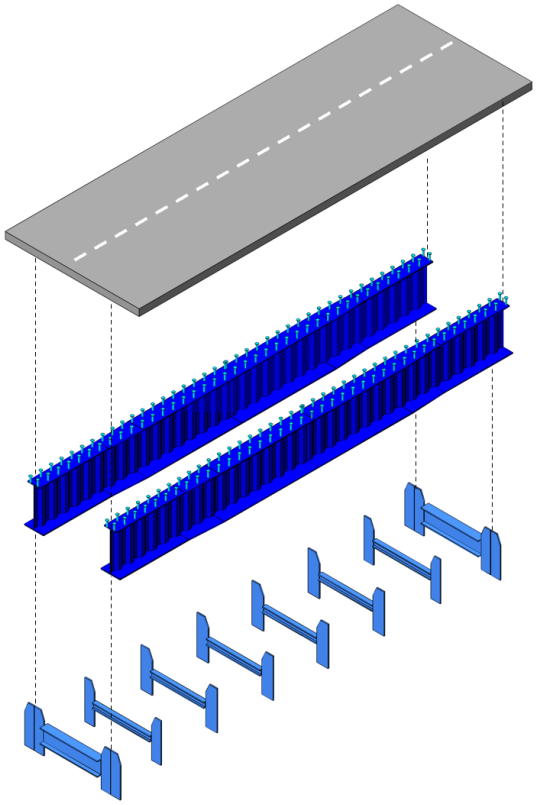






Design Optimization of Composite Road Bridges using Genetic Algorithms

Corrugated Web Stainless Steel Girders versus Flat Web Carbon Steel Girders

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

Cecilia Hallgren
Vilma Johansson

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's Thesis 2022

Design Optimization of Composite Road Bridges using Genetic Algorithms

CECILIA HALLGREN
VILMA JOHANSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Architecture and civil engineering
Division of Structural Engineering
Research Group of Lightweight Structures
Chalmers University of Technology
Gothenburg, Sweden 2022

Design Optimization of Composite Road Bridges using Genetic Algorithms
CECILIA HALLGREN
VILMA JOHANSSON

Supervisors: Professor Mohammad Al-Emrani, Chalmers, Industrial PhD Student Fatima Hlal, WSP/Chalmers and PhD Peter Nilsson Strand, WSP.
Examiner: Senior Lecturer Mozhdeh Amani, Chalmers.

Master's Thesis 2022
Department of Architecture and civil engineering
Division of Structural engineering
Research Group of Lightweight Structures
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover:
Left: Optimization convergence plots, see Appendix D, F and E.
Right: Exploded view of a composite road bridge with girders with corrugated webs, see Section 3.3.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Design Optimization of Composite Road Bridges using Genetic Algorithms
CECILIA HALLGREN
VILMA JOHANSSON
Department of Architecture and Building Technology
Chalmers University of Technology

Abstract

Steel bridges today are mostly constructed in traditional carbon steel, with corrosion as a common issue leading to high maintenance costs and a limited service life. This problem can be minimized by using stainless steel as it is less prone to corrode. However, stainless steel is more expensive leading to a higher investment cost. The aim of the thesis is therefore to develop a design optimization program of concrete-steel composite road bridges. The program, written in Python, is based on the Eurocode design procedure and optimized with the use of a genetic algorithm with the option to optimize towards minimum life cycle cost (LCC), environmental impact (LCA) or steel material usage. The life cycle performance tool used is developed in a parallel master's thesis by Nissan and Woldeyohannes (2022).

A case study is conducted where the program is set to redesign an existing bridge with flat web girders of carbon steel S355. The optimization is run for two concepts: girders of carbon steel grade S355 with flat webs and Duplex stainless steel girders with corrugated webs. The main study is on minimizing the life cycle cost, where sub-studies are performed to see how the result is affected by limitations of the web height, the amount of traffic, and the material price. Additionally, the program is run to minimize the environmental impact (CO_2 equivalents) and steel material mass to better understand the design choices of the program.

The results shows that the optimization program reduced the material use with 20 % for the carbon steel concept and with 30 % for stainless steel concept, compared to the original design. The first concept design also gave a life cycle cost reduction of 6 %. Comparing the two concepts with each other, the results showed that when allowing a web height of up to two meters, considering today's steel prices (60 SEK/kg for Duplex steel and 20 SEK/kg for S355), the stainless steel alternative is 9 % more expensive. However, when allowing a web height of up to three meters, the stainless steel alternative is 3 % cheaper. Further, with decreased material prices and increased amount of traffic, the stainless steel alternative could be proven even more competitive.

Lastly, an important conclusion is that the optimizations against mass and minimize life cycle cost gave similar results for the stainless steel alternative, showing it enough to optimize against mass. However, for the carbon steel alternative, where the maintenance aspect is highly important, a life cycle cost optimization is required.

Keywords: Composite Bridge, Corrugated Web, Stainless Steel, Optimization, Genetic Algorithms, Life Cycle cost (LCC), Life Cycle Assessment (LCA)

Designoptimering av samverkansvägbroar genom användning av genetiska algoritmer

CECILIA HALLGREN
VILMA JOHANSSON
Avdelningen för arkitektur och samhällsbyggnadsteknik
Chalmers Tekniska Högskola

Sammanfattning

Stålbroar är idag mestadels konstruerade i traditionellt kolstål, med korrosion som ett vanligt problem vilket leder till höga underhållskostnader och en begränsad livslängd. Detta problem kan minimeras genom att använda rostfritt stål, då det är mindre benäget att korrodera, dock till en högre investeringskostnad. Syftet med denna masteruppsats är därför att utveckla ett designoptimeringsprogram för samverkansbroar vägbroar av betong och stål. Programmet, skrivet i Python, är baserat på Eurocodes designprocedur och optimerat med användning av en genetisk algoritm med möjlighet att optimera mot minimal livscykelkostnad (LCC), miljöpåverkan (LCA) eller stålmaterialanvändning. Livscykelanalys-verktyget som används i optimeringsproceduren är utvecklat parallellt av Nissan and Woldeyohannes (2022).

En fallstudie genomförs där programmet ska omdesigna en befintlig samverkansvägbro gjord av kolstål S355 med platta liv. Optimeringen genomförs för två koncept: brobalkar med platta liv av S355 kolstål och brobalkar av Duplex rostfritt stål med korrugerade liv. Huvudstudien handlar om att minimera livscykelkostnaden, där delstudier genomförs för att se hur resultatet påverkas av begränsningar av livhöjd, trafikmängd och materialpris. Dessutom körs programmet med målet att minimera miljöpåverkan (CO_2 ekvivalenter) och stålmaterialmassa för att bättre förstå programmets designval.

Resultaten visar att optimeringsprogrammet minskade materialanvändningen med 20 % för kolstålkonceptet och 30 % för det rostfria stålkonceptet, jämfört med den ursprungliga designen. Designen av det första konceptet gav också en livscykelkostnadsreduktion med 6 %. En jämförelse mellan de två koncepten visade att när man tillåter en livhöjd på upp till två meter, med dagens stålpriser (60 kr/kg för Duplex och 20 kr/kg för S355), så är det rostfria alternativet 9 % dyrare. Tillåter man dock en livhöjd på upp till tre meter är alternativet av rostfritt stål 3 % billigare. Vidare så visade det rostfria konceptet mer sig konkurrenskraftigt med minskade materialpriser och ökad trafikmängd.

Slutligen så visade studien att optimeringarna mot massa och livscykelkostnad gav liknande resultat för det rostfria stålalternativet, vilket visar att det är tillräckligt att optimera mot massan. För alternativet av kolstål, där underhållsaspekten är viktig för den total kostnaden, krävs dock en optimering mot livscykelkostnad.

Keywords: Samverkansbro, Korrugerade balkliv, Rostfritt stål, Optimering, Genetiska algoritmer, Livscykelkostnad, Livscykelanalys

Acknowledgements

The thesis is part of an ongoing industrial research project conducted within the BBT research project 'Sustainable Bridges' which is funded by Trafikverket. It was executed with support from the department Bridge and Hydraulic at WSP, Gothenburg and the department of Architecture and Civil Engineering at Chalmers University of Technology. During the work of the thesis, we have gotten guidance and feedback from several people of which we would like to thank.

To Professor Mohammad Al-Emrani and PhD Student Fatima Hlal, we are very grateful for all meetings and discussions, for sharing your knowledge and for your enthusiasm as well as encouraging support throughout the whole project. Additionally, Fatima, your time and patience when guiding us through the design process of composite bridges has been crucial for the success of the thesis.

We would also like to say thank you to PhD Peter Nilsson Strand for you helpful input regarding the initial stages of designing the parametric program as well as providing us with documents and information from WSP's practice to help us develop our calculations and case study. Further, we like to thank Senior Lecturer Mozhdeh Amani for helpful feedback on the design process and the report writing.

Also, to the parallel master's thesis authors providing us with a Life Cycle Performance tool, Ashur Nissan and Yohannes Woldeyohannes, we are very grateful and would not have received such interesting results without you.

Lastly, to the colleagues at WSP, thank you for creating such a welcoming and happy working environment for us both in Halmstad and in Gothenburg. Me, Vilma, would also like to especially thank Louise Vibe, Bridge Designer at WSP, for the valuable discussions, for providing us with additional data and information regarding the case study, as well as for your support and encouragement.

Cecilia Hallgren and Vilma Johansson, Gothenburg, June 2022

Nomenclature

Below is the glossary, roman letters and greek letters that have been used throughout the thesis.

Glossary

Composite member	A structural member with components of concrete and of structural or cold-formed steel, interconnected by shear connection so as to limit the longitudinal slip between concrete and steel and the separation of one component from the other.
Construction Phase	The physical process of building and all other associated activities such as landscaping, refurbishing, site clearance, and demolition.
Design Optimization	An engineering design methodology using a mathematical formulation of a design problem to support selection of the optimal design among many alternatives.
Eurocode	The collection of the European standards stating the technical rules of the structural design of whole structures and component products of both a traditional and an innovative nature.
Fatigue	The process of initiation and propagation of cracks through a structural part due to action of fluctuating stress.
Genetic algorithm	An optimization algorithm inspired by the evolution of nature, mimicking two primary processes; natural selection and reproduction.
Krav brobyggande	A document with requirements developed by Trafikverket used for the design of bridges.

LCA	*Life Cycle Assessment* is a methodology for assessing environmental impacts associated with all the stages of the life cycle of a commercial product, process, or service.
LCC	*Life Cycle Cost* is the total cost of ownership over the life of an asset and is commonly referred to as "cradle to grave" costs.
Load Model	Vehicle traffic may differ between bridges depending on its composition, its density, its conditions, the extreme likely weights of vehicles and their axle loads, and, if relevant, the influence of road signs restricting carrying capacity. These differences should be taken into account through the use of load models suited to the location of a bridge.
Python	A computer programming language often used to build websites and software, automate tasks, and conduct data analysis.
Service Phase	The phase of the project related to the management of the contracts.
Serviceable Limit State	The state of design beyond which a structural system loses operationally its serviceability for the actual service load that the structure is subjected to.
TSFS	*Transportstyrelsens föreskrifter och allmänna råd om tillämpning av eurokoder* is a supplementary document containing provisions on how to use Eurocode and the national annex but also when to use alternative calculation methods. It is developed by Transportstyrelsen.
Ultimate Limit State	The design for the safety of a structure and its users by limiting the stress that materials experience.

Roman Letters

Material:

$E_{c,eff}$	Effective Modulus of Elasticity [MPa]
E_{cm}	Elastic Modulus of Concrete [MPa]
E_s	Elastic Modulus of Steel [GPa]

$E_{sf,c}$	Secant Modulus of the Compression Flange [GPa]
$E_{sf,t}$	Secant Modulus of the Tension Flange [GPa]
f_{ck}	Characteristic Compressive Strength for Concrete [MPa]
f_{cm}	Mean Value of the Compressive Strength for Concrete [MPa]
$f_{ctk,0.05}$	Characteristic Tensile Strength 5 % of Concrete [MPa]
$f_{ctk,0.95}$	Characteristic Tensile Strength 95 % of Concrete [MPa]
f_{ctm}	Mean Value of the Tensile Strength for Concrete [MPa]
f_{sk}	Characteristic Compressive Strength for Concrete [MPa]
f_u	Ultimate strength of Steel [MPa]
f_y	Yield Strength of Steel [MPa]
Geometry:	
A_c	Concrete Cross-Section Area [mm^2]
$A_{c,eff}$	Effective Area of Concrete Section [mm^2]
$A_{r,min}$	Minimum Reinforcement Area in Longitudinal Direction [mm^2]
A_{si}	Gross cross-sectional steel area [mm^2]
$A_{si,eff}$	Effective Area of Steel Parts in Cross-sectional Class 4 [mm^2]
B	Width of Bridge [mm]
B_s	C-C Distance between Main Girders [mm]
C	Distance between Crossbeams [mm]
I	Moment of Inertia [mm^4]
I_z	Moment of Inertia [mm^4]
L	Span Length [mm]
W_{eff}	Effective Sectional Modulus [mm^3]
W_{el}	Elastic Sectional Modulus [mm^3]
W_{pl}	Plastic Sectional Modulus [mm^3]
a_1	Length of the horizontal panel in the corrugated web [mm]
a_2	Length of the inclined panel in the corrugated web [mm]
a_3	Depth of the corrugation [mm]
a_4	Horizontal length of the inclined panel in the corrugated web [mm]
b_0	Distance Between Shear Studs [mm]
$b_{c,eff}$	Effective Width of the Concrete Section [mm]

b_{fo}	Width of the top flange [mm]
$b_{fo,eff}$	Effective Width of the Compressed Flange [mm]
b_{fu}	Width of the bottom flange [mm]
d_{tot}	Total Height of the Bridge including eventual Crash Barriers and Vehicles. Used in Calculations of the Contributing Wind Force [mm]
h_c	Height of Concrete Deck [mm]
h_w	Height of web [mm]
r_c	Additional Length of Corrugation Compared to Flat Webs [mm]
t_{fo}	Thickness of the top flange [mm]
t_{fu}	Thickness of the bottom flange [mm]
t_w	Thickness of web [mm]
w	Width of Traffic Lane [mm]

Loads:

F_{cs}	Force Contribution from Shrinkage [N/mm]
F_{temp}	Force Contribution from Temperature Action [N]
F_w	Force Contribution from Wind Load [N/mm]
$G_{k,j}$	Characteristic value of the self-weight [N/mm]
H_i	Horizontal Force due to Unintended Inclination used in Design of Crossbeams [N]
M_d	Design Moment [Nmm]
P	Other Permanent Loads [N or N/mm]
Q_{ik}	Point Load from Traffic Load Model 1 [N/m]
$Q_{k,1}$	Main Variable Loads [N or N/mm]
$Q_{k,acc}$	Characteristic Acceleration/Braking Load [N]
$Q_{k,i}$	Main Variable Loads [N or N/mm]
c_{fx}	Force Coeffictient used in Calculations of the wind load [-]
m	Number of Braced Elements used in Design of Crossbeams [-]
q_{ik}	Distributed Load from Traffic Load Model 1 [N/m]
q_p	Characteristic Velocity Pressure []
v_b	Reference Wind Velocity [m/s]
w	Width of one Traffic Lane used during Calculations from Traffic Load Contribution [mm]

Variables:

k_T	Buckling Shear Coefficient used for Calculation of Shear Buckling
n_0	Short-Term Modular Ratio [-]
n_L	Modular Ratio [-]
$n_{L,sc}$	Modular Ratio for shrinkage [-]

Design Strength:

$M_{B,Rd}$	Moment Capacity [Nmm]
M_{Ed}	Moment Resistance [Nmm]
$M_{f,Rd}$	Moment Capacity Contribution from Flanges [Nmm]
$M_{pl,Rd}$	Plastic Moment Capacity [Nmm]
N_{Ed}	Axial Resistance [N]
N_{Rd}	Axial Resistance [N]
P_{Rd}	Shear Capacity of Shear Studs [N]
V_{Ed}	Shear Resistance [N]
$V_{f,Rd}$	Shear Capacity Contribution from Flange [N]
V_{Rd}	Shear Capacity [N]
$V_{w,Rd}$	Shear Capacity Contribution from Web [N]

Greek Letters

α_i	Principal Stress in Weld [MPa]
$\alpha_\perp$	Stress Perpendicular in Weld [MPa]
α	Inclination of the Corrugation [Degrees]
α_{LT}	Imperfection Factor used for Calculations of the Lateral-Torsional buckling [-]
α_{Qi}	Adjustment Factors for Traffic Load Model 1 [-]
α_{qi}	Adjustment Factors for Traffic Load Model 1 [-]
α_{ss}	Coefficient of Linear Expansion for Stainless Steel [-]
α_c	Coefficient of Linear Expansion for Concrete [-]
γ_{Mi}	Partial Safety Factor used to Determine the Design Value of the Steel Capacity [-]
γ_d	Partial Safety Factor used in the Load Combination depending on Safety Class [-]

γ_c	Safety Factor used for Calculations of Design Resistance of Concrete [-]
γ_i	Safety Factor used for Load Combination [-]
$\Delta T_{N,sho}$	Maximum Temperature Component for Shortening [$^\circ C$]
$\Delta T_{N,ext}$	Maximum Temperature Component for Extension [$^\circ C$]
$\Delta T_{c,s}$	Temperature Difference between different Components [$^\circ C$]
$\Delta\sigma_E$	Stress Contribution from Considered Loads in Fatigue Limit State [MPa]
$\Delta\sigma_C$	Detail Category in Fatigue Limit State [MPa]
ϵ_{cc}	Creep Deformation [-]
ϵ_{cd}	Drying Shrinkage Strain [-]
ϵ_{ca}	Autogenous Shrinkage Strain [-]
ϵ_{cs}	Total Shrinkage Strain [-]
ϵ_{temp}	Temperature Strain [-]
λ_{LT}	Slenderness Parameter used for calculations of the Lateral-Torsional buckling [-]
λ	λ-factor used for Calculation of Fatigue Design [-]
ν	Poission's Ratio [-]
ξ	Reduction Factor used in Load Combination
ρ_s	Density of Steel [kg/m^3] or [kN/m^3]
σ_c	Constant Compressive Concrete Stress [MPa]
$\sigma_{f.SLS}$	Stress in Flange during Serviceable Limit State [MPa]
$\tau_{cr,g}$	Critical Shear Stress for Global Buckling [MPa]
$\tau_{cr,l}$	Critical Shear Stress for Local Buckling [MPa]
$\varphi(t, t_0)$	Creep Coefficient [-]
χ_{LT}	Reduction Factor due to Lateral-Torsional Buckling [-]
χ_i	Factor Regulating Variable Loads in Load Combination [-]
ψ_L	Additional Creep Factor [-]

Contents

List of Figures

List of Tables

1

Introduction

One common issue with steel bridges is corrosion resulting in limited service life and high maintenance costs, especially in high traffic areas. The problem could be minimized by using stainless steel as it is less prone to corrosion than traditionally used carbon steel. On the other hand, stainless steel is more expensive than carbon steel, resulting in high investment cost. High material utilization is therefore of great interest, which could be achieved with the use of optimization algorithms.

1.1 Background

The thesis is part of an ongoing industrial research project conducted within the BBT research project 'Sustainable Bridges' which is funded by Trafikverket. It is executed with support from WSP, Gothenburg and the department of Architecture and Civil Engineering at Chalmers University of Technology. It is a continuation of the master's theses *Design of Composite Steel-Concrete Bridges using Stainless Steel Girders with Corrugated Webs* (Henrysson and Yman, 2020) and *Design of Continuous Composite Road Bridges-Bridge girders with corrugated webs in stainless steel* (Steffner and Öman, 2021). The thesis by Henrysson and Yman (2020) studied the applicability and efficiency of replacing flat web carbon steel girders in composite bridges with stainless steel girders with corrugated webs. They concluded that even though there were some issues regarding how well the design procedure of Eurocode suits girders with corrugated webs, at least 20-30 % material savings could be proven.

The thesis by Steffner and Öman (2021) further investigated the design of composite bridges using corrugated webs and stainless steel, this time for continuous bridges instead of simply supported. The results showed material savings of up to 19 % when replacing the girders with flat web of carbon steel with stainless corrugated web girders. When comparing the total cost, both theses presented lower costs for the alternative with stainless steel corrugated web even though the resulting investment cost was still slightly higher for that alternative. Based on these conclusions, this thesis will further investigate the optimization of composite bridge girders in stainless steel with corrugated webs as well as flat web girders of carbon steel in order to be able to have a fair comparison between the total life cycle cost and environmental impact of the two design solutions.

1.2 Aim

The master's thesis aims to develop a program that can be used to optimize the design of a steel-concrete composite bridge in terms of life cycle cost and environmental impact. The result is to be used to compare traditional carbon steel girders with flat webs with alternatives made of stainless steel and corrugated webs.

1.3 Limitations

To narrow down the project, a few limitations are listed:

- The investigated bridges are composite steel-concrete road bridges with a bridge deck in concrete and twin girders made of steel. However, only the steel cross-section of the bridge will be subject to the optimization. Neither will the thesis study the bridge substructure.
- The focus will be on simply supported road bridges.
- For the corrugated web, the considered shape is trapezoidal since it is the most commonly used.
- The carbon steel material considered in the design is S355 and S460, however, the program is designed so that more steel grades easily can be added.
- The stainless steel material considered in the design is duplex stainless steel of grade 1.4162, however, the program is designed so that more steel grades easily can be added.
- For the girders, only welded I-sections are considered.
- For the crossbeams, only hot-rolled sections are considered.
- The design procedure is according to the parts of Eurocode listed in Table 3.1.
- The structural analysis is linear elastic.

1.4 Approach

During the beginning of the project, a literature study is conducted on composite bridges, stainless steel, corrugated webs, optimization, genetic algorithms and Life Cycle Cost/Life Cycle Assessment to gain good background knowledge about the various parts of the project. Further, different established design procedures are studied and a parametric design template is created using Python scripting. The design output is linked to a Life Cycle Performance tool developed by a parallel master's thesis project by Nissan and Woldeyohannes (2022) and a Genetic Algorithm is implemented to optimize the design of the bridge girders. A case study is used to

display the results and finally, the optimized design is assessed in terms of life cycle cost and life cycle assessment. By changing the available material and geometrical data, the results can be used to compare different optimized designs, for example stainless steel with carbon steel and flat webs with corrugated webs.

2
Literature Study

The literature study is conducted to give an introduction to the different topics of the project. The study includes books and manuals on the fundamentals as well as research articles and reports. The main focus is within the limits of the thesis, however for a wider understanding the study is extended on selected areas. This chapter includes additional design studies outside of the rules of Eurocode for a wider perspective, while the design procedure used and described in Chapter 3 strictly follows Eurocode. Similarly, this chapter gives an introduction to optimization and Genetic Algorithms while Chapter 4 explains the specific method used in detail. Lastly, the concepts of Life Cycle Cost (LCC) and Life Cycle Assessment (LCA) are explained in this chapter, while the Life Cycle Performance tool included in the optimization are described in detail in the parallel master thesis *Life cycle assessment, LCA, and life cycle cost, LCC, for composite bridges - Corrugated web stainless steel girders vs. flat web carbon steel girders* by Nissan and Woldeyohannes (2022).

2.1 Steel and Concrete Composite Bridges

There are different types of composite bridges; box girder bridges and twin or multi-girder bridges (Sarraf et al., 2013). However, in this report only simply supported twin-girder road bridges will be considered and a 2D drawing of the cross-section is visualized in Figure 2.1.

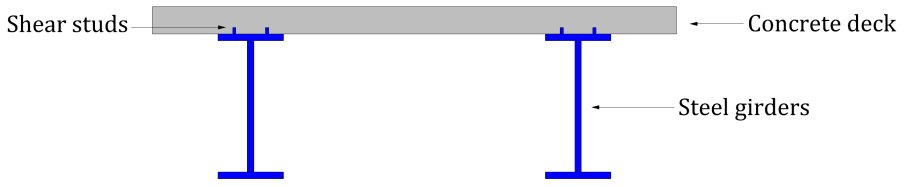
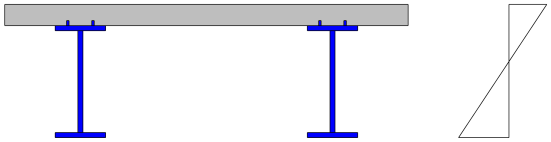


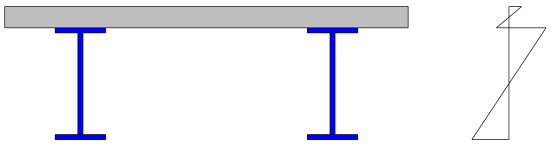
Figure 2.1: Cross-section of a twin girder composite bridge.

Composite bridges are mainly used because of their material effectiveness when concrete and steel are exposed to the stresses they have the highest strength in, namely concrete in compression and steel in tension (Sarraf et al., 2013). See Figure 2.2a. Figure 2.2b visualizes a bridge without composite action as well as the important role of the shear connectors in creating the composite action between the steel and concrete. Hällmark (2018) also states that another advantage of using composite

bridges of this kind is the possibility of prefabrication to minimize the production time. The prefabricated girders can be lifted or launched in place and the framework used for the casting of the concrete can be carried directly by the girders meaning no need for temporary constructions. This is, in particular, beneficial at sites where minimal impact on the underlying activity is required.



(a) With composite action.

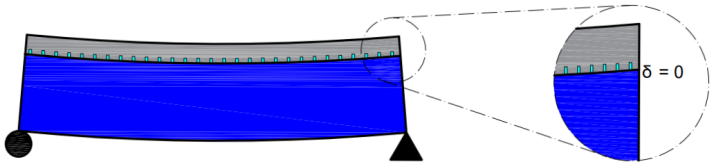


(b) Without composite action.

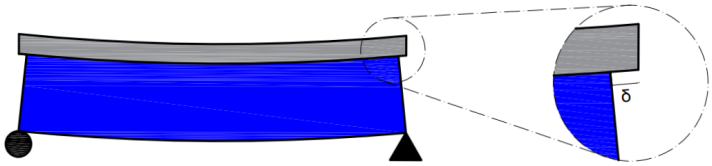
Figure 2.2: Strain distribution of a steel-concrete bridge.

2.1.1 Shear Connections

The composite action between the concrete and steel is achieved using shear connections welded to the top flange of the steel girders. Enough strength, stiffness, and ductility are required for the connectors to ensure full composite action with no slip between steel and concrete (Hällmark, 2018), as visualized in Figure 2.3. There are also different types of shear connections and the most commonly used is headed shear studs welded to the flange of the girders (Hällmark, 2018).



(a) Composite action from shear connections.



(b) Non composite action between steel and concrete.

Figure 2.3: Difference between a steel-concrete composite beam with composite- or non-composite action.

2.2 Stainless Steel

According to International Stainless Steel Forum (ISSF, 2019) the main reason for choosing stainless steel over traditional carbon steel is the corrosion resisting properties. Stainless steel is a group of alloys containing at least 10.5 % chromium that, when reacting with oxygen and water, creates a corrosion protective layer on the surface of the steel. There are several types of stainless steel where austenitic stainless steel is the most common. Another stainless steel type is Duplex stainless steel, which is considered in this work and consists of austenite and ferrite. Ferrite contributes to increased strength, while austenite is suitable for structural purposes due to its ductility, toughness, and superior corrosion resistance (ISSF, 2019). Stainless steel is also beneficial for civil engineering due to the increased fire resistance (Francis and Byrne, 2021).

2.2.1 Production of Stainless Steel Sections

The availability of hot-rolled profiles of stainless steel is less than that of traditional carbon steel and therefore welded and cold-formed profiles are most common (Stålbyggnadsinstitutet, 2017). Further, the strength of duplex stainless steel is increased by cold-working processes while heat processing can reduce it. Cold-working does however decrease the ductility of the material and does often result in asymmetric stress-strain relation along the profile. This master thesis will consider welded I-girders with flat webs and cold formed corrugated webs, with a production method further described in Section 2.3.1.

For joining stainless steel members, corresponding stainless steel fasteners are recommended by Stålbyggnadsinstitutet (2017), but other materials are accepted if they fulfill similar corrosion resisting criteria. Duplex stainless steel is weldable, but the welding should be performed carefully to avoid intermetallic phases (Francis and Byrne, 2021) and damage to the corrosion resistant layer (Stålbyggnadsinstitutet, 2017). Stålbyggnadsinstitutet (2017) recommends that welding of stainless steel should be conducted by highly qualified professionals and the added weld material should have the same corrosion resisting material.

2.2.2 Constitutive Relation

An important difference between stainless steel and carbon steel is the stress-strain relation (Stålbyggnadsinstitutet, 2017). Due to the high ductility of stainless steel, especially Duplex, the material does not exhibit a clear yielding point as carbon steel does, see Figure 2.4. Instead, stainless steel shows a continuous non-linear behavior. The yielding stress for stainless steel is therefore usually determined as the stress corresponding to 0.2 % plastic strain, see Figure 2.5.

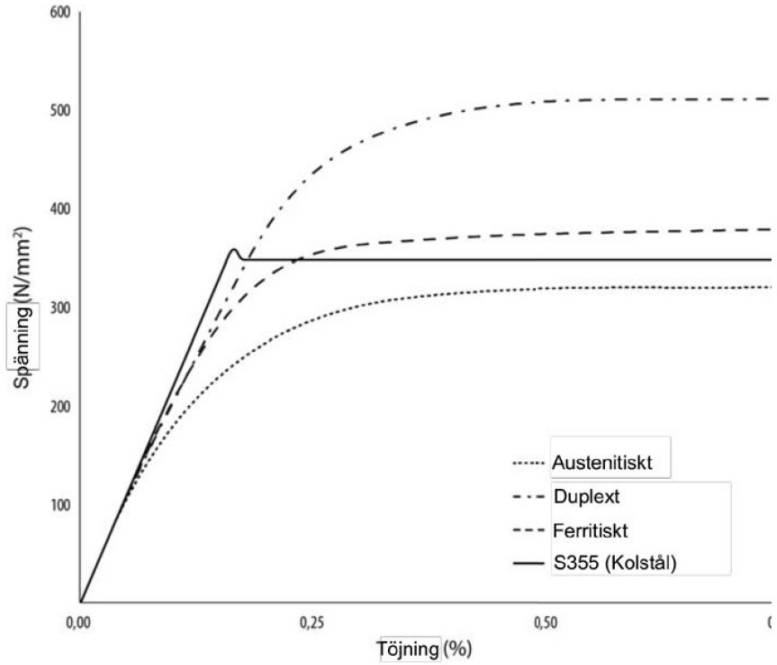


Figure 2.4: Stress-strain relation for carbon steel and stainless steel of different types. Image from Stålbyggnadsinstitutet (2017).

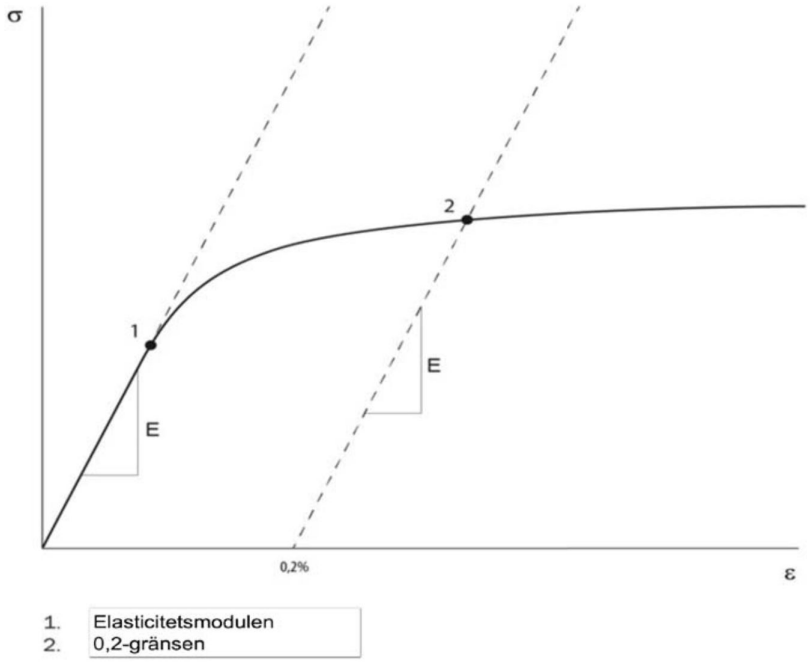


Figure 2.5: Definition of yield point (the 0.2% limit) of stainless steels. Image from Stålbyggnadsinstitutet (2017).

2.2.3 Deflection of Stainless Steel Beams

The non-linear behavior of stainless steel needs to be accounted for when evaluating the deflection in *Serviceability Limit State* (SLS). Stålbyggnadsinstitutet (2017) as well as Eurocode (Swedish Standards Institute, 2002b) instructs to use the secant modulus instead of the elastic modulus, with the secant modulus defined in Equation 2.1.

$$E_s = \frac{E_{sf.t} + E_{sf.c}}{2} \tag{2.1}$$

where $E_{sf.t}$ and $E_{sf.c}$ is the secant modulus for the flange in tension and compression, respectively, calculated as Equation 2.2.

$$E_{sf} = \frac{E}{1 + 0.002(\frac{E}{\sigma_{f.SLS}})(\frac{\sigma_{f.SLS}}{f_y})^n} \tag{2.2}$$

where $\sigma_{f.SLS}$ is the stress in the flange at serviceability load level and n is the Ramberg-Osgood parameter considering the non-linearity of the materials. For Duplex stainless steel Swedish Standards Institute (2002b, Table 6.5) instructs $n = 5$, however Stålbyggnadsinstitutet (2017, Table 6.4) states that the Eurocode 3 value is based on narrow test data and advises instead the use of $n = 8$ that is expected to be the replaced value in the next version of Eurocode 3.

2.2.4 Thermal Expansion of Stainless Steels

Another aspect to consider when using stainless steel instead of carbon steel is the increased thermal expansion coefficient. The thermal impact on the shrinkage stress of composite beams is, as highlighted by Steffner and Öman (2021), neglected in Eurocode 4 as the thermal expansions coefficients of concrete and carbon steel are assumed identical. However, for stainless steel which has an up to 60 % larger thermal expansion coefficient compared to concrete, temperature variations will result in added stresses. The highest difference is between carbon steels and austenitic stainless steels, while for Duplex stainless steel the difference is around 30%, see Table 2.1.

Table 2.1: Coefficient of linear expansion for different steel materials.

Material	α [1/°C]
Austenitic	$16 \cdot 10^{-6}$
Duplex	$13 \cdot 10^{-6}$
Carbon steel	$12 \cdot 10^{-6}$

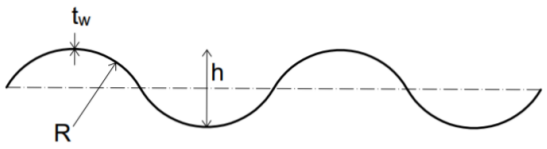
2.3 Girders with Corrugated Webs

Girders of steel are normally made of plates with flat webs and flanges. Laboratory tests of I-beams have proved that an increased height of the web improves the bending resistance of the beam. With an increased sectional area, also the shear

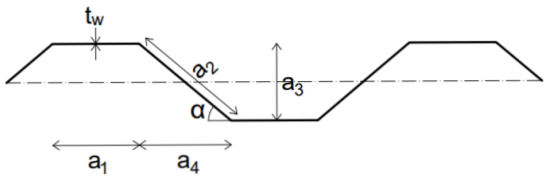
resistance increases, meaning that an increase in web height both affects the bending and shear resistance positively (Górecki and Śledziewski, 2021). However, an increase in web height increases the risk of buckling and therefore the thickness of the web need to be increased or stiffeners need to be added to ensure stability. A thicker web will though result in large material use and an increase in stiffeners means a higher risk for fatigue due to the increased number of welded connections. Therefore, another solution to the problem is to use a corrugated web instead of a flat. Then, both the thickness of the web and the number of stiffeners can be decreased which has a positive effect on the material used.

2.3.1 Types and Production of Corrugated Webs

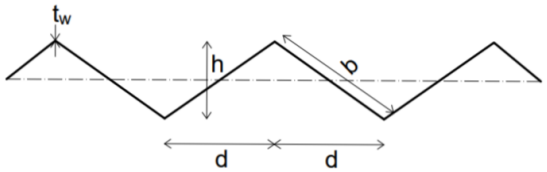
There are different types of corrugation used in corrugated webs, see Figure 2.6, where the trapezoidal is most commonly used (Sayed-Ahmed, 2007) and sinusoidal after (Górecki and Śledziewski, 2021). For small thicknesses of the web, the trapezoidal shape has higher ultimate strength while for high thicknesses, the sinusoidal has both higher ultimate bearing and stiffness (Hosseinpour et al., 2015).



(a) Sinusiodal corrugation shape.



(b) Trapezoidal corrugation shape.



(c) Triangular corrugation shape.

Figure 2.6: Different corrugation shapes: sinusiodal, trapezoidal, and triangular.

Corrugated webs are usually cold-formed by a plate being pressed onto a form with the corrugation (Pasternak and Kubieniec, 2010). The process of cold bending prevents the reduction of steel ductility and surface cracking (Boutillon et al., 2015). As this process requires form-work, some standardization of the corrugation parameters is necessary. However, the flanges are welded to the corrugated web and can

have more varying dimensions.

2.3.2 Design of Beams with Corrugated Webs

Due to the disrupted geometry, the corrugated web cannot transfer longitudinal stresses (Johansson et al., 2007). Therefore, the bending resistance of the corrugated beam is determined by the bending strength of the flanges and the contribution from the web is neglected. Some positive effects of the corrugation on the lateral-torsional buckling resistance could however be expected as it gives the web some transverse stiffness compared to a flat web (Johansson et al., 2007). A master's thesis study by Larsson and Persson (2013) identified increased rotational stiffness of the corrugated web to positively influence the lateral-torsional buckling. Although, this is not yet studied in detail and therefore not included in Eurocode (Johansson et al., 2007). The shear strength is dependent on the corrugated web and the shear capacity is determined by either buckling or yielding of the web (Sayed-Ahmed, 2007). Buckling can occur both locally in a panel or globally over several panels as described in Figure 2.7 (Hosseinpour et al., 2015). Additionally, interaction between the two buckling modes can occur and therefore needs to be verified (Sayed-Ahmed, 2007).

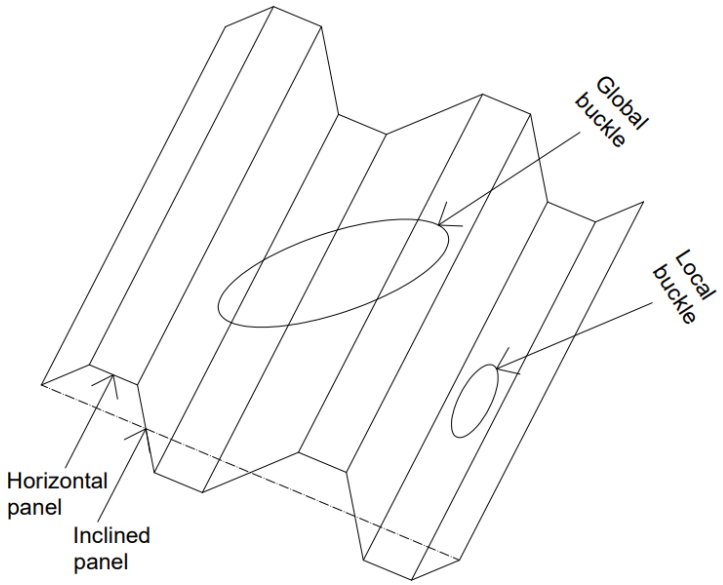


Figure 2.7: Illustration of local and global buckling.

2.3.3 Shear Buckling of Corrugated Webs

The shear capacity of corrugated webs is of interest to many researchers. Johansson et al. (2007) present, in a commentary paper to the Eurocode 3 guidelines, four additional models considering shear buckling. The difference between the four models and the model in Eurocode is however large, but they concluded that the reason for this can be that the Eurocode model does not include interaction between global and local shear buckling. This is motivated by the interaction mode being less likely to happen. However, in cases where it does appear, the effect of it is assumed smaller than for the other two modes and therefore already covered by the separate checks.

Tests by Johansson et al. (2007) confirm this to be true and concludes the Eurocode model to be the best choice available in terms of safety.

A previous master's thesis by Hlal and Mohra (2021) studied the shear behavior of corrugated webs in stainless steel using several different calculations models. The results show that the shear capacity of these beams is sensitive to initial imperfections, but concludes, similarly to Johansson et al. (2007), that the Eurocode model with recommended initial imperfection $h_w/200$ is safe to use. Hlal and Mohra (2021) also conducted a parametric study and suggested that increasing the height, thickness, and corrugation depth of the web leads to a higher shear capacity of the beam.

Another previous master's thesis, by Karlsson (2018), investigated which corrugation parameters that affect the buckling mode and the shear strength. Karlsson (2018) concluded that the angle of corrugation, α, and length of the horizontal panel, a_1, (Defined in Figure 2.6) were the parameters that had the largest impact on both buckling mode and shear capacity. By increasing the angle, α, the shear capacity increases. However, the increase is less after a specific point and the buckling mode goes from global to interactive. Additionally, the length of the horizontal panel, a_1, has an optimal value for a certain web design giving the maximum shear capacity. For smaller lengths of the horizontal panel global buckling is the governing buckling mode, and for intermediate values, interactive buckling is governing. Further, for high values the dominant mode is local buckling.

2.3.4 Local Buckling of Flanges

As the bending moment is only taken by the flanges, local buckling of the flanges is important to consider (Johansson et al., 2007). The local buckling is accounted for in the design procedure by cross-section classification and by reducing the effective area if needed (in more detail explained in Section 3.1.1). The flange outstand length, c, used to evaluate the cross-section class is in Eurocode 3 defined as the longest distance between the web and the flange edge (Swedish Standards Institute, 2005c). For a beam with a narrow flange and a deep corrugation, plate-type buckling within the larger outstand will be governing (Johansson et al., 2007). Therefore it seems reasonable to take c as stated in Eurocode 3. However, for a beam with narrow corrugation and wide flanges, the buckling mode of the flanges will be rotation around the web centerline (Johansson et al., 2007). Here Johansson et al. (2007) suggest taking c instead as half the flange width. Johnson and Cafolla (1997, as cited in Johansson et al., 2007) suggest, as a general approach, to evaluate c as the average outstand distance if the criteria of Equation 2.3 is fulfilled.

$$\frac{(a_1 + a_4)a_3}{(a_1 + 2a_4)b_1} < 0.14 \tag{2.3}$$

where b_1 is the width of the compressed flange and a_i is the corrugation parameters defined in Figure 2.6.

2.4 Engineering Optimization

Most optimization methods used by engineers aim to minimize or maximize an objective. Commonly, the methods also contain constraints to narrow down the number of solutions. Within structural engineering, one common purpose of optimization could be to minimize the material use with a constrain to keep the material utilization below 1. Mathematically, an optimization problem can be described by Equation 2.4 to 2.6 (Yang, 2010).

$$\text{minimize} \quad f_i(\mathbf{x}) \quad i = 1, 2, ..., M \tag{2.4}$$

$$\text{subjected to} \quad g_j(\mathbf{x}) = 0 \quad j = 1, 2, ..., J \tag{2.5}$$

$$\text{subjected to} \quad h_k(\mathbf{x}) \leq 0 \quad k = 1, 2, ..., K \tag{2.6}$$

where $f_i(\mathbf{x})$ is the objective functions, $g_j(\mathbf{x})$ the equality constrains and $h_k(\mathbf{x})$ the inequality constrains. These functions are all functions of the design vector $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ containing the n-number of design variables. M, J and K are the number of objective functions and constrains.

2.4.1 Single- or Multi-objective Optimization

One way of classifying optimization problems is by differentiating between single-objective ($M = 1$) and multi-objective ($M > 1$) problems. A single-objective optimization can be to minimize material use, while in multi-objective optimization, the goal can be to both minimize material cost and environmental impact. There are also cases where there are only constraints and no objective functions, which Yang (2010) classifies as feasibility problems, rather than optimization problems. The other extreme, where there are large number of objectives dependent on many different constraints, Yang (2010) calls a black-box problem.

All multi-objective optimization problems create an extra complexity as it requires the user to define a balance between the objectives (Yang, 2010), which in most cases requires a parametric study (Blank and Deb, 2020). Often the objectives are conflicting, were the optimization then results in a set of possible solutions rather than one solution. This set of solution is called a Pareto set or a Pareto front that defines within which boundaries possible solutions can be found. Blank and Deb (2020) describes Multi-Criteria Decision Making to be the next step to reach fewer or one single solution. One method of Multi-Criteria Decision Making is to normalize the objectives so they can be compared. Another method is to decompose the Pareto front by weights that the user defines for each objective.

2.4.2 Deterministic or Stochastic Optimization Algorithms

When it comes to classifying optimization algorithms, one common way according to Yang (2010), is to differentiate them by their determinacy. If the algorithm follows a rigid path in a repeatable way, the algorithm is categorized as deterministic. If the algorithm is instead based on a fully or partly randomized search, it is categorized as

stochastic. Stochastic algorithms can be further sub-categorized into heuristic and meta-heuristic algorithms. The first, heuristic, is fully based on the concept of trial and error, which, as described by Yang (2010), results in a solution found rather quickly. However, there is no guarantee that the found solution is the true optimal. The second, meta-heuristic, mixes randomization with local search. According to Yang (2010), the local search part of the method is used to improve the quality of the solution, while the randomization helps to avoid getting stuck at a local minimum or maximum.

2.4.3 Genetic Algorithms

The concept of Genetic Algorithms was first developed by Holland in the mid-1960s. Holland (1992) describes how he by studying the adaption and evolution of nature, artificially mimicked the two primary processes: natural selection and reproduction. The technique was then further tested and advanced from the 1970s and forward by Holland himself and other researchers, with Goldberg being particularly influential (Holland, 1992). The process and components of a simple genetic algorithm are described by Sivanandam and Deepa (2008) where the optimal solution is found by testing a set of possible solutions. Each solution is called a chromosome or an individual (Yang, 2010), where the first will be preferred in this thesis (Bozorg-Haddad et al., 2017). A set of chromosomes is called a population. The initial population is usually randomly generated while new populations are generated by reproduction operators that combine and modify the chromosomes by some degree of selection and randomness. The selection of which to keep and possibly further combine is done by a fitness function evaluating the population by finding the maximum or minimum (Sivanandam and Deepa, 2008). In literature, fitness and objective are commonly interchangeable terms used to define the same function (Sivanandam and Deepa, 2008)(Yang, 2010), while in this master's thesis these terms will be differentiated as described in Section 4.2.5 (Bozorg-Haddad et al., 2017).

The Genetic Algorithm is defined by Yang (2010) as a population-based metaheuristic algorithm, meaning that the population is randomly generated while the selection is done by searching for the most optimal solution within the set. To reach a quality solution, the parameters of new sets need to be carefully chosen. Yang (2010) states that inappropriate choices can cause the algorithm to never converge or lead to a meaningless result. Chisari and Amadio (2018) claim genetic algorithms useful for structural engineering problems as they often include black-box problems such as parameter identification and topology optimization. Similarly, Holland (1992) also enhanced genetic algorithms as a way of solving problems without fully understanding their structure and full features. For this purpose Chisari and Amadio (2018) recommends an initial population with a wide range. If randomly generated, a wide range equals a large initial population. Likewise, Yang (2010) describes that a small population for any single- or multi-objective optimization problem often leads to the algorithm finding a local minimum or maximum rather than the global. However, a larger population requires more evaluations and longer computational time.

2.5 Life Cycle Cost and Life Cycle Assessment

As described in the standard SS-EN ISO 14040:2006 (Swedish Standards Institute, 2006), a *Life Cycle Assessment* (LCA) considers the life cycle of a product and the whole chain from material extraction to manufacturing, usage, and disposal/recycling are analyzed to get an overview of the environmental impact from the different stages. In this way, potential environmental burdens can be identified and possibly avoided. Commonly, only environmental impacts are considered in a Life Cycle Assessment meaning that economical and social sustainability are not included. The analysis contains an iterative work in four phases: the goal and scope definition, inventory analysis, impact assessment, and interpretation.

Life Cycle Cost (LCC) is, on the other hand, a tool to investigate the cost of a product during a specific time frame. Both investment and maintenance costs are considered in the analysis leading to the possibility of comparing alternatives with different costs during the time frame. Therefore, also future costs need to be considered. The rules for a Life Cycle Cost Analysis are found in the standard SS-ISO 15686-1 (Swedish Standards Institute, 2003).

3
Design Procedure

The guideline to follow when designing a bridge in Sweden is *Krav Brobyggande* developed by Trafikverket (2019). The document includes requirements to be followed during design of bridges and refers to the design codes Eurocode and *Transportstyrelsens föreskrifter och allmänna råd om tillämpning av eurokoder* (TSFS 2018:57). The Eurocode consists of ten different parts from SS-EN 1990 to SS-EN 1999 and most of them are in turn divided into subsections. *Transportstyrelsens föreskrifter och allmänna råd om tillämpning av eurokoder* is conducted by Transportstyrelsen and *Krav Brobyggande* states that whenever the two codes contradict eachother, *Transportstyrelsens föreskrifter och allmänna råd om tillämpning av eurokoder* is the dokument to follow.

The parts of Eurocode used in this master's thesis are summarized in Table 3.1. The Eurocodes are provided by the Swedish Standards Institute, but for clarity, references to the specific Eurocode will here from be made. Similarly, the steering documents provided by Trafikverket and Transportstyrelsen mentioned above will be referred to by their codes.

Table 3.1: Eurocodes used in this master's thesis.

Eurocode	**Sections**
SS-EN 1990 - Basis of structural design	-
SS-EN 1991 - Actions on structures	Part 1-1: General actions Part 1-4: Wind actions Part 1-5: Thermal actions Part 2: Traffic loads on bridges
SS-EN 1992 - Concrete structures	Part 1-1: General rules Part 2: Concrete bridges
SS-EN 1993 - Steel structures	Part 1-1: General rules Part 1-2: Structural fire design Part 1-4: Stainless steels Part 1-5: Plated structural elements Part 1-8: Design of joints Part 1-9: Fatigue Part 2: Steel bridges
SS-EN 1994 - Composite steel and concrete structures	Part 1-1: General rules Part 2: General rules and rules for bridges

For the design of composite bridges, SS-EN 1994 should be used. The code often refers to SS-EN 1993 and SS-EN 1992 for specific design and verification of steel and concrete parts, respectively. Therefore the following chapter will present the design procedures for steel and concrete in separate sections, Section 3.1 and 3.2, before presenting the composite system in Section 3.3, the loads in Section 3.4, the analysis in Section 3.5, and finally the design verification in Section 3.6. The relevant states considered are Ultimate limit state during construction and service life as well as Serviceability limit state. Additionally, Fatigue is verified.

3.1 Steel Material

The steel types considered in this master's thesis are commonly used construction carbon steel of grade S355, less common high strength structural carbon steel of grade S460, and Duplex stainless steel of grade 1.4462. Their material properties are presented in Table 3.2 (SS-EN 1993-1-1, Table 3.1 EN 10025-3) and 3.3 (SS-EN 1993-1-4 A1:2015, Table 2.1). The material strength depends on the thickness, t, of the various parts. For both steels, the assumed density is 7800 kg/m^3, and Poisson's ratio 0.3.

Table 3.2: Steel properties of carbon steels S355 and S460.

Parameter	**S355 N/NL**	**S460 N/NL**
Yield strength, f_y [MPa]		
$t \leq 40mm$	355	460
$40mm < t \leq 80mm$	335	430
Ultimate strength, f_u [MPa]		
$t \leq 40mm$	490	540
$40mm < t \leq 80mm$	470	540
Elastic modulus, E_s [GPa]	210	210

Table 3.3: Steel properties of stainless steel Duplex 1.4462.

Parameter	**Duplex 1.4462**
Yield strength, f_y [MPa]	
$t \leq 8mm$	530
$t \leq 13.5mm$	480
$t \leq 75mm$	450
Ultimate strength, f_u [MPa]	
$t \leq 8mm$	700
$t \leq 13.5mm$	680
$t \leq 75mm$	650
Elastic modulus, E_s [GPa]	200

The material design resistance should be calculated according to SS-EN 1994-1-1, Section 2.4.1.4. The document refers to equation 6.6a or 6.6c in SS-EN 1990, see equation 3.1.

$$R_d = \eta \frac{R_k}{\gamma_m} \tag{3.1}$$

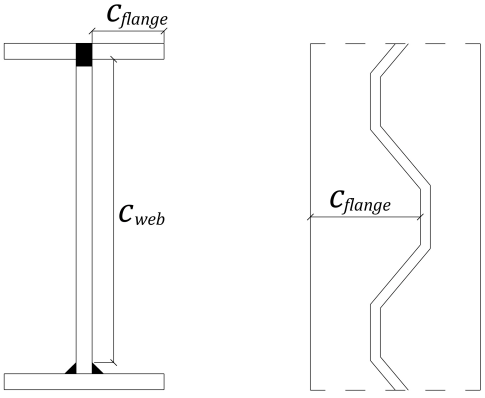
where R_d is the design resistance, R_k is the characteristic value of the resistance and γ_m is a partial factor accounting for deviations in the material. However, γ_m could be replaced by the factor γ_M including both γ_m and η. The values of γ_M for steel depend on the considered resistance. The different factors and their recommended national values for steel bridge design in Ultimate Limit State are listed in Table 3.4 for carbon steel and stainless steel (SS-EN 1993-2, Section 6.1) (SS-EN 1993-1-4, Section 5.1). According to SS-EN 1990, Section 6.5.4 the value for γ_M in Serviceable Limit State is equal to 1.0.

Table 3.4: Safety factors for steel in Ultimate limits state and Fatigue analysis for carbon and stainless steel.

$\gamma_{\mathbf{Mi}}$	**Resistance considered**	**Carbon Steel**	**Stainless Steel**
γ_{M0}	Yield failure or buckling.	1.0	1.1
γ_{M1}	Instability.	1.1	1.1
γ_{M2}	Tension failure and resistance of joints and welds.	1.25	1.25

3.1.1 Cross-Section Classification

To account for local buckling of steel sections, the cross-section class is evaluated for each part. This is done by comparing the ratio of the length, c, and thickness, t, against set criteria dependent on the yield strength and modulus of elasticity of the material (SS-EN 1993-1-1, Table 5.2). The length c for flat webs and attached flanges is defined in Figure 3.1a. For flanges attached to corrugated webs, SS-EN 1991-1-5, Annex D instructs the length to be taken as the longest distance to the web axis, as seen in Figure 3.1b. However, as described in Section 2.3.2, depending on the design of the corrugated beam, c could also be set as half the flange width or as the average distance to the web.



(a) Flat web. **(b)** Corrugated web.

Figure 3.1: The c-distances for a I-section girder.

The four cross-section classes and their definitions are presented in Table 3.5 (SS-EN 1993-1-1, Section 5.5.2). The less the ratio length/thickness, c/t, the higher possibility for a high-level classification. If Class 1 or 2 is achieved, the plastic contribution can be included in the capacity calculations. However, since Linear Elastic Analysis will be considered in this thesis, the elastic capacity will be considered for Class 1 and 2 also. If Class 3 is reached, the full elastic capacity can be considered, while for Class 4, the area needs to be reduced to account for the effects of local buckling.

Table 3.5: Definition of the different cross sectional classes and corresponding appropriate section modulus.

Class	**Definition**	**Section modulus**
1	Enough capacity to form plastic hinges with sufficient rotational capacity.	$W_{pl,y}$
2	Enough capacity to form plastic hinges, but with limited rotational capacity.	$W_{pl,y}$
3	Capacity to reach yield strength, but unlikely to develop plastic hinges.	$W_{el,y}$
4	Local buckling is likely to occur.	$W_{eff,y}$

For parts in Class 4 without longitudinal stiffeners, the effective area is evaluated according to Equation 3.2 (SS-EN 1993-1-5, Section 4.4). However, Equation 3.2 is only used for verification in Ultimate Limit State, while cross-section reduction of the steel section in Serviceable Limit State and Fatigue is done according to Section 3.1.4 accounting for shear lag deformations.

$$A_{si,eff} = \rho \cdot A_{si} \tag{3.2}$$

with $i = 1, 2, ..., n$ with n as the number of cross-sectional parts in Class 4. A_{si} is the gross cross-sectional area of the part and ρ is the reduction factor dependent on the plate slenderness and stress distribution of the part. For carbon steel, the calculation procedure of ρ is described in SS-EN 1993-1-5, Section 4.4 and for stainless steel, ρ should be calculated according to SS-EN 1993-1-4, Section 5.2.3. To account for the corrugated web when determining the effective area of the flanges, the buckling factor used to determine the plate slenderness should be evaluated as described in SS-EN 1991-1-5, Annex D, rather than in SS-EN 1993-1-5, Table 4.1 or 4.2 for flat webs and flat web flanges.

The exact classification criteria for flat webs and flanges are found in SS-EN 1993-1-1, Table 5.2 for carbon steel and in SS-EN 1993-1-4, Table 5.2 for stainless steel (with a few replacement values found in SS-EN 1993-1-4 A1:2015, Table 5.2). For this type of bridge design with negligible axial forces and local bending of flanges, the flat webs should be classified considering bending and the flanges considering compression due to global bending. Since, as described in Section 2.3.2, the bending capacity of girders with corrugated webs is only handled by the flanges, the corrugated web does not need to be classified. Further, for Ultimate Limit State verification in the service state when the upper flange of the steel girder is attached to the concrete

by shear connections, the flange can be assumed to be restrained from buckling and classified in Class 1. An assumption is then made that the spacing of the shear connections is sufficiently according to Section 6.6.5.5 in SS-EN 1994-1-1.

3.1.2 Lateral-Torsional Buckling

For the final service state, the compressed flanges of the bridge girders will be restrained by the concrete deck and can be concluded laterally stable (SS-EN 1994-1-1, Section 6.4.1). However, in the construction phase before the hardening of concrete, the girders need to be verified against lateral-torsional buckling (SS-EN 1993-1-1, Section 6.3.2.1). The lateral stability is decided by the non-dimensional slenderness, λ_{LT}, depending on the yield strength and section modulus of the cross-section. If λ_{LT} is equal or smaller than $\lambda_{LT,0}$, lateral-torsional buckling can be ignored (SS-EN 1993-1-1, Section 6.3.2.2). However, if λ_{LT} is larger than $\lambda_{LT,0}$, the moment capacity of the girder should be reduced with a factor χ_{LT}, calculated by Equation 3.3. The maximum recommended value of $\lambda_{LT,0}$ is 0.4 (SS-EN 1993-1-1, Section 6.3.2.3).

$$\chi_{LT} = \left[\frac{1}{\Phi_{LT} + \sqrt{\Phi_{LT}^2 - \lambda_{LT}^2}} \right] \leq 1.0 \tag{3.3}$$

with Φ_{LT} evaluated in Equation 3.4 or 3.5 depending on the steel material (SS-EN 1993-1-1, Section 6.3.2.2) (SS-EN 1993-1-4, Section 5.4.3.1).

$$\text{Carbon steel:} \quad \Phi_{LT} = 0.5(1 + \alpha_{LT}(\lambda_{LT} - 0.2) + \lambda_{LT}^2) \tag{3.4}$$

$$\text{Stainless steel:} \quad \Phi_{LT} = 0.5(1 + \alpha_{LT}(\lambda_{LT} - 0.4) + \lambda_{LT}^2) \tag{3.5}$$

The imperfection factor, α_{LT}, depends on the buckling curve. For welded I-sections of carbon steel α_{LT} equals 0.49 if the height of the beam is equal to or less two times the width of the flanges. Otherwise α_{LT} equals to 0.76 (SS-EN 1993-1-1, Table 6.3 and 6.4). For all welded I-sections of stainless steel, α_{LT} is set equal to 0.76.

3.1.3 Shear Buckling

For flat webs with transverse stiffeners, shear buckling should be considered if the criteria in Equation 3.6 for carbon steel (SS-EN 1993-1-5, Section 5.1) or Equation 3.7 for stainless steel (SS-EN 1993-1-4, Section 5.6) is not fulfilled. In such a case, the shear capacity should be reduced by a factor χ_w based on the end post rigidity, shear slenderness parameter λ_w and a factor η with a recommended value of 1.20. The equations for χ_w are found in SS-EN 1993-1-5, Table 5.1 for carbon steel and in SS-EN 1991-1-4, Section 5.6 for stainless steel.

$$\text{Carbon steel:} \quad \frac{h_w}{t_w} \leq \frac{31}{\eta} \varepsilon \sqrt{k_\tau} \tag{3.6}$$

$$\text{Stainless steel:} \quad \frac{h_w}{t_w} \leq \frac{23}{\eta}\varepsilon\sqrt{k_\tau} \tag{3.7}$$

with the geometrical parameters defined in Section 3.3.1 and ε defined in Section 3.1.1. The buckling shear coefficient, k_τ, is defined in SS-EN 1991-1-5, Annex A.3 and depends on the height of the web and the distance between transverse stiffeners.

Corrugated webs should for all cases be verified against shear buckling. The reduction factor, χ_c, should be taken as the smallest value of $\chi_{c,l}$ and $\chi_{c,g}$, where the first is the reduction factor accounting for the local buckling mode and the second the reduction factor accounting for the global buckling mode (SS-EN 1993-1-5, Annex D). The reduction factors are calculated by Equation 3.8 and 3.9.

$$\chi_{c,l} = \left[\frac{1.15}{0.9 + \lambda_{c,l}}\right] \leq 1.0 \tag{3.8}$$

$$\chi_{c,g} = \left[\frac{1.5}{0.5 + \lambda_{c,g}^2}\right] \leq 1.0 \tag{3.9}$$

The shear slenderness parameters for corrugated webs, $\lambda_{c,l}$ and $\lambda_{c,g}$, are defined by the yield strength of the material and the critical shear stress of the web. The critical shear stresses for local and global buckling, $\tau_{cr,l}$ and $\tau_{cr,g}$, are calculated by Equation 3.10 and 3.11.

$$\tau_{cr,l} = 4.83E_s\left[\frac{t_w}{max(a_1, a_2)}\right]^2 \tag{3.10}$$

$$\tau_{cr,g} = \frac{32.4}{t_w h_w^2}\left[\frac{E_s t_w^3 w}{12(1-\nu^2)s}\left(\frac{E_s I_{z,w}}{w}\right)^3\right]^{1/4} \tag{3.11}$$

with the geometrical parameters defined in Figure 2.6 and 3.6. Additionally, the distances w and s are defined as the sum of a_1 and a_4 and a_1 and a_2, respectively. $I_{z,w}$ is the moment of inertia of one corrugation length w, calculated according to Equation 3.12 (Johansson et al., 2007).

$$I_{z,w} = \frac{t_w a_3^2}{12}(3a_1 + a_2) \tag{3.12}$$

3.1.4 Shear Lag for Steel Flanges

To ensure composite action, the concrete and steel are connected by shear studs transferring the shear stresses between materials, as explained in Section 2.1.1. Naturally, this transfer will never be as perfect as in theory and therefore the possibility of shear lag between the two materials needs to be considered. According to SS-EN 1994-2, Section 5.4.1.2 this can either be done by a detailed analysis where the true shear lag is identified, or by reducing the effective width of the compressed steel flange and the concrete deck. The steel section only needs to be reduced due to shear lag in Serviceable Limit State and Fatigue checks, while concrete reduction should be performed also for Ultimate Limit State verification.

The effective width of the compressed steel flange, $b_{fo,eff}$, is calculated by Equation 3.13 (SS-EN 1993-1-5, Section 3.2.1) where $b_{fo,0}$ is half the flange width and β is a factor given in SS-EN 1993-1-5, Table 3.1 depending on geometry and boundary conditions.

$$b_{fo,eff} = \beta \cdot b_{fo,0} \tag{3.13}$$

Calculation of the effective concrete width is described in Section 3.2.3 and the reduced widths are used in Section 3.3 when evaluating the composite cross-sectional parameters.

3.2 Concrete Material

Concrete is divided into several strength classes from C12/15 up to C60/75 where the first number indicates the strength by cylinder tests and the latter the strength by cube tests. However, classes up to C90/105 are taken into account in the tables in Eurocode (SS-EN 1992-1-1, Table 3.1). The concrete classes considered in this master's thesis are C30/37 to C40/50 and the strength parameters for each class are stated in Table 3.6.

Table 3.6: Strength properties of concrete class C30/37, C35/45 and C40/50.

Parameter	**C30/37**	**C35/45**	**C40/50**
Characteristic compressive strength, f_{ck}	30 MPa	35 MPa	40 MPa
Mean value of compressive strength, f_{cm}	38 MPa	43 MPa	48 MPa
Mean value of tensile strength, f_{ctm}	2.9 MPa	3.2 MPa	3.5 MPa
Characteristic tensile strength 0.5%, $f_{ctk.0.05}$	2.0 MPa	2.2 MPa	2.5 MPa
Characteristic tensile strength 95%, $f_{ctk.0.95}$	3.8 MPa	4.2 MPa	4.6 MPa
Elastic modulus, E_{cm}	33 GPa	34 GPa	35 GPa

The design resistance for concrete is, as previously mentioned, calculated using Equation 3.1 which is the same for both concrete and steel. However, the safety factors are different for concrete and specified in Table 3.7.

Table 3.7: Different safety factors for concrete used for calculations of design resistance.

Factor	**Resistance considered**	**Value**	**Reference**
γ_c	Persistant and transient in ULS	1.5	SS-EN 1992-1-1, Table 2.1N
γ_c	Serviceable limit state	1.0	SS-EN 1992-1-1, Section 2.4.2.4
$\gamma_{c,fat}$	Fatigue verification	1.5	SS-EN 1992-1-1, Section 2.4.2.4
α_{cc}	Unfavourable and long term effects on compressive strength	1.0	(SS-EN 1992-1-1, Section 3.1.6)
α_{ct}	Unfavourable and long term effects on tensile strength	1.0	(SS-EN 1992-1-1, Section 3.1.6)

The reinforcement grades included in the program are SS260S, B500B and Ks600S. There properties are stated in Table 3.8. To determine the design capacity, the characteristic values are multiplied with the safety factor γ_s equal to 1.15 SS-EN 1992-1-1, Section 2.4.2.4.

Table 3.8: Strength properties of reinforcement class SS260S, B500B and Ks600S.

Parameter	**SS260S**	**B500B**	**Ks600S**
Characteristic yield strength of reinforcement, f_{sk}	260	500	600

3.2.1 Shrinkage and Creep

Concrete has time-dependent strength effects occurring during the service life of the structure. These effects are called shrinkage and creep where creep is a load-dependent phenomenon arising in concrete during long-term loading where small deformations occur. Both creep and shrinkage are affected by humidity in the surrounding air and material properties. Creep and shrinkage are defined in Section 3.1.4 in SS-EN 1992-1-1. The creep deformation of concrete at time $t = \infty$ can be determined by Equation 3.14 (SS-EN 1992-1-1, Equation 3.6) where $\varphi(\infty, t_0)$ is the creep coefficient defined between t_0 and ∞ and can be found in Figure 3.1 in SS-EN 1992-1-1. E_c is the concrete tangent modulus which could be set equal to 1.05 times E_{cm} and σ_c is the constant compressive concrete stress.

$$\varepsilon_{cc}(\infty, t_0) = \varphi(\infty, t_0) \cdot \frac{\sigma_c}{E_c} \tag{3.14}$$

The total shrinkage deformation is divided into two parts; drying shrinkage strain, $\varepsilon_{cd}(t)$, and autogenous shrinkage strain, $\varepsilon_{ca}(t)$. The autogenous strain occurs during hardening and drying shrinkage strain continues for a longer time. However, most of the shrinkage forces occur during the first days after concrete casting (SS-EN 1992-1-1). The drying shrinkage with time is calculated using Equation 3.15 SS-EN 1992-1-1, Equation 3.9.

$$\varepsilon_{cd}(t) = \beta_{ds}(t, t_s) \cdot k_h \cdot \varepsilon_{cd,0} \tag{3.15}$$

The value of $\varepsilon_{cd,0}$ is chosen from Table 3.2 in SS-EN 1992-1-1 and is dependent on relative humidity as well as concrete class. k_h is a coefficient dependent on the notional size, h_0, and can be chosen from Table 3.3 in SS-EN 1992-1-1. The notional size depends on the cross-section area, A_c, and the length of the section exposed to drying, u. The $\beta_{ds}(t, t_s)$ is calculated using Equation 3.10 in SS-EN 1992-1-1 and t is the age of the concrete at the considered time and t_s the time when drying shrinkage begins, both in the unit days. The autogenous strain is defined in Equation 3.16 (SS-EN 1992-1-1 Equation 3.11)

$$\varepsilon_{ca}(t) = \beta_{as}(t) \cdot \varepsilon_{ca}(\infty) \tag{3.16}$$

$\beta_{as}(t)$ is defined in Equation 3.13 and $\varepsilon_{ca}(\infty)$ in equation 3.12 in SS-EN 1992-1-1.

3.2.2 Modular Ratios Considering Creep and Shrinkage

In SS-EN 1994-2, Section 5.4.2.2, creep and shrinkage are accounted for by using a modular ratio, n_L, calculated by Equation 3.17. The modular ratio is used in Section 3.3 to calculate the transformed cross-section.

$$n_L = n_0(1 + \psi_L \cdot \varphi(t, t_0)) \tag{3.17}$$

The short-term modular ratio, n_0, is defined as the ratio between the modulus of elasticity of steel, E_s, and concrete, E_{cm}. The load duration is considered by the additional creep factor ψ_L. For creep due to permanent long-term loads $\psi_L = 1.1$ and for primary creep effects such as shrinkage $\psi_L = 0.55$. For short term loads, $\psi_L = 0$. Corresponding to these loads, the creep coefficient $\varphi(t, t_0)$ is set to the final creep coefficient $\varphi(\infty, t_0)$ (in detail explained in Section 3.2.1) with $t_0 = 1$ day when considering shrinkage and $t_0 = 1$ day when considering permanent loads if all casting is done in day one, otherwise (and usually) t_0 is taken as the average day of casting.

3.2.3 Shear Lag of the Concrete Section

As explained in 3.1.4, the effective width of the concrete deck is reduced due to shear lag. According to SS-EN 1994-2, Section 5.4.1.2 the effective width, $b_{c,eff}$, should be evaluated by Equation 3.18 for the mid-span section and by Equation 3.19 for the end support sections. As visualized in Figure 3.2, b_0 is the center to center distance between the shear connectors. b_{ei} is equal to L/8, but not larger than the actual distance from the shear connector to the edge of the concrete deck for the outer part (b_{e1}) and not larger than the actual distance between the shear connector and the middle of the concrete deck for the inner part (b_{e2}). L is the span length of the bridge.

$$b_{c,eff} = b_0 + \Sigma b_{ei} \tag{3.18}$$

$$b_{c,eff} = b_0 + \Sigma \beta_i b_{ei} \tag{3.19}$$

with

$$\beta_i = \left[0.55 + \frac{0.025L}{b_{ei}}\right] \leq 1.0 \tag{3.20}$$

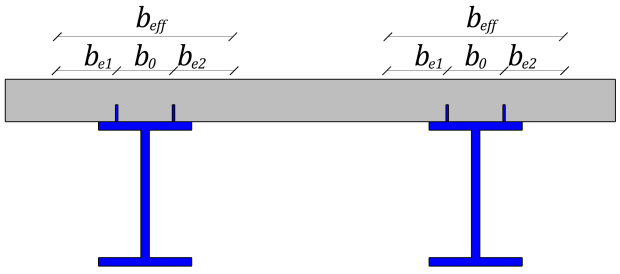


Figure 3.2: A visualization of the defined effective width of the concrete deck.

The effective width is used to calculate the effective area, $A_{c,eff}$ which is used when evaluating the cross-sectional parameters for the whole composite section (see 3.3). Note that the effective width is calculated per steel girder, meaning that for a symmetric twin girder bridge the total effective width is doubled.

3.3 Structural System

The composite bridge consists of two parallel main girders of steel and a concrete deck on top. The length of the bridge is defined by the parameter L and the width of the bridge deck by B, with B_s as the distance between the twin girders, taken as a value between $0.56B$ and $0.65B$. The main girders are divided into longitudinal segments where each segment can have different cross-sections. The number of segments is depending on the length of the bridge according to Table 3.9. Due to symmetry the segments are the same on both sides of the symmetry line, as illustrated in Figure 3.3. The full system is braced by crossbeams connected by vertical stiffeners to the main girders. The geometrical parameters are described in Figure 3.3 and the components of the bridge are displayed in Figure 3.4 and 3.5.

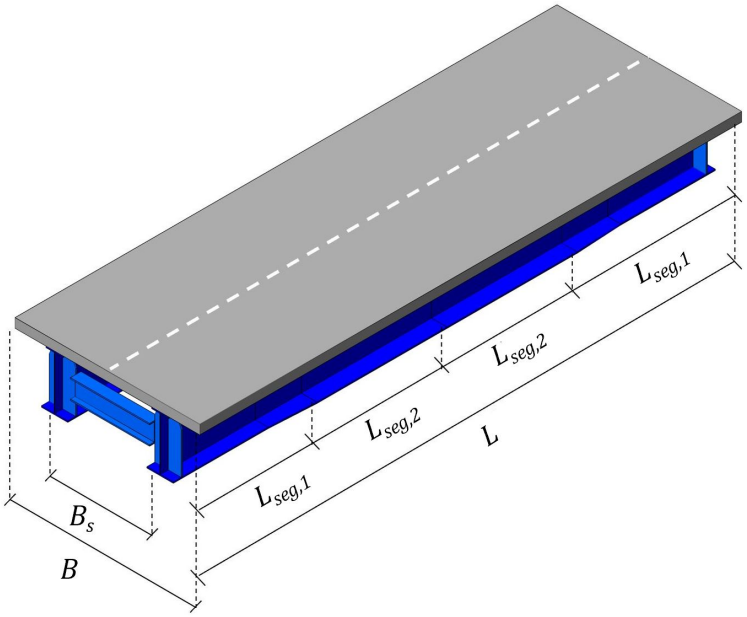


Figure 3.3: Specification of the major bridge parameters.

Table 3.9: Number of segments used in the design dependent on the length of the bridge, L.

Lenght, L [m]	**Number of segments**
$25 \leq L \leq 40$	6
$40 < L \leq 60$	10
$60 < L \leq 75$	14

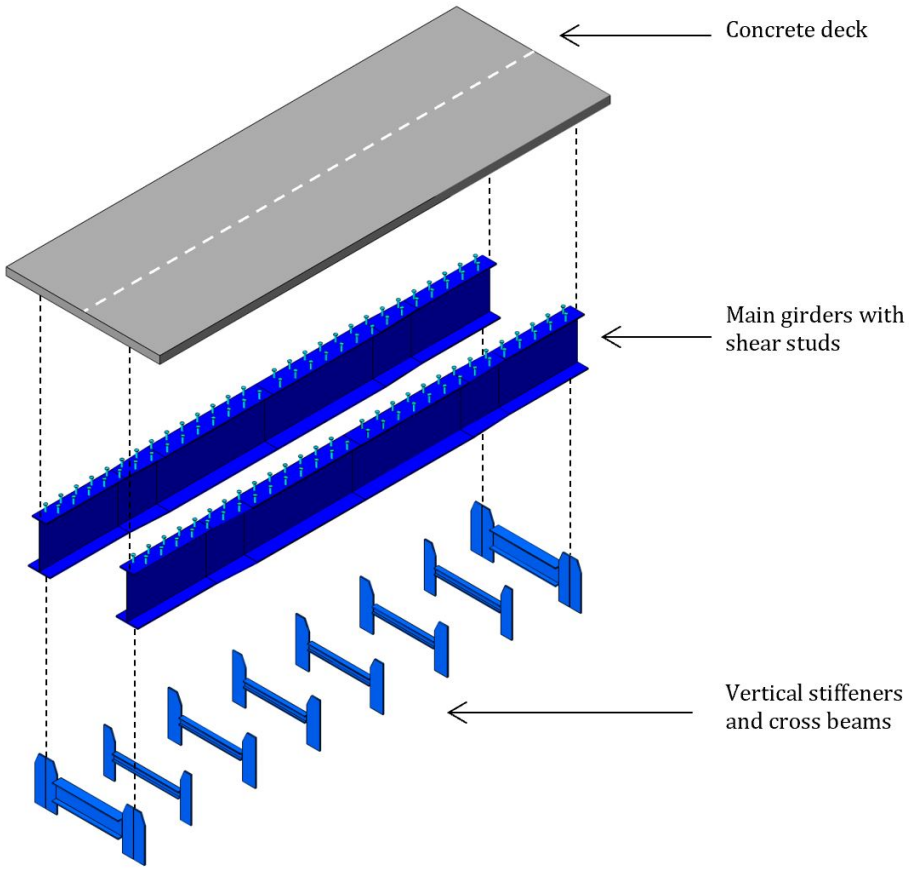


Figure 3.4: Exploded view of composite bridge.

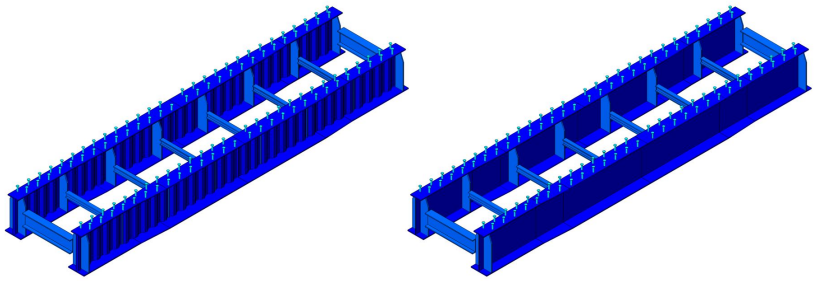


Figure 3.5: Steel system of composite bridge: main girders with corrugated webs (left) and with flat webs (right).

3.3.1 Composite Beam

The primary system consists of a composite beam with one steel girder and a concrete deck. Due to symmetry, the capacity can be verified by analyzing only one of the steel girders including the effective concrete part. The cross-section is then defined according to Figure 3.6. To optimize the material utilization considering the varying moment, the width and thickness of the flanges can vary between the lon-

gitudinal segments. The width can also be different for the top and bottom flange. Therefore, several sections will be considered during the optimization.

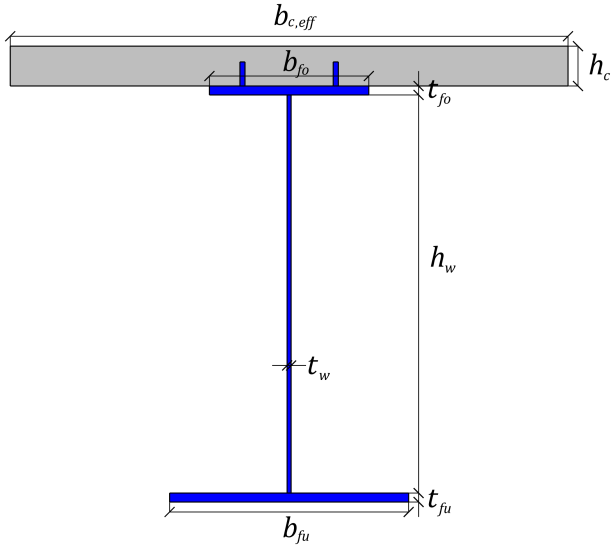


Figure 3.6: Cross-sectional parameters for one of the steel girders and the concrete deck.

The effective area of the composite section is calculated as the sum of the area of the steel parts, $A_{s,eff}$, and the contributing concrete deck, $A_{c,eff}$. See Equation 3.21. To account for any reduction of the steel, the effective steel area needs to be calculated separately for the Ultimate Limit State and Serviceable Limit State verification, according to Section 3.1.1 and Section 3.1.4, respectively. The effective width of the contributing concrete part is the same for both verification, as described in Section 3.1.4. The concrete section is transferred to an equivalent steel section with the use of modular ratios considering creep and shrinkage of the concrete. The effective area is calculated according to Equation 3.22 with modular ratio n_L depending on the load considered, as described in Section 3.2.2.

$$A_{composite} = A_{s,eff} + A_{c,eff} \tag{3.21}$$

with

$$A_{c,eff} = h_c \cdot \frac{b_{eff}}{n_L} \tag{3.22}$$

where b_{eff} and h_c are defined in Figure 3.6. Further, the moment of inertia is calculated by Steiner's theorem. Since it depends on the effective area, it will vary for different load cases and combinations.

3.3.2 Lateral Bracing

The primary system is braced by crossbeams connected by vertical stiffeners to the main girders, see Figure 3.4. The profile of the crossbeams in the span is HEA beams. If the height of the main girders are smaller than 2 meters, the crossbeam is

a single beam and if the height is equal to or larger than 2 meters, a K-truss structure is used. The distance between the crossbeams is defined with the parameter C and they are designed to handle the horizontal loads during the construction phase, such as wind and eccentricity effects caused by imperfections. In the service phase, the horizontal loads are taken by the concrete deck. In this phase, the end crossbeams should also be designed to handle the uplifting force from temporary supports in case the permanent supports need any maintenance. These beams are welded I-sections.

3.3.3 Welds

The different welds in the structure are pre-designed and verified according to their capacity. The longitudinal welds between the web and flanges in the main girders, for both flat and corrugated webs, are fillet welds connecting the bottom flange with the web and butt welds connecting the top flange with the web. The bottom fillet welds are verified at the support section where the shear action is the highest. In this weld, the shear parallel to the weld is calculated while normal stress and shear stress perpendicular to the weld equals to zero. The top butt weld is verified for the maximum shear action in each segment, as well as the traffic load from Load Model 2. The shear contributes to stresses parallel to the weld, while the traffic load contributes to normal stresses perpendicular to the weld. Here, the shear stresses perpendicular to the weld equals the normal stresses perpendicular to the weld. The welds are automatically done in the manufacture. The thickness of the fillet weld determines how many runs of welding that is needed and thereby the welding length used for the Life Cycle Cost analysis.

The main girder and the stiffeners are assumed to be welded together in the workshop, with fillet welds connecting the stiffeners to the web and bottom flange and but welds connecting the stiffeners to the top flange. The capacity check of these welds are not done but the length of them are included in the Life Cycle Cost analysis. Since the bridge consist of at least two lanes with a width of 3 meters, the twin girders are assumed to not be able to be transported connected on one truck. Therefore, the intermediate crossbeams are bolted to the stiffeners at site and the end crossbeams are welded to the stiffeners at site. The longitudinal segments are welded together in the workshop. If the bridge is longer than 15 meters, the bridge is subdivided and the splices are welded together at site by butt welds. Further, the shear studs are automatically attached in the workshop. These welds are not verified, but rather assumed to be less critical than the bolt shear capacity.

3.4 Loads and Load Combinations

For the design of a composite road bridge, the considered loads are divided into permanent and variable loads. The permanent loads are permanently acting on the structure, such as the weight of the construction materials, while the variable loads acts in different times and of different magnitude, such as traffic load and wind load. The considered permanent loads are self-weight and shrinkage load while the considered variable loads are traffic, wind, acceleration/braking, and temperature.

Snow load is not considered for road bridges due to snow removal to enable traffic.

3.4.1 Load Combinations

The load combinations used in design of composite steel and concrete bridges are regulated in Eurocode 4, Section 2.4.2 (SS-EN 1994-1-1). The Section refers to the load combinations given in Eurocode 0, and in Eurocode 0 there is an Annex A2 regulating, among others, the load combination for bridges (SS-EN 1990). In TSFS 2018:57, Chapter 2, 8§, it is stated that when using the Partial Factor Method, a partial factor γ_d should be used in Ultimate Limit State. The partial factor depends on the specified safety class and is summarized in Table 3.10 for different classes. The safety class is chosen with help from Table 2.1 in TSFS 2018:57. Values of the safety factors γ_i and ψ_i for different load combinations are stated in Appendix A.

Table 3.10: Different values of the partial factor, γ_d, depending on the safety class.

Safety class	**Value [-]**
Safety class 1	0.83
Safety class 2	0.91
Safety class 3	1.0
Safety class 4	1.1

Section A2.2.1 (3) in SS-EN 1990 clarifies that for Ultimate Limit State, the less favorable of Equation 6.10a and 6.10b in Eurocode 0 is used during design in STR. STR means that the material failure is governing, for example by excessive deformation or internal failure. In Equation 3.23 and 3.24, the partial factor γ_d is included.

$$\sum_{j\geq 1} \gamma_d \gamma_{G,j} G_{k,j} + \gamma_P P + \gamma_d \gamma_{Q,1} \psi_{0,1} Q_{k,1} + \sum_{i>1} \gamma_d \gamma_{Q,i} \psi_{0,i} Q_{k,i} \tag{3.23}$$

$$\sum_{j\geq 1} \gamma_d \xi_j \gamma_{G,j} G_{k,j} + \gamma_P P + \gamma_d \gamma_{Q,1} Q_{k,1} + \sum_{i>1} \gamma_d \gamma_{Q,i} \psi_{0,i} Q_{k,i} \tag{3.24}$$

where,

$G_{k,j}$	Characteristic value of the self-weight
P	Other permanent loads
$Q_{k,1}$	Main variable load
$Q_{k,i}$	Other variable loads
γ_i	Safety factor
ψ_i	Factor regulating the variable loads
ξ_j	Reduction factor for unfavourable permanent actions

For Serviceable Limit State, frequent and quasi-permanent load combination is used. The frequent combination is used for reversible limit states during calculations of deflection and is regulated by Equation 6.15b. The quasi-permanent combination account for long-term effects and the appearance of the structure. It is used for

calculations of crack width and is regulated in Equation 6.16b (SS-EN 1990). Since only deflection will be verified in the Serviceable Limit State in this master's thesis, the frequent load combination is the only one stated in Equation 3.25. Here SS-EN 1990, Section 6.5.3 (2) indicates that all partial factors should be set to 1.0. For the Fatigue Limit state, the partial safety factors should also be set equal to 1.0.

$$\sum_{j\geq 1} G_{k,j} + P + \psi_{1,1} Q_{k,1} + \sum_{i>1} \psi_{2,i} Q_{k,i} \tag{3.25}$$

3.4.2 Self-Weight

The self-weight is considered as a line load and includes the total weight of the main girders, concrete deck with pavement, crash barriers, and form-work. The corresponding load of the steel and concrete parts is calculated by multiplying their weight from Table 3.11 by the area. The addition from the concrete deck differs between short- and long-term since the density varies with the hardening of the concrete. This as unhardened concrete contains a large amount of water before it dries and therefore has a higher density. For the steel girders with corrugated webs, the additional length of the corrugation is considered using a ratio between the flat and corrugated length, presented in Equation 3.26. The parameters a_1, a_2, and a_4 are specified in Figure 2.6. The cross-sectional area of the corrugated girders is then calculated by Equation 3.27 with the geometrical parameters defined in Figure 3.6.

$$r_c = \frac{a_1 + a_2}{a_1 + a_4} \tag{3.26}$$

$$A_{corrugated} = t_{fu} \cdot b_{fu} + r_c \cdot t_w \cdot h_w + t_{fo} \cdot b_{fo} \tag{3.27}$$

Further, the contribution from pavement, the weight of crash barriers, and form-work need to be considered. The load from form-work is added as a distributed load multiplied by the width of the bridge plus two times the height of the concrete deck, while the load from the crash barriers is considered as an extra line load.

Table 3.11: A summary of the considered permanent loads for a steel and concrete composite road bridge.

Construction part	**Weight**	**Reference**
Steel	78 kN/m^3	SS-EN 1991-1-1, Table A.4
Stainless steel	78 kN/m^3	SS-EN 10088-1:2014, Table E.2
Reinforced concrete	25 kN/m^3	SS-EN 1991-1-1, Table A.1
Unhardened concrete	26 kN/m^3	SS-EN 1991-1-1, Table A.1
Pavement	23 kN/m^3	Krav Brobyggande, Section B.3.1.1
Form-work	0.5 kN/m^2	Assumed
Crash barrier	0.5 kN/m	Assumed

3.4.3 Shrinkage Load

Shrinkage is defined in SS-EN 1992-1-1, Section 3.1.4 and further described in Section 3.2.1. As mentioned before, the shrinkage is divided into two parts: autogenous and drying shrinkage. The total shrinkage strain, ε_{cs}, is used to calculate the shrinkage force, F_{cs}, acting in the center of gravity of the concrete plate, see Equation 3.28.

$$F_{cs} = \varepsilon_{cs} \cdot E_{c,eff} \cdot A_c \tag{3.28}$$

where A_c is the concrete area and $E_{c,eff}$ is the effective modulus of elasticity defined in Equation 3.29.

$$E_{c,eff} = \frac{n_0}{n_{L,sc}} E_{cm} \tag{3.29}$$

where n_0 is the modular ratio defined in Section 3.2.2 and $n_{L,sc}$ is the modular ratio for shrinkage calculated using Equation 3.17 with ψ_L=0.55 and $\varphi(t, t_0)$ as the final creep coefficient for shrinkage.

3.4.4 Traffic Load

The traffic load acting on a bridge is dependent on the composition, density, frequency, and weight of the vehicles. This is taken into account by using different load models specified in SS-EN 1991-2. In SS-EN 1991-2, Section 4.3.1, four different load models are specified and more described in Table 3.12.

Table 3.12: Specification of the load models defined in SS-EN1991-2.

Load model	**Explanation**
Load model 1	Considering concentrated loads and an uniformly distributed load for general and local verification.
Load model 2	Considering the dynamic effects of normal traffic from a single axle load.
Load model 3	Considering special vehicles of general and local verification.
Load model 4	Considering possible crowds for general verification.

In this master's thesis, Load Model 1 will be considered in the global analysis while Load Model 2 will be used for local checks of welds. Load Model 1 consists of a double axle concentrated load and a uniformly distributed load. The concentrated load is defined as $\alpha_Q \cdot Q_k$ for each axle ($0.5\alpha_Q Q_k$ for each wheel) and placed in predefined lanes of three meters in which the bridge is divided, see Figure 3.7. The lines are numbered 1, 2 etcetera where the first one is giving the most unfavorable effect. The uniformly distributed load is defined as $\alpha_q q_k$ and applied in the unfavorable sections. Values of the different axle loads and distributed loads are defined in Table 3.13 for the different lanes (SS-EN 1991-2, Table 4.2). The adjustment factors α_Q and α_q are defined in Table 3.14.

Table 3.13: Specification of the axle, Q_{ik}, and distributed loads, q_{ik}, for the different lanes.

Lane number	q_{ik} $[kN/m^2]$	Q_{ik} **[kN]**
Lane 1	9	300
Lane 2	2.5	200
Lane 3	2.5	100
Other areas	2.5	0

Table 3.14: Specification of the adjustment factors in load model 1.

Factor	**Value [-]**	**Reference**
α_{Q1}	0.9	TSFS 2018:57, Section 11 3 §
α_{Q2}	0.9	TSFS 2018:57, Section 11 3 §
α_{Q3}	0	TSFS 2018:57, Section 11 3 §
α_{q1}	0.8	TSFS 2018:57, Section 11 3 § & Krav Brobyggande, Section B.3.2.1.3
α_{qi}	1.0	TSFS 2018:57, Section 11 3 §
α_{qr}	1.0	TSFS 2018:57, Section 11 3 §

The squares representing the wheels of the vehicle have a side of 0.4 meters (SS-EN 1991-2, Section 4.3.2) and the distance in both transverse and longitudinal direction are two meters.

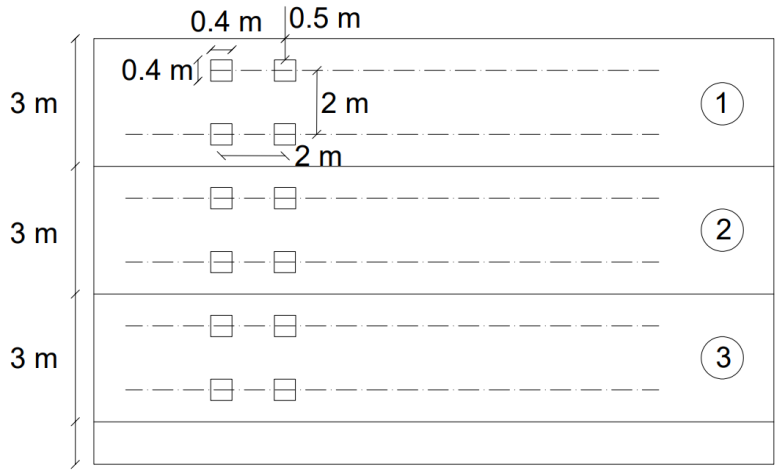


Figure 3.7: Visualization of Load Model 1.

Load model 2 is defined in SS-EN 1991-2, Section 4.3.3 and considers a single axle load $\beta_Q Q_{ak}$ where Q_{ak} equals 400 kN ($0.5\beta_Q Q_{ak}$ for each load). The factor β_Q is recommended to be equal to α_{Q1}. The wheels are represented by rectangles with sides of 0.35 meters and 0.60 meters and a c-c distance of 2 meters. See Figure 3.8.

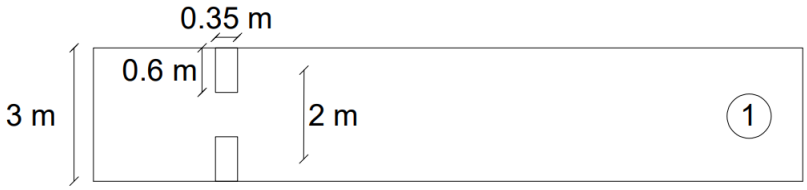


Figure 3.8: Illustration of Load Model 2.

Further, for the Fatigue Analysis, five different load models are described in SS-EN 1991-2, Section 4.6 and more specified in Table 3.15. In this master's thesis, Load Model 3 is considered since it is a road bridge.

Table 3.15: Specification of load models for fatigue by traffic load.

Load model	**Explanation**
Load model 1	Same configuration as Load model 1 specified in Table 3.12 with axle load 0.7 Q_{ik} and distributed load $0.3q_ik$.
Load model 2	Consist of different ideally decided lorries which separately should be considered while travelling along a lane to get the maximum and minimum stress.
Load model 3	Consist of two sets of 4-wheel vehicles with c-c distance 6 meter.
Load model 4	Considers the probability of different standard lorries to travel pass the bridge.
Load model 5	Is based on realistic data recorded at site.

Load Model 3 is defined with two sets of 2-axle loads. Each axle has a weight of 120 kN and the squares representing the wheels have a side of 0.4 meters See Figure 3.9.

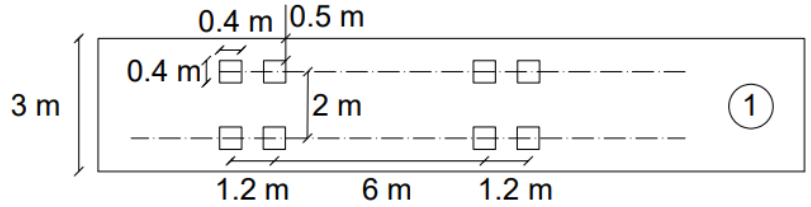


Figure 3.9: Illustration of Load Model 3 for Fatigue Analysis.

3.4.5 Acceleration and Braking

Actions from braking and acceleration are defined in SS-EN 1991-2, 4.4.1. The contributing force is equal for both acceleration and braking, however they are defined positive or negative, and it is applied horizontally in the longitudinal direction at the top of the pavement. The characteristic value is calculated using Equation 3.30 with a maximum and minimum value according to Equation 3.31.

$$Q_{k,acc} = 0.6 \cdot \alpha_{Q1} \left(2Q_{1k}\right) + 0.1 \cdot \alpha_{q1} \cdot q_{1k} \cdot w_1 L \tag{3.30}$$

$$180\alpha_{Q1}(kN) \leq Q_{k,acc} \leq 900(kN) \tag{3.31}$$

where L is the length of the deck and w_1 is the width of the lane. The loads q_{1k} and Q_{1k} as well as the adjustment factors are defined from Load model 1, more described in Section 3.4.4.

3.4.6 Wind Load

The wind load is considered for two different cases, when it is acting on the structure as well as on the vehicles and when it is only acting on the structure. However, in the global analysis, the contribution from wind is neglected since the magnitude is small compared to other loads. Further, the dynamic effects are not considered in the calculations. The wind force can be calculated according to Equation 3.32. d_{tot} is the total height of the bridge plus eventual crash or vehicle height.

$$F_w = C \cdot d_{tot} \tag{3.32}$$

$$C = q_p \cdot c_{f,x} \tag{3.33}$$

where the force coefficient, $c_{f,x}$ equals to c_{fx0} according to Equation 8.1 in SS-EN 1991-1-4. c_{fx0} is determined through figure 8.3 in SS-EN 1991-1-4 and depends on the total height and width of the bridge. Further, the characteristic velocity pressure, q_p, can be determined from Table 7.1 in TSFS 2018:57 and depends on the height over ground level as well as terrain type and the reference wind velocity. The reference wind velocity, v_b, is chosen according to Figure 7.1 in TSFS 2018:57 and depends on where the bridge is located in the country.

3.4.7 Temperature Load

The temperature load in a composite structure depends on two parts; a uniform temperature component and a linear temperature component (SS-EN 1991-1-5, Chapter 6). It can also both lead to shortening and extension of the bridge. The uniform temperature component is dependent on the max- and minimum temperatures arising in the bridge. This is taken into consideration by using the maximum and minimum ambient temperatures at site. However, there could also be differences in the uniform temperature component between different parts of the structure. This is regulated in (SS-EN 1991-1-5, Section 6.1.6) where the difference is set to $\Delta T_{c,s} = 15°C$. Further, the maximum, T_{max}, and minimum, T_{min}, ambient temperatures are presented in TSFS 2018:57, 8 Ch 2§ and can be used to calculate the maximum temperature component for shortening ($\Delta T_{N,sho}$) and extension ($\Delta T_{N,ext}$), see Equation 3.34 and Equation 3.35 (Vayas and Iliopoulos, 2013, Equation 4.16 and 4.17).

$$\Delta T_{N,sho} = T_0 - T_{e,min} \tag{3.34}$$

$$\Delta T_{N,ext} = T_{e,max} - T_0 \tag{3.35}$$

where,

$T_{e,min}$ $= T_{min} + 4°C$, the corresponding minimum temperature
$T_{e,max}$ $= T_{max} + 4°C$, the corresponding maximum temperature
T_0 $= 10°C$ (SS-EN 1991-1-5, Annex A)

The different parts of the uniform temperature component is then combined to get the two worst cases which is governing for shortening and extension. This is done according to Equation 3.37 by calculating the strains arising in the structure. Further, the temperature force, F_{temp}, can be calculated using Equation 3.36 where ϵ_{temp} is the temperature strain, α_i the coefficient of linear expansion and ΔT_i the temperature difference. The temperature force, F_{temp}, is applied in the center of gravity of the steel section.

$$F_{temp} = \varepsilon_{temp} \cdot E_s \cdot A_s \tag{3.36}$$

$$\varepsilon_{temp} = \sum \alpha_i \cdot \Delta T_i \tag{3.37}$$

The worst-case scenarios giving the largest strains in the material are when the temperature component for shortening, $\Delta T_{N,sho}$, (which is negatively defined) are combined with a negative temperature difference, $\Delta T_{c,s}$, and when the temperature component for extension, $\Delta T_{N,ext}$, is combined with the positive temperature difference, $\Delta T_{c,s}$. The combination is done according to Equation 3.38 and 3.39.

$$\varepsilon_{max,1} = (\alpha_{ss} - \alpha_c) \cdot \Delta T_{N,ext} + \alpha_{ss} \cdot \Delta T_{c,s} \tag{3.38}$$

$$\varepsilon_{max,2} = -(\alpha_{ss} - \alpha_c) \cdot \Delta T_{N,sho} - \alpha_{ss} \cdot \Delta T_{c,s} \tag{3.39}$$

Values of the linear expansion coefficients are stated in Table 3.16 (SS-EN 1991-1-5, Table C.1). However, for composite bridges made of carbon steel and concrete, the coefficient of linear expansion of the steel can be set to $10 \cdot 10^{-6}$ $1/°C$ meaning that both materials have the same coefficient. Therefore, effects caused by the restraint due to differences in expansion and contraction can be neglected.

Table 3.16: Coefficient of linear expansion for different materials.

Material	α_i $[10^{-6}/°C]$
Concrete, α_c	10
Carbon steel, α_s	12
Duplex stainless steel, α_{ss}	13

3.5 Structural Analysis

The analysis in Ultimate Limit State is, as earlier described, divided into two phases; construction phase and service phase. During the analysis of the main girders in the construction phase, only self-weight is considered consisting of the steel girders, unhardened concrete, form-work and crash barriers. Since the concrete is not yet hardened, there are no composite action. Therefore, there is an imminent risk of buckling of the girders and the crossbeams are in this stage assumed to carry all the horizontal loads, see Section 3.6.4. Therefore, the design of the crossbeams also includes wind load and unintended inclination. For the analysis in Ultimate Limit State during service phase the considered loads are self-weight (including steel beams, concrete deck, and crash barriers), traffic (including acceleration/braking), shrinkage, and temperature. However, for calculations of shear, only self-weight and traffic load will contribute.

The traffic load is for service phase considered in both transverse and longitudinal direction. In the transverse directions it is placed to cause maximum moment in the structure, see Figure 3.10. Since each lane has two point loads and a distributed load, the load application will depend on the number of lanes. If only one lane fit in half the bridge width, only the axle- and distributed load for lane 1 in Load Model 1 will be added to the structure. However, if more than one lane fit, more axle- and distributed loads will be added as Load Model 1 describes. The contribution from the transverse direction is then added to the longitudinal direction as two point loads and a distributed moment.

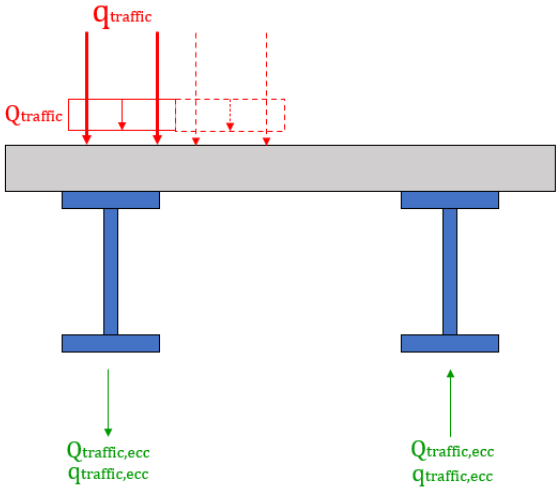


Figure 3.10: Considered loads in Ultimate Limit State during service life in transverse direction causing forces in the main girders.

Further, the total considered loads in the longitudinal direction for Ultimate Limit State during service life are visualized in Figure 3.11. The eccentric traffic loads from the transverse directions are marked with a green colour in the figure and the point loads are moved along the bridge to get the most unfavorable effect. The

shrinkage force, F_{cs}, is applied in the centre of gravity of the concrete. However, since the shortening due to shrinkage is constrained by the steel, tensile stresses will occur in the concrete with equal magnitude as the shrinkage force. Also a moment due to eccentricity will be applied. The temperature force, F_{temp}, is applied in the centre of gravity of the steel. In the same way as for shrinkage, the temperature load will also cause moment due to eccentricity. Since the temperature load can be both positive and negative, the worst case scenario is when all the loads creates a moment in the same direction. Additionally, the acceleration/braking force applied at the top of the concrete deck causes a moment due to eccentricity.

The analysis in Serviceable Limit State includes self-weight, shrinkage, traffic, acceleration/braking, and temperature. In the same way as for Ultimate Limit State in service phase, the traffic load is considered in both longitudinal and transverse direction. Further, since the bridge is divided into sections with varying dimensions of the cross-section there are several parameters used in the Serviceable Limit State calculations. Therefore, the used values of the self-weight, moment of inertia, neutral axis, and sectional modulus will be average values calculated from all the segments. For the Fatigue analysis, only the traffic load is considered according to Load Model 3, see Section 3.4.4. This is because one is interested in the difference of stresses in the structure when calculating fatigue.

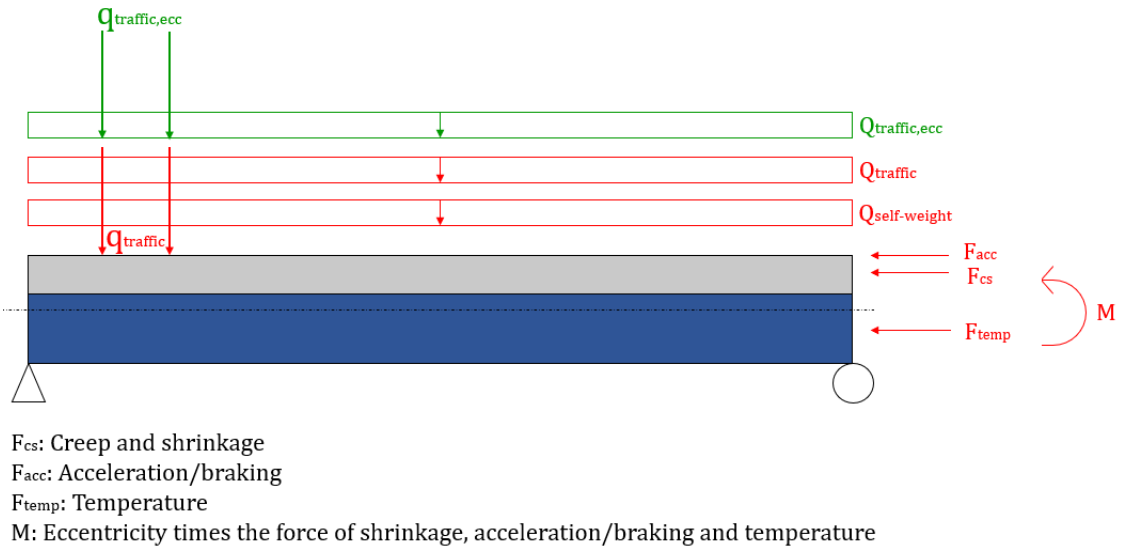


Figure 3.11: Considered loads for Ultimate Limit State during service phase in longitudinal direction.

3.5.1 Bending Moment

The design moment in longitudinal direction is calculated using the Superposition Method meaning that the moment curves from the different load-types are summed to get the resulting moment. Different moment curves for different load-types are visualized in Figure 3.12. The moment curve from distributed loads is calculated with Equation 3.40 where Q_k is the characteristic distributed load, L the span length, and

x the section where the moment is to be calculated for. Further, the moment contribution from the traffic point loads is calculated using the Influence Line Method. For the bending moment the influence line diagram is plotted for the x-coordinate closest to the mid of each segment and the traffic point loads are moved along the bridge to find the highest moment for the considered x-coordinate.

The distributed moments from shrinkage, temperature and acceleration are already distributed moments and applied evenly as Figure 3.12 indicates. Since the temperature load and acceleration/braking load can be both positive and negative, also the moment due to eccentricity can have different signs.

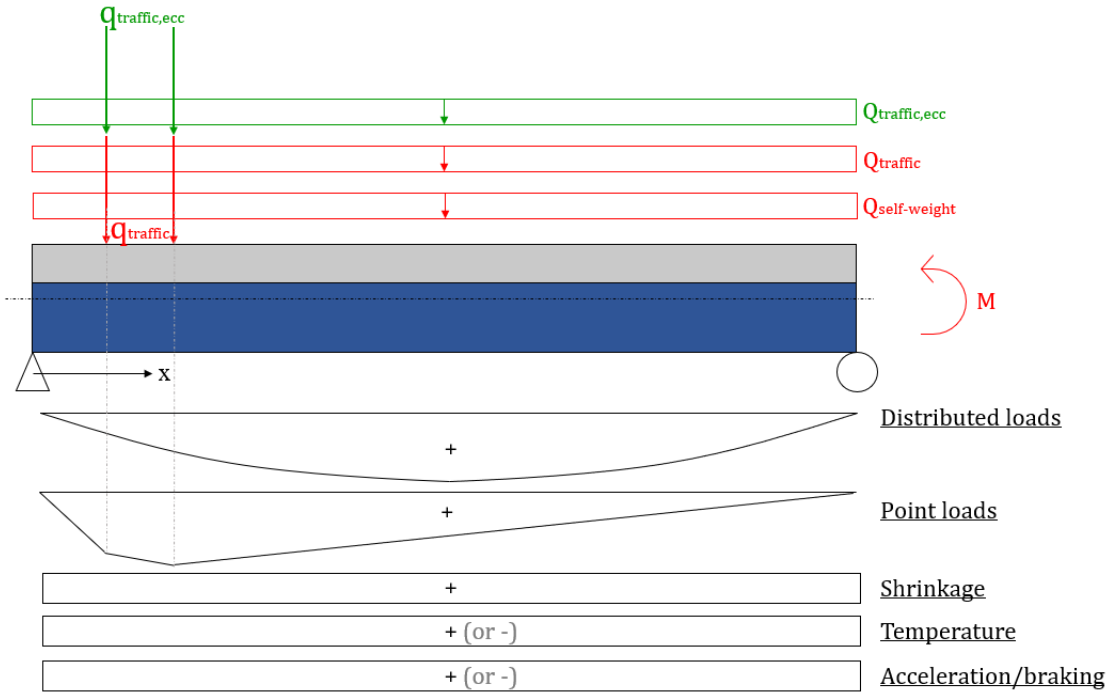


Figure 3.12: Moment diagrams for the considered loads in Ultimate Limit State during service phase.

$$M_{Ek,distr} = \frac{Q_k}{2} \cdot \left(Lx - x^2\right) \tag{3.40}$$

The bridge is, as earlier described, divided into segments. The different segments can have varying cross-section and will have different resulting moment. Therefore, during the analysis of the main girders in Ultimate Limit State for both construction and service phase, the main girders are verified for each segment. By this, it will be ensured that the whole bridge achieve the capacity requirements. This is also used for the calculations in Fatigue Limit State, but here only the actions from Fatigue Traffic Load 3 is considered. Since the fatigue load model consist of four point loads, the moment contribution is calculated by using the Influence Line Method, as for Traffic Load Model 1. When the moment contribution from the different load-types are calculated, the values are added together and at the same time the load combination is done. Therefore, the characteristic moments are multiplied with the

different partial safety factors for the considered states. More described in Section 3.4.1.

3.5.2 Shear Force

The Superposition Method is also used for calculations of the design shear force. The different shear diagrams for different load-types are visualized in Figure 3.13. The shear contribution from distributed loads are calculated using Equation 3.41, while the contribution from point loads are calculated using the Influence Line Method, more described in Section 3.5.1.

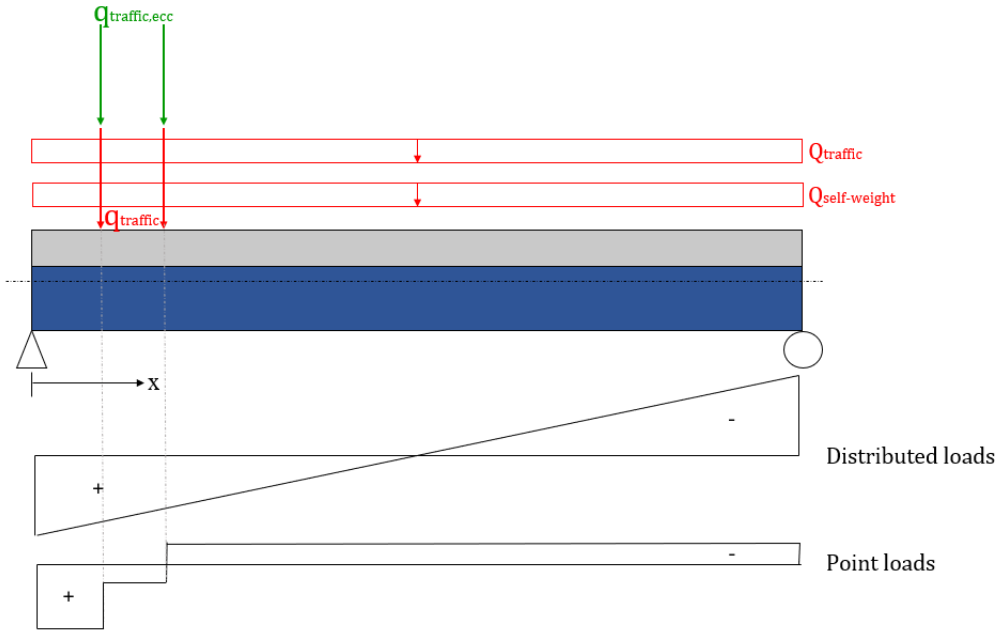


Figure 3.13: Shear diagrams for the considered load-types in Ultimate Limit State during service phase.

$$V_{Ed,distr} = q_d \cdot \left(\frac{L}{2} - x\right) \tag{3.41}$$

q_d is the distributed load, L the length of the span and x the section where the shear is to be calculated defined from the left support. As for the moment verification, the shear force is for Ultimate Limit State verified in the different segments. Additionally, the shear force is calculated in the different sections using Fatigue Load Model 3.

3.5.3 Maximum Deflection

During the calculation of deflection, some simplifications are made. Due to the different sections having different widths and thicknesses, parameters such as stiffness and self-weight varies along the beam. However, the difference is not considered to be of a large magnitude and therefore an average value of the varying parameters

of the beam is calculated and used for deflection calculations. By this, simple equations can be used to calculate the maximum deflection from uniformly distributed loads (Equation 3.42), point loads (Equation 3.43), and moment (Equation 3.44).

$$\delta = \frac{5QL^4}{384EI} \tag{3.42}$$

$$\delta = \frac{qaL^2}{48EI} \tag{3.43}$$

$$\delta = \frac{ML^2}{16EI} \tag{3.44}$$

where Q is the distributed load in $[N/mm]$, q is the point load in $[N]$, and L is the length of the bridge $[mm]$. a is the shortest distance from the point load to the edge of the beam $[mm]$, E the modulus of elasticity of steel defined in Table 3.2 for carbon steel and in Table 3.3 for stainless steel. I is the moment of inertia $[mm^4]$ and M is the moment $[Nmm]$.

3.5.4 Crossbeams

In the construction phase before the concrete is hardened, the main girders and crossbeams need to be able to restrain also the horizontal forces coming from wind and unintended inclination of the main girders. Therefore, when designing the crossbeams, these are the loads to consider, see Figure 3.14. The wind force, H_w, acting on each crossbeam is calculated by dividing the total force of the distributed load by the number of crossbeams. The crosbeams are placed in the centre of the steel section and no moment due to the wind is generated before the concrete is cast. After casting, the moment due to eccentricity should be considered, taken as the wind force, H_w, times the eccentricity, e_w. The unintended inclination is due to lateral displacement of the compressed flange. The corresponding horizontal force, H_i, can be calculated according to Equation 3.45.

$$H_i = \sqrt{0.5\left(1 + \frac{1}{m}\right)} \cdot \frac{N_{Ed}}{100} \tag{3.45}$$

where N_{Ed} is decided by the maximum moment, M_{Ed}, divided by the total height of the steel girder and m is the number of elements braced, here $m = 2$. The crossbeam and the contributing part of the stiffener should also be able to handle the moment due to the eccentric load application, taken as the force, H_i, times the eccentricity, e_i.

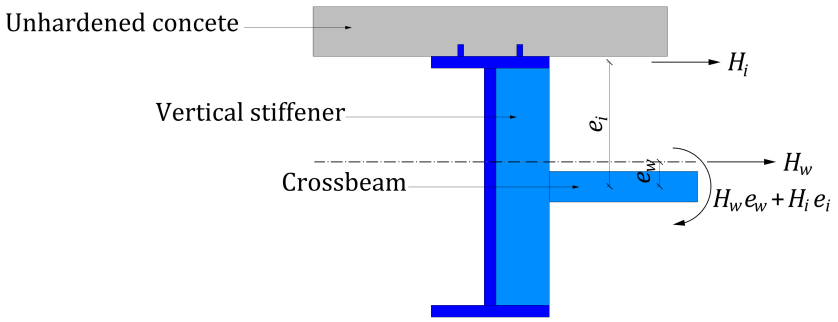


Figure 3.14: Considered loads for the design of cross beams.

Additionally, the end crossbeams should be designed to handle the total uplifting force of the bridge in cases where the permanent supports is not utilized, for example during maintenance of the supports. The shear force that the end crossbeams should be design for should then be the maximum shear force divided by the number of support points on the beam.

3.6 Design Verification

To certain the capacity of the bridge, four design states are verified. The first is the Ultimate Limit State during construction, before the hardening of the concrete. The second is the final Ultimate Limit State during the service life of the bridge. The third is the Serviceability Limit State and the fourth is the Fatigue Limit State with respect to the service life of the bridge. The different utilization verifications that need to be performed for each state are summarized in Table 3.17.

Table 3.17: Capacity checks per design state.

State	**Checks**
ULS construction	Bending moment capacity of main girders Shear capacity of main girders Interaction of shear and bending moment Capacity of crossbeams Weld capacity
ULS service	Bending moment capacity composite section Shear capacity of composite section Interaction of shear and bending moment Capacity of shear studs Capacity of end crossbeams Weld capacity
SLS	Deflection of composite section
FAT	Service life of weld details with respect to fatigue

3.6.1 Bending Moment Capacity

In the construction phase before the hardening of the concrete, the bending moment capacity depends solely on the steel structure. The capacity, $M_{B,Rd}$, is calculated according to Equation 3.47 and stated in SS-EN 1993-1-1, Section 6.3.2.1 for flat webs and SS-EN 1993-1-5, Annex D for corrugated webs. The capacity is verified by the stress relation in Equation 3.46, where M_{Ed} is the moment to resist.

$$f_{yd} = \frac{M_{B,Rd}}{W_y} \geq \frac{M_{Ed}}{W_y} \tag{3.46}$$

$$M_{B,Rd} = \begin{cases} \chi_{LT} W_y \frac{f_y}{\gamma_{M1}} & \text{for flat web} \\ min \begin{cases} \frac{b_{fu} t_{fu} f_{y,r}}{\gamma_{M0}} \left(h_w + \frac{t_{fo}+t_{fu}}{2} \right) \\ \frac{b_{fo} t_{fo} f_{y,r}}{\gamma_{M0}} \left(h_w + \frac{t_{fo}+t_{fu}}{2} \right) \\ \frac{b_{fo} t_{fo} \chi_{LT} f_y}{\gamma_{M1}} \left(h_w + \frac{t_{fo}+t_{fu}}{2} \right) \end{cases} & \text{for corrugated web} \end{cases} \tag{3.47}$$

with χ_{LT} being the lateral-torsional buckling described in Section 3.1.2 if the slenderness λ_{LT} is larger than 0.4, otherwise, $\chi_{LT} = 1$ and the safety factor γ_{M0} should be used instead of γ_{M1}. For beams with corrugated webs, the bending is only taken by the flanges, as described in Section 2.3.2. Therefore the web contribution should be neglected when calculating the sectional modulus W_y. $f_{y,r}$ is a reduced yield strength due to transverse stress in the flanges. Since no transverse moment is accounted for, $f_{y,r}$ equals the yield strength, f_y. The geometrical parameters are defined in Figure 3.6.

During the service phase, the linear elastic bending moment capacity of the composite section should be verified by the design strength of each part of the section (SS-EN 1994-2, Section 6.2.1.5). For this case, the stress at the top of the concrete is verified against the design compressing strength $f_{cd} = f_{ck}/\gamma_c$ (SS-EN 1992-1-1, Section 3.1.6). For the steel parts, the stress at the bottom of the lower flange is checked against the design yield strength $f_{yd} = f_y/\gamma_{M0}$. Additionally, the shrinkage and temperature stresses added to the stress from the bending moment can result in tensile stresses in the concrete layer. The tensile stress is then verified against the design strength of the reinforcement, $f_{sd} = f_{sk}/\gamma_s$ (SS-EN 1992-1-1, Section 2.4.2.4). Strength properties and safety factors are previously listed in Section 3.1 and 3.2.

3.6.2 Shear Capacity

When calculating the shear capacity of a composite beam, only the contribution from the steel section is considered (SS-EN 1994-2, Section 6.2.2). Therefore, the shear capacity, V_{Rd}, is calculated similarly for both the construction and service phase in Ultimate Limit State. Regardless of the web shape, the shear capacity should fulfill the first criteria in Equation 3.48 (SS-EN 1991-1-1, Section 6.2.6) (SS-EN 1991-1-5, Section 5.2). The capacity of flat web beams should additionally be

limited by the second criteria in Equation 3.48 as the flanges are also contributing and the buckling reduction factor for flat webs includes post-buckling capacity.

$$\begin{cases} V_{Ed} \leq V_{Rd} \\ \\ \text{Additionally, for flat web:} \quad V_{Rd} \leq \frac{\eta f_y h_w t_w}{\sqrt{3}\gamma_{M1}} \end{cases} \tag{3.48}$$

where η is a factor with recommended value 1.20 and the geometrical parameters are defined in Figure 3.6. The safety factor γ_{M1} is stated in Table 3.4. For flat web girders, the shear capacity is the sum of the web capacity, $V_{w,Rd}$, and the flange contribution, $V_{f,Rd}$, see Equation 3.49 (SS-EN 1991-1-5, Section 5.2 and 5.4). For beams with corrugated webs only the web contribution, $V_{w,Rd}$, is considered.

$$V_{i,Rd} = \begin{cases} V_{w,Rd} = \chi \frac{f_y h_w t_w}{\sqrt{3}\gamma_{M1}} & \text{web} \\ \\ V_{f,Rd} = \frac{b_f t_f^2 f_y}{c\,\gamma_{M1}} \left(1 - \left(\frac{M_{Ed}}{M_{f,Rd}}\right)\right) & \text{flange} \end{cases} \tag{3.49}$$

where the factor χ equals χ_w or χ_c for flat or corrugated web, respectively. The factor χ is only applied if the criteria described in Section 3.1.3 is not fulfilled. Then it is calculated as described in the same section. c is a factor dependent on the yield strength, cross-section dimensions, and distance between the rigid end posts and $M_{f,Rd}$ is the moment capacity when considering only the flange contribution. f_y is the yield strength for steel and M_{Ed} is the moment contribution from the loads acting on the structure.

3.6.3 Interaction of Moment and Shear

For girders with corrugated web, the shear force is only restricted by the web, while the bending is taken by the flanges, as described in Section 2.3.2. Therefore, verification of interaction between shear and bending is only needed for flat web girders when Equation 3.50 is not fulfilled (SS-EN 1994-2, Section 6.2.2.4) (SS-EN 1993-1-5, Section 7.1).

$$V_{Ed} \leq 0.5 V_{Rd} \tag{3.50}$$

where V_{Ed} is the shear contribution from applied loads and V_{Rd} is the shear capacity. The steel girders should be verified against the criteria in Equation 3.51 if the plastic flange moment capacity, $M_{f,Rd}$, is less than the design moment, M_{Ed} (SS-EN 1993-1-5, Section 7.1).

$$\frac{M_{Ed}}{M_{pl,Rd}} + \left(1 - \frac{M_{f,Rd}}{M_{pl,Rd}}\right)\left(2\frac{V_{Ed}}{V_{w,Rd}} - 1\right)^2 \leq 1 \tag{3.51}$$

In the construction phase, $M_{f,Rd}$ is the effective plastic capacity of the flanges and $M_{pl,Rd}$ is the plastic moment capacity considering the effective area of the flanges and the full area of the web, regardless of the cross-section class. In the service phase also the capacity contribution of the effective area of the concrete deck is added to $M_{f,Rd}$ and $M_{pl,Rd}$. In both phases, $V_{w,Rd}$ is the shear resistance of the web.

3.6.4 Capacity of Crossbeams

The crossbeams should be verified against the axial force from the horizontal contribution of wind and unintended inclination and moment from the load positions of the horizontal forces, as described in Section 3.5.4. The capacity should be verified according to Equation 3.52 where M_{Ed} and N_{Ed} are the contributions from loads on the structure and M_{Rd} and N_{Rd} are the design capacities.

$$\frac{N_{Ed}}{N_{Rd}} + \frac{M_{Ed}}{M_{Rd}} \leq 1 \tag{3.52}$$

M_{Rd} is calculated according to the first equation in Equation 3.47 and N_{Rd} according to Equation 3.53.

$$N_{Rd} = \frac{\chi A f_y}{\gamma_{M1}} \tag{3.53}$$

where χ is a reduction factor due to buckling. It is calculated using the same equation as χ_{LT} in Section 3.1.2, with the slenderness parameter, λ, depending on the area, yield strength, and critical axial force. If λ is less than 0.2, $\chi = 1$ and the safety factor γ_{M0} should be used instead of γ_{M1}, defined in Table 3.4. A is the cross-sectional area and f_y is the yield strength.

3.6.5 Capacity of Shear Studs

The shear capacity of the studs is calculated according to SS-EN 1994-2, Section 6.6.3.1 where the smallest of Equation 3.54 and 3.55 are used. To calculate the required number of studs, the horizontal forces from temperature, shrinkage, and acceleration are combined and divided by the stud capacity. The number is then rounded up to get an even number of studs.

$$P_{Rd} = \frac{0.8 \cdot f_u \cdot \pi \cdot d^2/4}{\gamma_V} \tag{3.54}$$

$$P_{Rd} = \frac{0.29 \cdot \alpha \cdot d^2 \cdot \sqrt{f_{ck} \cdot E_{cm}}}{\gamma_V} \tag{3.55}$$

where f_u is the specified ultimate tensile strength of the material of the stud, but not greater than 500 MPa. d is the diameter of the shank of the stud which should be between 16 and 25 millimeters and the partial factor γ_V is recommended to be equal to 1.25. Further, f_{ck} is the characteristic cylinder compressive strength of the concrete and E_{cm} is the modulus of elasticity of concrete.

3.6.6 Weld Capacity

The capacity of the welds is verified according to Equation 3.56 to 3.57.

$$\sigma_i = \sqrt{\sigma_\perp^2 + 3\left(\tau_\perp^2 + \tau_\parallel^2\right)} \leq \frac{f_u}{\beta_w \cdot \gamma_{M2}} \tag{3.56}$$

$$\sigma_{\perp} \leq \frac{0.9 \cdot f_u}{\gamma_{M2}} \tag{3.57}$$

where $\sigma_{\perp}$ is the perpendicular normal stress and $\tau_{\perp}$ is the shear stress perpendicular to the axis of the weld. f_{ut} is the nominal ultimate tensile strength of the weaker part of the joint and β_w is a correlation factor chosen from Table 4.1 in SS-EN 1993-1-8 for carbon steel and Section 6.3 (1) in SS-EN 1993-1-4 for stainless steel and depends on steel grade. The partial safety factor γ_{M2} is defined in Table 3.4.

3.6.7 Deflection

The deflection of the road bridge should not be larger than L/400 in both transverse and longitudinal direction (Krav Brobyggande, Section B3.4.2.2). The deflection check is done with regards to the traffic load. The resulting deflection due to permanent loads are evaluated but instead handled by an initial elevation of the beam with the same magnitude, resulting in zero deflection when installed.

3.6.8 Fatigue

For fatigue verification, an additional partial factor, γ_{Mf}, should be used as well as the partial safety factor for equivalent constant amplitude stress, γ_{Ff}. The factor γ_{Mf} can be determined according to Table 3.18 and depends on which method to be used for the fatigue analysis. There are two methods available, the damage tolerant method and the safe life method. The damage tolerant method assumes that there will be regular inspections and maintenance of the welds while the safe life method is not requiring any regular inspections. One also needs to decide whether there is a low or high consequence of failure (SS-EN 1993-1-9, Table 3.1). The factor γ_{Ff} is set to 1.0.

Table 3.18: Values of the partial safety factor for fatigue strength, γ_{Mf}.

Method	Low consequence failure	High consequence failure
Damage tolerant method	1.0	1.15
Safe life method	1.15	1.35

For the fatigue verification, the λ-method is used according to SS-EN 1993-1-9. The used equations are the same for carbon and stainless steel as well as for flat and corrugated webs. However, girders with corrugated webs have zero normal stresses in the web. In the λ-method, a final λ-factor is calculated through Equation 3.58 with the restriction that it should be less than λ_{max} (Equation 3.59).

$$\lambda = \lambda_1 \cdot \lambda_2 \cdot \lambda_3 \cdot \lambda_4 \tag{3.58}$$

$$\lambda_{max} = \begin{cases} 2.5 - 0.5 \cdot \frac{L-10}{15} & \text{if } 10\text{m} \leq L \leq 25\text{m, in span section} \\ 2 & \text{if } 25\text{m} \leq L \leq 80\text{m, in span section} \\ 1.8 & \text{if } 10\text{m} \leq L \leq 30\text{m, at support section} \\ 1.8 + 0.9 \cdot \frac{L-30}{50} & \text{if } 30\text{m} \leq L \leq 80\text{m, at support section} \end{cases} \tag{3.59}$$

λ_1 is determined through Figure 9.5 in SS-EN 1993-2 and summarized in Equation 3.60. The value is dependent on the critical length and for a simply supported beam, the critical length can be set to the span length L for bending moment calculations and shear at support. However, for shear in span, the critical length equals 0.4 times the span length.

$$\lambda_1 = \begin{cases} 2.55 - 0.7 \cdot \frac{L-10}{70} & \text{in span section} \\ 2 - 0.3 \cdot \frac{L-10}{20} & \text{if } 10\text{m} \leq L \leq 30\text{m, at support section} \\ 1.7 + 0.5 \cdot \frac{L-30}{50} & \text{if } 30\text{m} \leq L \leq 80\text{m, at support section} \end{cases} \tag{3.60}$$

λ_2 is calculated through Equation 3.61, where $Q_0 = 480$ kN and $N_0 = 0.5 \cdot 10^6$. N_{Obs} is determined by Table 4.5 in SS-EN 1991-2 and depends on road and vehicle type. According to Krav Brobyggande, Q_{m1} is equal to 445 kN.

$$\lambda_2 = \frac{Q_{m1}}{Q_0} \cdot \left(\frac{N_{Obs}}{N_0}\right)^{1/5} \tag{3.61}$$

Further, the factor λ_3 is dependent on the service life of the bridge, t_{Ld}, and is calculated through Equation 3.62. The value for λ_4 is set to 1.0 according to TSFS 2018:57, Chapter 27 3§.

$$\lambda_3 = \left(\frac{t_{Ld}}{100}\right)^{1/5} \tag{3.62}$$

For the verification of fatigue, Equation 3.63 to 3.65 should be used where σ_E and τ_E are calculated by multiplying λ with the stresses from considered loads in the checked detail. σ_C and τ_C are determined through the detail category depending on the checked detail. γ_{Ff} and γ_{Mf} are partial safety factors stated in Table 3.18 where the Safe Life Method and high consequence of failure are assumed.

$$\frac{\gamma_{Ff} \cdot \Delta\sigma_E}{\Delta\sigma_C/\gamma_{Mf}} \leq 1.0 \tag{3.63}$$

$$\frac{\gamma_{Ff} \cdot \Delta\tau_E}{\Delta\tau_C/\gamma_{Mf}} \leq 1.0 \tag{3.64}$$

$$\left(\frac{\gamma_{Ff} \cdot \Delta\sigma_E}{\Delta\sigma_C/\gamma_{Mf}}\right)^3 + \left(\frac{\gamma_{Ff} \cdot \Delta\tau_E}{\Delta\tau_C/\gamma_{Mf}}\right)^5 \leq 1.0 \tag{3.65}$$

The details that need to be verified for fatigue are the longitudinal welds between the web and the flange of the main girder. The detail category for the welds is $\Delta\sigma_c$= 80 MPa for shear stress in weld and $\Delta\sigma_c$ = 100 MPa for shear stress in the web at the support section (Detail A and B1 in Figure 3.15). Further, the detail category for normal stress in the span section is $\Delta\sigma_c$ = 125 MPa (Detail B2). The

welds between the stiffeners and the web of the main girder also need to be verified for principle stresses with detail category $\Delta\sigma_c = 80$ MPa for stiffeners thickness less than 50 millimeters and $\Delta\sigma_c = 71$ MPa for stiffeners thickness between 50 and 80 millimeters (Detail C). Furthermore, the weld between stiffeners and flanges of the main girders also needs to be verified and they have detail category $\Delta\sigma_c = 80$ MPa for stiffeners thickness less than 50 millimeters and $\Delta\sigma_c = 71$ MPa for stiffeners thickness between 50 and 80 millimeters (Detail C). For the corrugated web, an additional failure mode for the longitudinal welds between web and flange can occur, see Figure 3.15 failure mode E, with detail category $\Delta\sigma_c = 100$ MPa. An additional detail will occur between the different sections due to bending stresses in the flange. The detail category is $\Delta\sigma_c = 112$ MPa. For this detail there is also a size effect, k_s that should be considered if the plate is thicker than 25 millimeters.

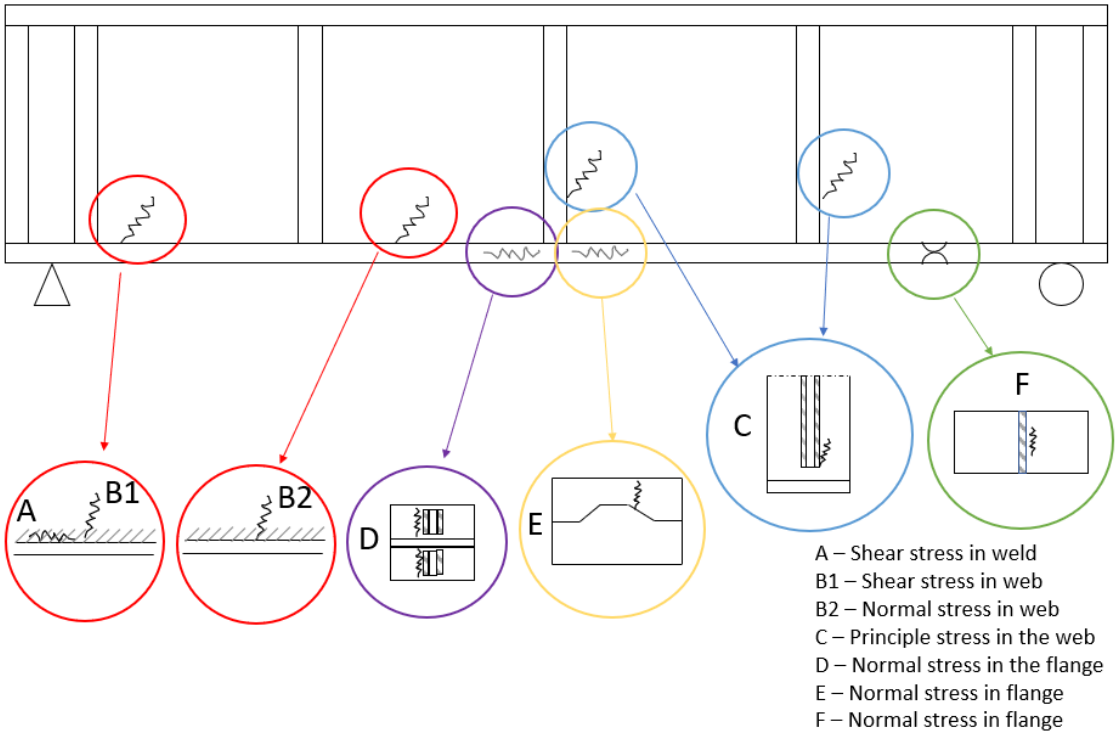


Figure 3.15: Welds that need to be verified in the Fatigue Analysis.

4

Optimization Procedure

Generally, for optimization procedures with meta-heuristic algorithms where randomization is mixed with local search, the routine is described as two separate yet connected parts: the simulation model and the optimization algorithm (Bozorg-Haddad et al., 2017). In this master's thesis, the simulation model is the Parametric Design Program presented in Section 4.1, built on the design procedure described in Chapter 3 and further visualized in Figure 4.1. For the optimization, a Genetic Algorithm is used with the specific method and parameters described in Section 4.2.

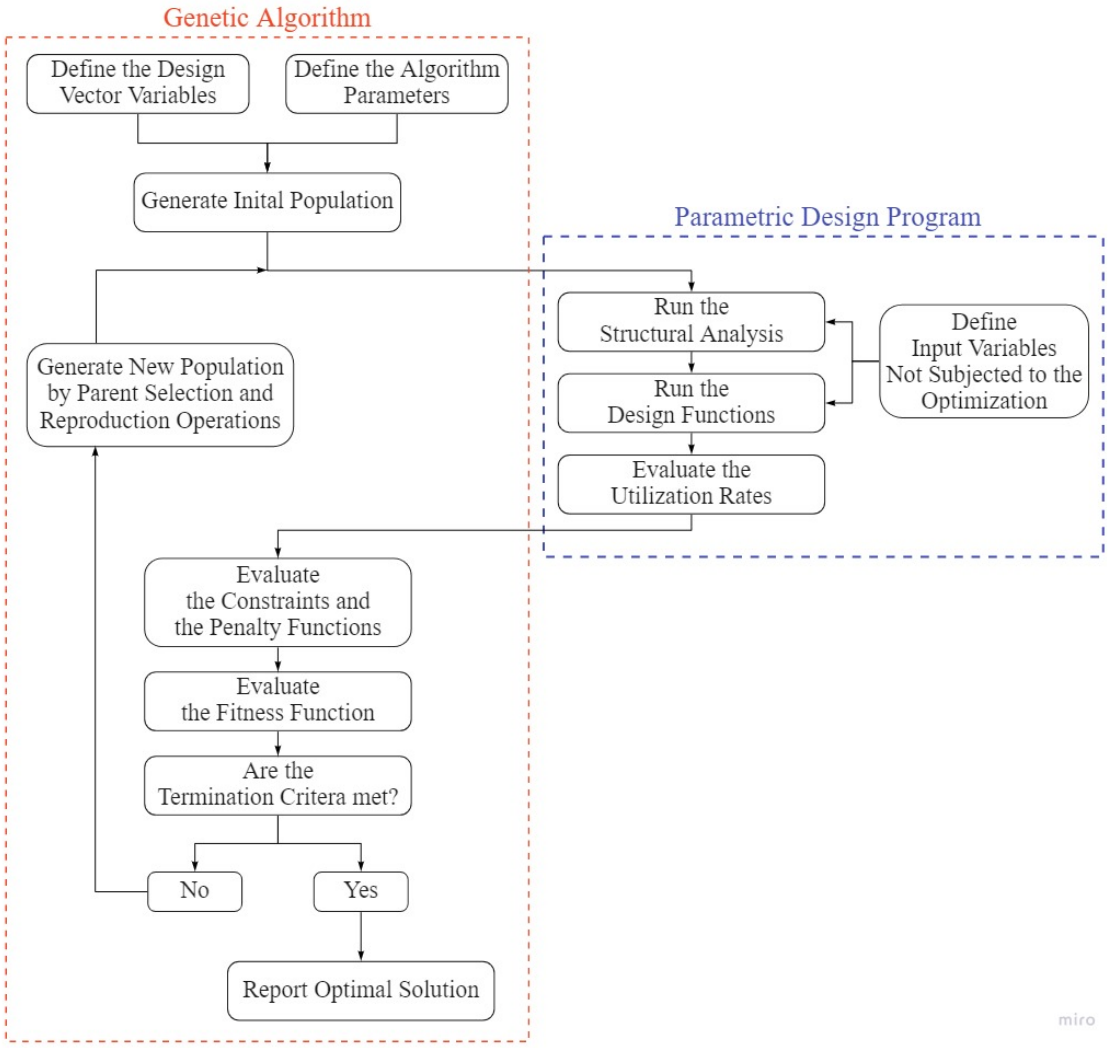


Figure 4.1: Diagram of the optimization routine.

4.1 Parametric Design Program

In this context, parametric refers to the program being valid when choosing the different design parameters, in detail described in 4.1.2. The program can be used with the genetic algorithm to optimize the design with regard to the weight of the material, Life Cycle Assessment, Life Cycle Cost, or by itself to calculate useful parameters such as the material weight, life cycle assessment, and life cycle cost of a given bridge or section. A brief outline of the program is given below, while the full parametric program, including the genetic optimization part, is found in Appendix C.

- The design parameters are defined from given design domains.
- Different design modules used in the calculations are imported, see further Section 4.1.1.
- Self-weight and temperature forces are calculated, as well as the shrinkage force and the different modular ratios for short-term, long-term, and shrinkage are calculated.
- The coordinates of the different segments are obtained and possible interference with crossbeam positions is checked.
- The following calculations and verification are done per segment:
  - The steel material parameters are generated depending on the thickness of the steel elements.
  - The cross-sectional class for the upper steel flange is evaluated.
  - The sectional parameters of the steel section are generated, with effective width reduction for all elements in cross-section class 4.
  - The governing Ultimate Limit State load combination for the construction phase is decided.
  - The resistance capacities are calculated and the section is verified against design stresses and shear.
  - Lastly, for the construction phase, the assumption that the upper flange and part of the web is in compression is checked.
  - Regarding the service phase, the effective width of the concrete is calculated and the sectional properties of the composite section are generated for long-term, short-term, and shrinkage loads.
  - The governing Ultimate Limit State load combination for the service phase is decided.
  - The design capacities are calculated and the utilization criteria verified.
  - The assumption regarding the upper flange and concrete section being in compression is verified.
  - The welds between the flanges and the web of the main girder are checked.
  - The governing Fatigue Limit State combination is calculated and the details defined in Section 3.6.8 are checked.
- For the whole bridge, the governing Service Limit State combination is calculated and the maximum deflection is calculated. An average value of the second moment of inertia is used to verified against the deflection requirement.
- The design concept of the intermediate crossbeams are decided based on the

height of the bridge, as described in Section 3.3.2. The load on the beams or trusses are calculated and the sectional parameters are chosen by the program from a list of HEA sections to achieve the required capacity.
- For the end crossbeams, the flange width and height is set depending on the height of the steel section. The thickness of the flange and web are then chosen by the program to achieve required moment and shear capacity.
- The required input data for the Life Cycle Assessment and Life Cycle Cost calculations are generated as follows:
  - The total steel weight of the plated sections and stiffeners, as well as the hot rolled sections and the weight of the weld filler material. Regarding the concrete decks, the total volume is given.
  - For carbon steels, the total painting area as well as the area around splices that require on site painting. For stainless steels, the total pickling area.
  - For corrugated webs, the gross length of the web is stated, for manufacturing cost calculations. For carbon steel the total plate grinding length.
  - For on site assembly, the number of splices as well as the number of cross beams.
  - The welding length and thickness of each weld. The welding between the potential longitudinal segments with varying cross sections is only considered if there is in fact a cross section change. This gives the program a possibility to choose less section changes than what is stated in Table 3.9.
  - The Service Life of the bridge as well as the expected Average Daily Traffic.
- Finally, the Life Cycle Assessment and Life Cycle Cost functions developed by Nissan and Woldeyohannes (2022) is called and the costs is returned.

4.1.1 Design Modules

To make the script easier to read and understand, the different functions are placed in a module. There are five modules; material functions, geometrical class functions, load functions, design functions, and structural analysis functions. The included functions in the different modules are stated in Table 4.1 to 4.5. The first module *MaterialClass* contains different material parameters for steel, concrete, pavement, and reinforcement. These functions are written as classes in python making it possible to reduce the number of input values in the functions where the class is used. The steel material parameters function contains for example the elastic modulus, yield strength, and density.

Table 4.1: Included functions in the module *MaterialClass*.

Module Name	**Included Functions**
MaterialClass	1. Steel Material Parameters 2. Concrete Material Parameters 3. Pavement Material Parameters 4. Reinforcement Material Parameters

The module *GeometricalClassFunctions* includes functions that are connected to the geometrical properties of the bridge. For example, the Sectional Properties function includes calculations of the area, the moment of inertia, and the neutral axis.

Table 4.2: Included functions in the module *GeometricalClassFunctions*.

Module Name	**Included Functions**
GeometricalClassFunctions	0. Input Modules 1. Corrugation Parameters 2. Coordinates of Longitudinal Segments 3. Sectional Properties 4. Connections (LCC Analysis) 5. Quantities (LCC Analysis) 6. Painting (LCC Analysis)

In the module named *LoadFunctions*, all the considered loads are collected. In some of the functions, the load is calculated, for example in Temperature Load, while in others the given load is only multiplied with given partial factors, for example in the Traffic Load Models.

Table 4.3: Included functions in the module *LoadFunctions*.

Module Name	**Included Functions**
LoadFunctions	0. Input Modules 1. Self-weight 2. Traffic Load: Load Model 1 (LM1) 3. Traffic Load: Load Model 2 (LM2) 4. Acceleration/ Braking Load 5. Fatigue Traffic Load: Load Model 3 (FLM3) 6. Wind Load 7. Temperature Load 8. Shrinkage Load

The *DesignFunctions* module contains all the different functions used to design both concrete and steel parts. It also contains calculations of the design resistances, such as shear resistance and weld resistance.

Table 4.4: Included functions in the module *DesignFunctions*.

Module Name	Included Functions
DesignFunctions	0. Input Modules 1. Cross-Section Class 2. Cross-Section Reduction of Parts in Cross-Sectional Class 4 (CSC 4) 3. Lateral Torsional Buckling 4. Bending Moment Capacity of Steel Girders 5. Shear Buckling 6. Shear Resistance 7. Moment and Shear Interaction 8. Shear Lag Steel 9. Shear Lag Concrete 10. Modular Ratio 11. Weld Resistance 12. Shear Stud Resistance 13. Fatigue 14. Concrete deck Design - Linear Analysis 15. Bending Moment Capacity of Composite Beam (Including Interaction) 16. Buckling (general) 17. Minimum Reinforcement in Longitudinal Direction

Lastly, by using the *StructuralAnalysisFunctions* module, the design loads are calculated for the different stages.

Table 4.5: Included functions in the module *StructuralAnalysisFunctions*.

Module Name	Included Functions
StructuralAnalysisFunctions	0. Input Modules 1. psi: Stress Distribution as Input for CSC4 Width Reduction 2. Influence Line Diagram of Simply Supported Beam (with Overhang) 3. Serviceability Limit State (Deflection) 4. Ultimate Limit State (Bending Moment and Shear) 5. Ultimate Limit State (Welds) 6. Lambda-Method (Fatigue of Welds) 7. Fatigue Limit State 8. Ultimate Limit State (Horizontal Loads) 9. Ultimate Limit State of Concrete deck (Bending)

The modules are in whole found in Appendix C.

4.1.2 Design parameters and domains

The design parameters can be divided into fixed and variable. The fixed are the parameters depending on the location and geometry of the bridge, presented in Table 4.6 to 4.10. The variable parameters are the design parameters subjected to the optimization process, presented in Table 4.11. The variable parameters can manually be selected within a given domain or randomly generated by the Genetic Algorithm as further described in Section 4.2.1. The domain can be modified to suit different cases. The domains in Table 4.11 are used for the sensibility study of the program in Section 4.3.

Table 4.6: Geometrical fixed design parameters.

Parameter	Unit	Definition
L	mm	Length of bridge, 25 m $\leq$ L $\leq$ 75 m.
SubDiv	mm	Calculation subdivision of bridge.
w	mm	Width of traffic lane.
Nw	-	Number of traffic lanes.
PC1	mm	Width of edge beam or pedestrian lane, min 500 mm.
PC2	mm	Width of edge beam or pedestrian lane, min 500 mm.
a	mm	Fillet weld throat depth.
b0	mm	Distance between rows of shear studs.

Table 4.7: Material fixed design parameters.

Parameter	Unit	Definition
SteelGrade	object	Selected among steels included in *MaterialClass*.
Shape	object	Web concept, 'Flat' or 'Corrugated'.
ConcreteGrade	object	Selected among grades included in *MaterialClass*.
PavementType	object	Selected among types included in *MaterialClass*.
hp	mm	Pavement cover thickness.
ReinforcementGrade	object	Selected among grades included in *MaterialClass*.
CementClass	object	Reinforcement cement class, 'S' or 'N'.
d_re	mm	Reinforcement diameter.
d_stud	mm	Shear stud diameter, 16 mm $\leq$ d_re $\leq$ 25 mm.
h_stud	mm	Shear stud height.

Table 4.8: Environmental fixed design parameters.

Parameter	Unit	Definition
RH	%	Relative humidity, recommened value 80 %.
T_max	C°	Maximum ambient temperature.
T_min	C°	Minimum ambient temperature.
vb	m/s	Reference wind velocity (TSFS 2018:57).
qp	kN/m^2	Characteristic velocity pressure (TSFS 2018:57).

Table 4.9: Construction fixed design parameters.

Parameter	Unit	Definition
Delta_T_cs	C°	Temperature difference between parts, recommend 15 C°.
T0	days	Initial bridge temperature, recommended 10 C°.
ts	days	Age of concrete at beginning of drying.
t0	days	Age of concrete at loading.
t0_cs	days	Age of concrete at loading, shrinkage.
SafteyClass	-	1, 2, 3, or 4.

Table 4.10: Fatigue fixed design parameters.

Parameter	Unit	Definition
ServiceLife	years	Usually 120 or 80 years.
Method	object	Fatigue method, 'SafeLife' or 'DamageTolerant'.
Consequence	object	Fatigue consequence, 'High' or 'Low'.
Nobs	-	Number of observed vehicles (SS-EN 1991-2, Table 4.5).
ADT	-	Average daily traffic (Trafikverket, 2022, Section 5.6)

Table 4.11: Variable design parameters.

	Unit	Definition	Domain
hw	mm	Web height.	800, 810, ... , 2190, 2200
tw	mm	Web thickness.	4, 6, 8, 10, 12, 14, 16, 18, 20, 25, 28, 30
bfo_i	mm	Upper flange width.	400, 450, ... , 1450, 1500
tfo_i	mm	Thickness.	16, 18, 20, 25, 28, 30, 35, 40, 45, 50, 55, 60
bfu_i	mm	Lower flange width.	400, 450, ... , 1450, 1500
tfu_i	mm	Thickness.	16, 18, 20, 25, 28, 30, 35, 40, 45, 50, 55, 60
a1	mm	Corrugation flat fold length.	50, 60, ... , 390, 400
a3	mm	Corrugation depth.	50, 60, ... , 390, 400
alpha	°	Corrugation angle.	30, 31, ... , 59, 60
Ccb	mm	C-C crossbeams.	2000, 2050, ... , 9950, 10000*
m_j	mm	Moving coordinates of segment change.	-L/4Nseg, ... , L/4Nseg (500 mm step)

Nseg = number of segments, i = 1, 2, ... , Nseg and j = 1, 2, ... , Nseg-1
**However, only distances that are evenly divisible with the span length is allowed.*

4.2 Genetic Algorithm Optimization

The used Genetic Algorithm is developed by Solgi (2020) and distributed as a Python library available without charge on The Python Package Index website (PyPI.org). The library can be used for standard and elite genetic algorithm optimizations. The method described in this section is based on the instructions given by Solgi (2020)

as well as the descriptions of meta-heuristic genetic algorithms given by Bozorg-Haddad, Solgi, and Loáiciga (2017).

4.2.1 Design Vector and Domain

The variables subject to the optimization are collected in the design vector $\mathbf{X} = (x_1, x_2, ..., x_N)$. For Genetic Algorithms, each solution of the design vector is called a chromosome, and each variable of the design vector is a gene (Bozorg-Haddad et al., 2017). For each gene, a design domain needs to be defined from where the algorithm can choose different values. The domain can be continuous, discrete, or binary (Bozorg-Haddad et al., 2017).

For a continuous domain, the algorithm can choose freely between a lower and an upper bound value. Further, the continuous domain can be defined as either a real or integer domain. The real domain allows all real values between the bounds to be chosen, while the integer domain only allows the integers in the domain to be chosen (Solgi, 2020). The discrete domain is a set of defined allowed values. For the algorithm used in this thesis, a discrete domain is not directly supported. However, discrete values may be collected in a vector, and then a continuous integer domain between 0 and the length of the vector can allow the algorithm to choose the discrete values. Lastly, the binary domain, also called a Boolean domain, allows the algorithm to choose 1 or 0 (Solgi, 2020), usually used to code true-or-false statements.

The length, N, of the specific design vector used is dependent on the number of longitudinal segments considered, N_{seg}. As seen in Equation 4.1, there are six genes that are not depending on N_{seg}. One gene is dependent on $(N_{seg} - 1)$ (the positions of the segment changes) and four genes are dependent on N_{seg} (thickness and width of the flanges). The domains are coded as discrete domains, meaning that the genes collected in each chromosome are integers giving the domain index for each design variable. The design variables and domains are previously presented in Table 4.11.

$$N = 6 + (N_{seg} - 1) + 4N_{seg} \tag{4.1}$$

4.2.2 Population

As described in Section 2.4.3, the Genetic Algorithm is a population-based meta-heuristic algorithm that searches for the most optimal solution among a set (a population) of solutions. The population can be seen as an $M \times N$ matrix, where M is the number of possible solutions (design row vectors) and N is the number of design variables (genes) subjected to the optimization (Bozorg-Haddad et al., 2017). The initial population is randomly generated and for discrete and binary domains each value has an equal possibility to be chosen. However, for continuous domains, the initial values are chosen by a distribution function, where a uniform distribution is most common (Bozorg-Haddad et al., 2017).

4.2.3 Parent Selection and Reproduction Methods

After each iteration, a new population (generation) is created. The first step of the algorithm to create a new generation is to select the parents. Usually, the user defines the proportion of the old generation that is kept as parents (Solgi, 2020). The algorithm will then decide the parent chromosomes either by ranking them based on their fitness value from the previous iteration or by randomly driven methods (Bozorg-Haddad et al., 2017). The parent chromosomes are kept as possible solutions in the next generation, and their genes are to create new solutions (children). These children are generated by the reproduction operations: crossover and mutation (Bozorg-Haddad et al., 2017).

The crossover operator combines the genes of two parent chromosomes to create two children. This is done using one of the three methods: one-point crossover, two-point crossover, or uniform crossover. One-point crossover is described by Equation 4.2 and 4.3 where index c is the crossover position in the vector (Bozorg-Haddad et al., 2017).

$$\begin{aligned} &\text{Parent 1:} \quad \mathbf{X} = (x_1, x_2, ..., x_N) \\ &\text{Parent 2:} \quad \mathbf{X}' = (x'_1, x'_2, ..., x'_N) \end{aligned} \tag{4.2}$$

$$\begin{aligned} &\text{Child 1:} \quad \mathbf{X}_1^{new} = (x_1, x_2, ..., x_c, x'_{c+1}, ..., x'_N) \\ &\text{Child 2:} \quad \mathbf{X}_2^{new} = (x'_1, x'_2, ..., x'_c, x_{c+1}, ..., x_N) \end{aligned} \tag{4.3}$$

The two-point crossover operation is done similarly with two crossover positions. The crossing positions are generated randomly between integer 1 and N. However, the uniform crossover can be explained by Equation 4.4, considering the same parents given in Equation 4.2 (Bozorg-Haddad et al., 2017). Here only the values at the crossover positions c and d are exchanged.

$$\begin{aligned} &\text{Child 1:} \quad \mathbf{X}_1^{new} = (x_1, x_2, ..., x'_c, x_{c+1}, ..., x'_d, x_{d+1}..., x_N) \\ &\text{Child 2:} \quad \mathbf{X}_2^{new} = (x'_1, x'_2, ..., x_c, x'_{c+1}, ..., x_d, x'_{d+1}..., x'_N) \end{aligned} \tag{4.4}$$

The second reproduction operator, mutation, randomly replaces a gene of the child chromosome and is according to Bozorg-Haddad et al. (2017) an important aspect of the Genetic Algorithm as it helps the algorithm to avoid local optima. For the algorithm by Solgi (2020), the elite selection is also used to create a new generation. The user define the proportion of elite chromosomes, defined as the solutions with the best fitness values. Solgi (2020) explains that a small proportion of elite chromosomes can increase the convergence rate while a larger proportion can result in the algorithm finding a local optima.

4.2.4 Constraints and Penalty Functions

The constraints for the optimization problem considered in this master's thesis are the capacity demands on the design, described in Section 3.6, as well as other functional and computational constraints. The constraints can be applied in the algorithm by removal of non-feasible solutions, refinement of the solution, or by the use of penalty functions. Bozorg-Haddad et al. (2017) describe the removal of solutions

as problematic since the method does not consider how near the solution is to being feasible, which can result in potential good genes being disregarded. Furthermore, Bozorg-Haddad et al. (2017) highlight the difficulty of defining the refinement process and therefore recommend the penalty method.

In the penalty method, a magnitude is added to or subtracted from the objective function if the constraints are not fulfilled (Bozorg-Haddad et al., 2017). A simple way is to set the magnitude of the penalty to something larger than the largest value that the objective function could result in, within the design domain (Solgi, 2020). This is an efficient approach to removing non-feasible solutions. However, this method fails to teach the algorithm which genes result in chromosomes close to the optimal. Instead, to positively impact the convergence rate, Solgi (2020) suggests using the method described by Bozorg-Haddad et al. (2017). This method is implemented for the specific problem of this thesis, as shown in Equation 4.5 to 4.8.

$$penalty = \sum g_i(\eta_i(\mathbf{x}))\phi_i + h_j(\xi_j(\mathbf{x}))C \tag{4.5}$$

$$g_i(\eta_i(\mathbf{x})) = \begin{cases} 0 & \text{if} \quad \eta_i \leq 1 \\ 1 & \text{otherwise} \end{cases} \quad i = 1, 2, ..., J \tag{4.6}$$

$$h_i(\xi_i(\mathbf{x})) = \begin{cases} 0 & \text{if fulfilled} \\ 1 & \text{otherwise} \end{cases} \quad j = 1, 2, ..., K \tag{4.7}$$

$$\phi_i = \alpha(\eta_i - 1)^\beta + C \tag{4.8}$$

where J is the number of constraints regarding the utilization ratios and η_i, which are penalized using the penalty functions g_i and ϕ_i. The weight constants α, β, and C help to map the path for the Genetic Algorithm. Bozorg-Haddad et al. (2017) recommend that the constraints are calibrated by performing a sensibility study, see Section 4.3.1. For K number of other constraints, ξ_j, only the weight constant C is used to adjust the magnitude, as here it is less relevant how close to fulfilling the constraint the solution is. Also, this helps the algorithm to understand that the utilization constraints are prioritized before the functional constraints.

4.2.5 Objective and Fitness Functions

To evaluate the fittest chromosome in the design domain, each solution is evaluated by a fitness function, $F(\mathbf{x})$. For a minimization optimization problem, the fitness function is defined as Equation 4.9 (Bozorg-Haddad et al., 2017) dependent on the objective function $f(\mathbf{x})$ and the penalty contribution as defined in Section 4.2.4. For a maximization optimization problem, the penalty contribution would instead be subtracted from the objective function to decrease the value of the fitness function for all non-feasible solutions.

$$F(\mathbf{x}) = f(\mathbf{x}) + penalty \tag{4.9}$$

The objective functions considered in this thesis are the total cost or only the investment cost for the study related to the Life Cycle Cost analysis, and the sum of CO_2 equivalents for the study related to the Life Cycle Assessment. All functions are aimed to be minimized. To find an optimal solution, one method is to perform a multi-objective optimization including both objectives. However, as explained in Section 2.4.1, multi-objective optimizations with the potential contradicting objectives require post-processing such as Multi-Criteria Decision Making methods to reach one single optimal solution.

A second method is to incorporate the Multi-Criteria Decision Making method already in the optimization routine by adding a weight parameter to one of the objectives. This could be done by multiplying CO_2 emissions with a cost of K SEK per CO_2 equivalent and performing a single-objective optimization aiming to minimize the cost. One disadvantage of this method is that the magnitude of K could be difficult to choose since its influence on the total cost would depend on how large it is in relation to the other cost parameters, such as material and maintenance, which will differ for each solution.

A third method, which is used in this thesis, is to perform two separate optimizations, one per objective. The disadvantage is that no general optimal solution is found by the algorithm. However, the results will give a good basis to evaluate and discuss the impact of the objectives on the design and identify the effects of the different design choices in terms of material and web shape. Additionally, the program can also perform an optimization aimed to minimize the total steel weight. This objective function is used in the sensibility study of the program presented in Section 4.3.

4.2.6 Algorithm Parameters

In the genetic algorithm developed by Solgi (2020), some parameters should be defined by the user. The definitions and default values of the parameters are as follows:

- **Maximum number of iterations**: breaking criteria of the algorithm, chosen as an integer number.
- **Initial population size**: an integer number and Solgi (2020) suggests an initial population size of 100. However, a sensibility study should be performed for the specific problem and it should be set as discussed in Section 4.2.2 and 2.4.3.
- **Mutation probability**: a real number between 0 and 1, with a default value of 0.01 (1 %). Solgi (2020) claims this to be the parameter to study deepest as it heavily affects the result.
- **Elite ratio**: a real number between 0 and the Parent portion value, as it relates to the ranking of parents, with a default value of 0.01 (1 %). The

setting should be considered as discussed in Section 4.2.3 and different values of the parameter are evaluated during the sensibility study in section 4.3.5.

- **Crossover probability**: defines the probability of a solution becoming a parent and should be a real number between 0 and 1, with a default value of 0.5 (50 %).
- **Parent portion**: defines the portion of old solutions in the new generation and should be a real number between 0 and 1. The default value is 0.3 (30%).
- **Crossover type**: defined as one-point, two-point, or uniform, as explained in Section 4.2.3.
- **Maximum number of iterations without improvement**: used in cases where the algorithm is suspected to find the optima before the maximum number of iterations is reached. However, should be used with caution to avoid local optima.
- **Function timeout**: stopping criteria depending on the time of each iteration and the idea is to avoid infinity loops. Therefore, it is not a parameter that needs to be calibrated and the default value of 10 seconds is usually sufficient (Solgi, 2020). However, for this optimization problem with a large number of functions and constraints, the parameter might be increased.

Most parameters have a default value, however, Solgi (2020), as well as Bozorg-Haddad et al. (2017), highlight the importance of carefully selecting the parameters to achieve a true optimum. If the user does not hold any previous experience with the specific optimization problem, both recommend performing a sensibility analysis to calibrate the algorithm parameters.

4.3 Sensibility Study

A sensibility study is performed to understand the influence of the different algorithm parameters and to calibrate them to fastest reach an optimal solution. The parameters to study are the penalty function constants, the crossover type, the population size, the number of iterations, the elite ratio, and the mutation probability. The study aims to achieve a feasible solution, meaning that all constraints are fulfilled, and to reach the lowest objective. The study is conducted in the same order as the sections, with the concluded best value for each kept in the following study. The method is used to limit the study while still including the cross-impact of the different settings. The objective is the total weight of the welded plate sections and the hot-rolled crossbeams. The sensibility study is performed on the bridge described in the case study, see Section 5.1 using steel grade S355 and girders with a flat web.

The results of each sub-study are presented in the following sub-sections apart from the crossover type study and the mutation probability study. The study on crossover

type concluded with a similar result for all three types, and therefore no conclusion could be drawn. The type 'uniform' was further used as it is the default setting of the algorithm. Likewise, the pre-study on the effect of changing the mutation probability did not exhibit any important differences. Therefore, the default value of 0.2 is used.

4.3.1 Penalty Constants

The impact of the penalty constants described in Section 4.2.4 is studied first as it has a high influence on the feasibility. The hypothesis is that with a too low penalty contribution, the algorithm will discard the constraints and only aim for low weight, while for a too high penalty contribution, the algorithm will achieve feasibility but not minimize the weight. Therefore, the study aims to find the middle value between the two extremes. The constant b is not part of the study as $(\eta_i - 1)^b$ for all $\eta_i > 1$ can result in both positive and negative values. To avoid this, b_{pen} is set equal to 1. The constants a_{pen} and C_{pen} are set to an equal value throughout the study to assure that the algorithm prioritizes the utilization constraints over the functional constraints. The constants were tested for values between 10^3 to 10^{11}, five times each. For each run, the objective value, as well as the number of constraints that were not fulfilled, was annotated and presented in Table 4.13. The algorithm parameter settings are summarized in Table 4.12.

Table 4.12: Genetic algorithm parameters used in the sensibility study for penalty constants.

Parameter	**Value**
Iterations	100
Population	100
Mutation probability	0.2
Elite ratio	0.01
Crossover portion	0.3
Crossover type	'uniform'
Max iteration without improv	None

Table 4.13: Results from the sensibility study of the penalty constants 1e3 to 1e11.

Penalty ↓	**Run →**	1	2	3	4	5	Average
1e3	Objective Function [ton]:	102	97	95	92	100	97
	UR Constrain not OK:	1	2	2	3	1	
	Other Constrains not OK:	3	6	0	5	2	
1e4	Objective Function [ton]:	103	104	107	106	104	105
	UR Constrain not OK:	1	0	1	1	0	
	Other Constrains not OK:	4	3	1	3	4	
1e5	Objective Function [ton]:	108	103	102	102	106	104
	UR Constrain not OK:	0	1	1	0	1	
	Other Constrains not OK:	3	3	2	1	2	
1e6	Objective Function [ton]:	106	109	102	110	103	106
	UR Constrain not OK:	1	1	0	1	0	
	Other Constrains not OK:	3	2	4	3	2	
1e7	Objective Function [ton]:	114	105	109	106	111	109
	UR Constrain not OK:	0	0	0	0	0	
	Other Constrains not OK:	1	2	2	2	2	
1e8	Objective Function [ton]:	104	113	111	106	99	107
	UR Constrain not OK:	0	0	0	0	0	
	Other Constrains not OK:	2	2	2	2	5	
1e9	Objective Function [ton]:	102	103	103	108	107	105
	UR Constrain not OK:	0	0	0	0	0	
	Other Constrains not OK:	3	3	1	3	1	
1e10	Objective Function [ton]:	110	104	106	108	111	108
	UR Constrain not OK:	0	1	0	1	0	
	Other Constrains not OK:	3	4	4	1	3	
1e11	Objective Function [ton]:	110	114	111	108	111	111
	UR Constrain not OK:	0	0	0	0	1	
	Other Constrains not OK:	2	2	3	4	0	

From the displayed results in Table 4.13, the utilization ratio constraints were not always fulfilled for penalty constants of 1e3 to 1e6. Since the utilization criteria is important because it contains the design capacity, the parameter values are considered non-suitable. The two last values, 1e10 and 1e11, also contain a large number of utilization ratios that are not fulfilled. Together with 1e7 they have a higher value of the average objective function and are therefore considered non-suitable.

Additionally, there is not much possible improvement when increasing the parameters this high. To verify and distinguish between 1e8 and 1e9, five more iterations are made and the result is summarized in Table 4.14. This further study shows similar results for both parameters 1e8 and 1e9 where both of them have one run that does not fulfill the utilization constraints. However, when looking at the utilization ratios that are not fulfilled, the value is 101 % for 1e9 and 112 % for 1e8. Therefore, the chosen penalty constant is 1e9.

Table 4.14: Results from the sensibility study of the penalty constants 1e8 and 1e9.

Penalty ↓	**Run →**	1	2	3	4	5	Average
1e8	Objective Function [ton]:	108	111	103	103	104	106
	UR Constrain not OK:	0	0	1	0	0	
	Other Constrains not OK:	6	1	3	2	2	
1e9	Objective Function [ton]:	110	102	107	101	105	105
	UR Constrain not OK:	0	0	0	0	1	
	Other Constrains not OK:	4	5	4	5	1	

4.3.2 Population Size

The sensibility study for the population size is done using the input data summarized in Table 4.15. The considered population sizes are: 50, 100, 150, 200, 250.

Table 4.15: Genetic algorithm parameters used in the sensibility study for population size.

Parameter	**Value**
a_{pen}	10^9
b_{pen}	1
C_{pen}	a_{pen}
Iterations	100
Mutation probability	0.2
Elite ratio	0.01
Crossover portion	0.3
Crossover type	'uniform'
Max iteration without improv	None

As seen by the results presented in Table 4.16, the objective function and the non-fulfilled constraints vary for different values of the population size as well as the different runs. Population size 200 could be considered the best option, as it shows consistency between the runs and has the least unfulfilled constraints. However, due to vague results and the size 150 giving a lower weight and a lower computational time, a population size of 150 is used in the following studies. The hypothesis is to solve the problem with unfulfilled (other) constraints by increasing the number of iterations.

Table 4.16: Results from the sensibility study of the population size.

Population ↓	**Run →**	1	2	3	4	5	Average
50	Objective Function [ton]:	117	104	101	104	113	108
	UR Constrain not OK:	1	1	1	0	0	
	Other Constrains not OK:	1	1	2	1	1	
100	Objective Function [ton]:	104	106	110	113	120	111
	UR Constrain not OK:	0	1	1	0	0	
	Other Constrains not OK:	2	4	1	1	1	
150	Objective Function [ton]:	101	110	102	105	105	105
	UR Constrain not OK :	1	0	0	0	1	
	Other Constrains not OK:	2	3	3	2	3	
200	Objective Function [ton]:	112	112	110	119	112	113
	UR Constrain not OK:	0	0	0	0	1	
	Other Constrains not OK:	2	2	2	0	0	
250	Objective Function [ton]:	107	107	103	104	109	106
	UR Constrain not OK:	1	1	1	0	0	
	Other Constrains not OK:	2	3	3	3	2	

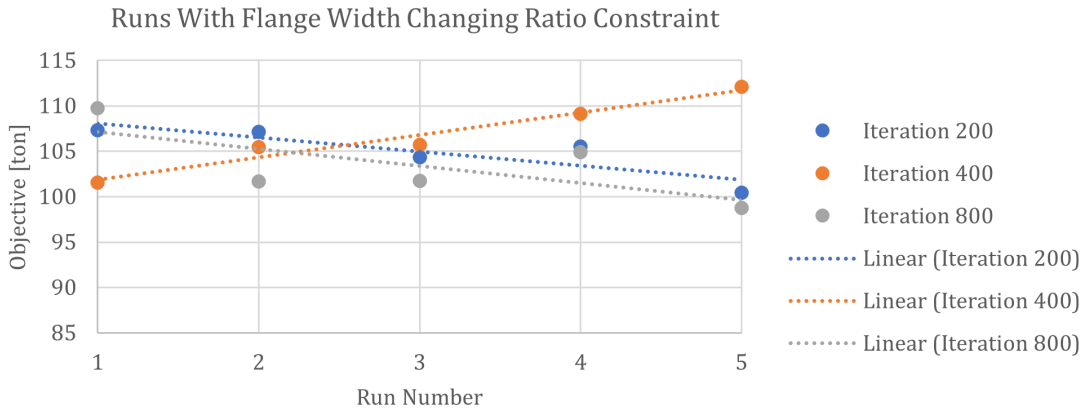
4.3.3 Number of Iterations

The previous study to determine the population size was performed with 100 iterations. Since the result is not satisfactory, this study is looking at 200, 400, and 800 iterations. The algorithm settings are summarized in Table 4.17. The study aims to find the number of iterations that fulfill all constraints and result in a low objective function with consistency over five runs.

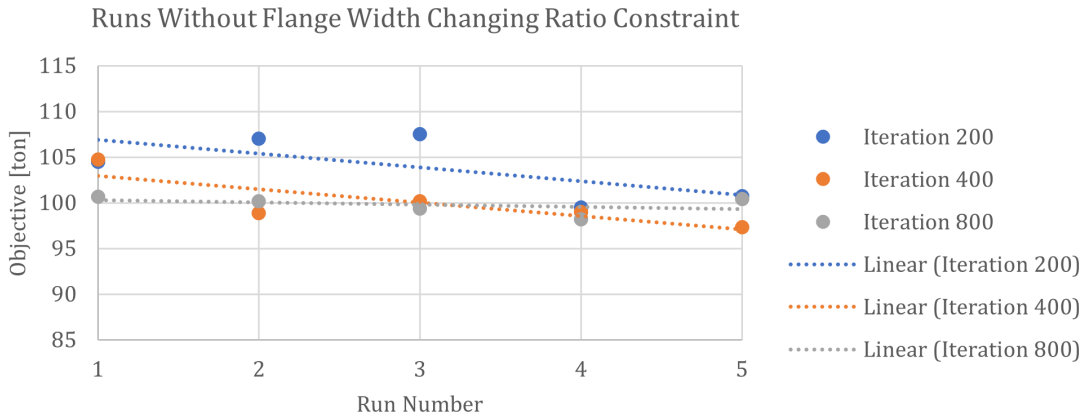
Table 4.17: Genetic algorithm parameters used in the sensibility study for number of iterations.

Parameter	**Value**
a_{pen}	10^9
b_{pen}	1
C_{pen}	a_{pen}
Population	150
Mutation probability	0.2
Elite ratio	0.01
Crossover portion	0.3
Crossover type	'uniform'
Max iteration without improv	None

The result is plotted in Figure 4.2a with the objective function on the y-axis in *tons* and the number of runs on the x-axis. The graph shows non-consistent weight even for 800 iterations and all the runs had at least one functional (other) constraint not fulfilled. The unfulfilled constraints were all related to the allowed change ratio for the width of the flanges between the segments. As a run of 800 iterations takes about half an hour, a further increase in the number of iterations was not considered sufficient for the aim of this program. Instead, the same procedure was done without considering the change ratio constraints. The result is plotted in Figure 4.2b. Here, 800 iterations give better consistency over the runs and lower average weight. However, as the solution was not optimal from a construction point of view, a reduction of the design variables was instead considered and studied, presented in Section 4.3.4.



(a) Program including the change ratio constraint between the segment flanges.



(b) Program without the change ratio constraint between the segment flanges.

Figure 4.2: Plot of the objective function over five runs for iteration number 200, 400 and 800 including linear trends. In Figure (a) all constrain are considered while in Figure (b) the change ratio constraint between segments are removed.

4.3.4 Reduction of Design Variables

To decrease the running time, the variety between the segments was reduced to only allow thickness change of the flanges (tfo_i and tfu_i with $i = 1, 2, ..., Nseg$ with N_{seg} being the number of segments), letting the flange widths be constant along the bridge ($\text{bfo}_i = \text{bfo}\ \forall i$ and $\text{bfu}_i = \text{bfu}\ \forall i$). As seen in the results plotted in Figure 4.3, not only is there a consistency between the runs regardless of iterations tested but also the objective is reduced for all runs compared to the results shown in Section 4.3.3. Therefore, the decision was made to reduce the design variables by only allowing thickness change between the segments, both for the following sensibility study and the case study.

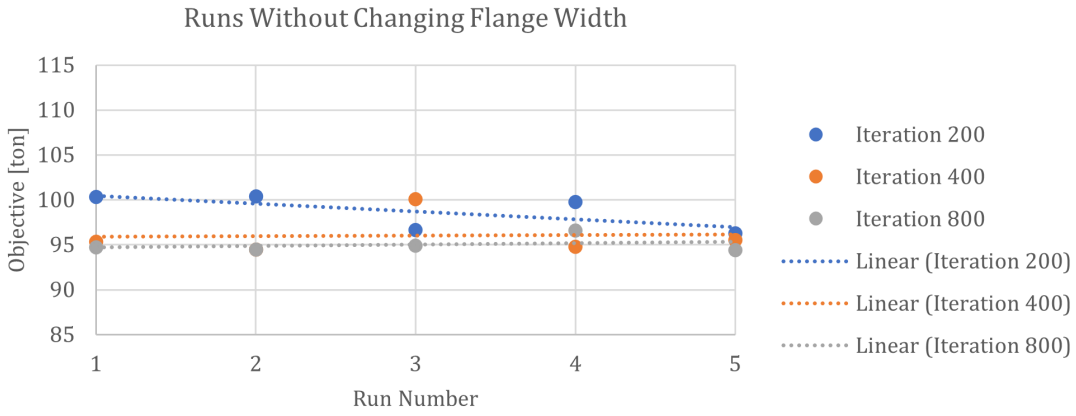


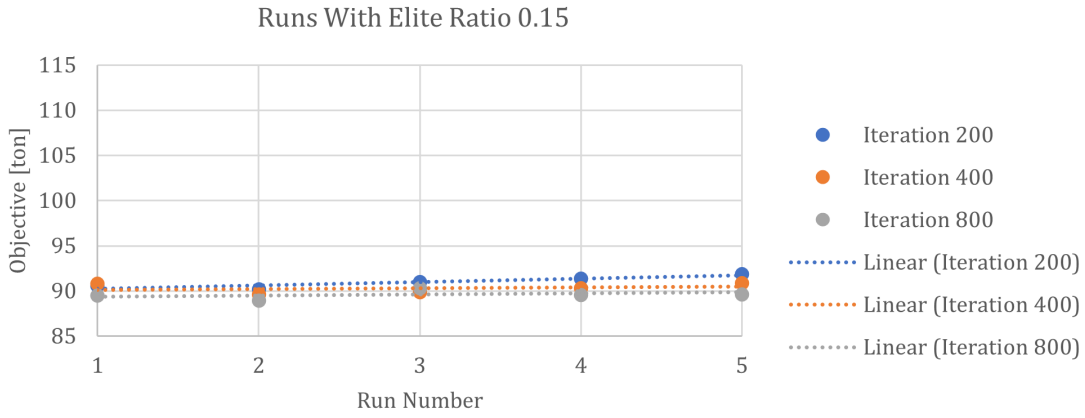
Figure 4.3: Plot of the objective function over five runs for iteration number 200, 400 and 800 including linear trends. In the program, the change of flange width between segments are disregarded leading to a reduced number of design variables.

4.3.5 Elite Ratio

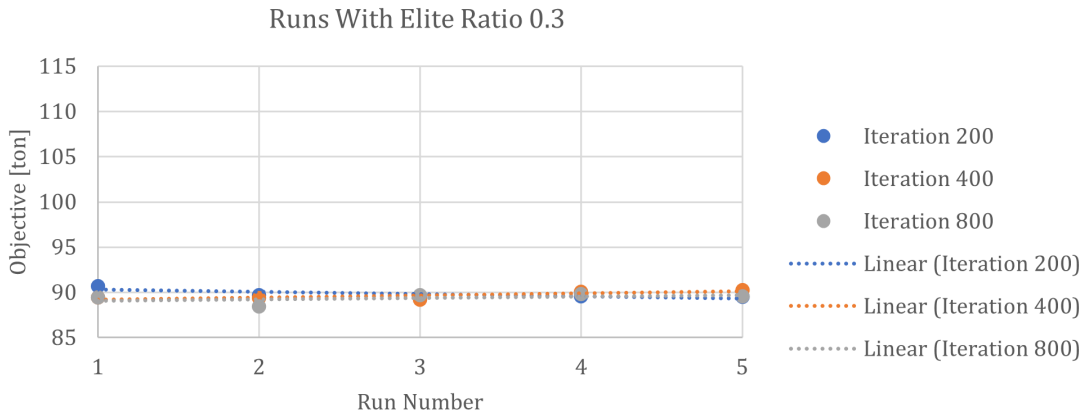
To minimize the required number of iterations and thereby decrease the running time, the elite ratio parameter was adjusted using three values: 0.01, 0.15, and 0.30 and each value was run five times for the iteration values 200, 400, and 800. The elite ratio can not be larger than the parameter *parents portion* that by default is set to 0.3. The results from the study for values of 0.15 and 0.3 is displayed in Figure 4.4 with the objective function on the y-axis and the number of runs on the x-axis. The third value, 0.01, is the default setting and was used during the earlier studies displayed in Figure 4.3. From studying the figures, it is concluded that using a higher elite value leads to a lower objective.

Additionally, the minimized objective was consistent between the runs, and also similar for all settings of iterations, showing that convergence is found. The developer claims that generally, a high setting of the elite ratio can cause the algorithm to find a local optima instead of a global one. However, the study shows little risk of this happening for this specific problem, as already for 50 iterations a high setting of the elite ratio resulted in an objective close to 800 iterations for a low setting.

The reason for this can be that for this optimization, it exists several alternatives that can lead to a minimum weight.



(a) The elite parameter equal to 0.15.



(b) The elite parameter equal to 0.3.

Figure 4.4: Plot of the objective function over five runs for iteration number 200, 400 and 800 including linear trends. In Figure (a) an elite ratio of 0.15 is used, and in Figure (b) the elite ratio equals to 0.3.

When studying the convergence plots in detail it was found that there was some runs where the objective decreased after 400 iterations, see Figure 4.5. Here, the fitness function is plotted on the y-axis and the number of iterations on the x-axis.

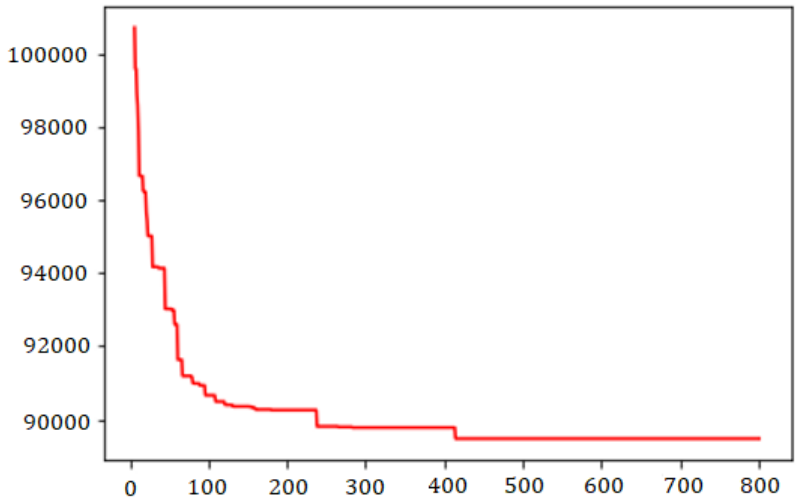
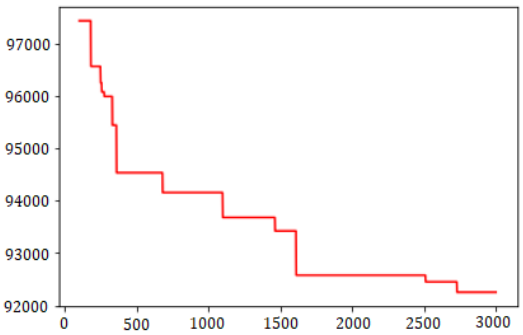
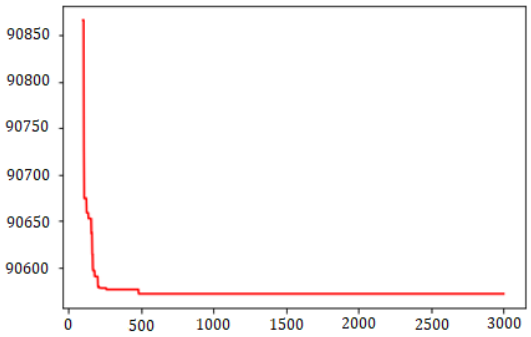


Figure 4.5: A plot from one of the runs of the optimization routine with elite ratio 0.3, population size 150 and 800 iterations. The value of the fitness function is stated on the y-axis, while the iteration numbers are plotted on the x-axis. To be able to visually see the fitness decrease, the first 5 iterations resulting in unfeasible solutions, with resulting fitness of magnitude 10^9 and above, is not shown in the graph.

Therefore, a short study with 3000 iteration where performed. The results is seen in Figure 4.6. When using a elite ratio of 0.01 it is concluded that it would require more iterations than 3000 to converge. However, when using the value 0.3 for the elite parameter, the results showed that it should be enough to use 500 iterations. To account for potential minimum found after 500 iterations, the program is run 3 times, and then the minimum of these runs is chosen as the optimal solution.



(a) The elite parameter equal to 0.01.



(b) The elite parameter equal to 0.3.

Figure 4.6: Plot of the fitness function as function of the number of iterations. In Figure (a) an elite ratio of 0.01 is used, and in Figure (b) the elite ratio equals to 0.30. To be able to visually see the fitness decrease the first 100 iterations is not shown.

4.3.6 Design Impact due to Life Cycle Cost Data

After calibrating the algorithm parameters, the link to the life cycle cost tool developed by Nissan and Woldeyohannes (2022) was investigated. As no additional design variables or constraints are introduced, the calibrated parameters are expected to work in the same way, apart from the penalty constants. As the penalty function needs to be related to the objective function, a_{pen} was increased to 10^{12}, relating to an expected cost of around 10 million SEK. Additionally, only short runs of 50 iterations with a smaller population size of 50 were used, as the study aimed to see how the program reacted to changing data rather than finding a true minimum. The elite ratio was kept at 0.3 to increase the convergence rate.

The two studies conducted related to the life cycle cost input data is about the cost of welding and crossbeams. The first investigates how the program chooses the design when the production cost of welding is increased 10^3, 10^6 and 10^9 times. The expectation is that for a higher welding cost, the program reduces the amount of longitudinal segments, while for a lower welding cost, the program prioritizes to reduce the steel material by changing the flange dimensions with increasing moment. The results show that with the real welding price, there were several sections and the thickness of the flanges varied for each segment. When the price increased by multiplying with the given numbers, the program chose less segments.

The second study looked at how the program chose the C-C distance of the crossbeams when the material cost of hot-rolled sections and the assembly cost of crossbeams were reduced. The expectation is that the program would add more crossbeams when the cost is reduced to be able to utilize thinner plates, as the crossbeam distance is the buckling length of the upper flange in the construction phase and their positions is also the positions of the transverse stiffeners affecting the shear buckling. The result showed that when reducing the crossbeam cost by half, also the C-C distance halved, resulting in about double the amount of intermediate crossbeams: from C-C 8.5 m (5 cross beams) to C-C 4.25 m (11 crossbeams). However, when removing the cost completely, the program still chose a distance of 4.25 m. This could be explained as for C-C distances smaller than this, the buckling factors are not dimensioning.

5
Case Study

A case study is conducted to test the design optimization program developed as described in Chapter 4. In order to use relevant fixed input data for the case study, the requirements and outlining geometry of a bridge that is already designed and built is considered. The girder design is optimized for the concept of girders with flat webs in steel grade S355 as well as for duplex stainless steel girders with corrugated webs. The concepts are optimized with the target to minimize the life cycle cost and the optimization results are monitored with different settings to determine how the costs relates to different requirements. Additionally, the optimization is run with the targets to minimize the weight as well as the life cycle assessment to understand how these can affect the design.

5.1 Bridge Over Delångersån in Näsviken

The chosen bridge is a bridge over Delångersån, located in Näsviken, Sweden. The designation by Trafikverket is 100-262-1 and it is a one span, simply supported composite bridge of 51 meters, see Figure 5.1. The bridge consists of two parallel I-girders, as seen in Figure 5.2, with a concrete deck on top. The width of the bridge is 10 meters with two lanes of each 3.25 meters, a pedestrian lane of three meters on one side, and 0.5 meters for installations on the other side, as seen in Figure 5.3.

Figure 5.1: Bridge 100-262-1 over Delångersån in Näsviken. Picture taken from BaTMan.

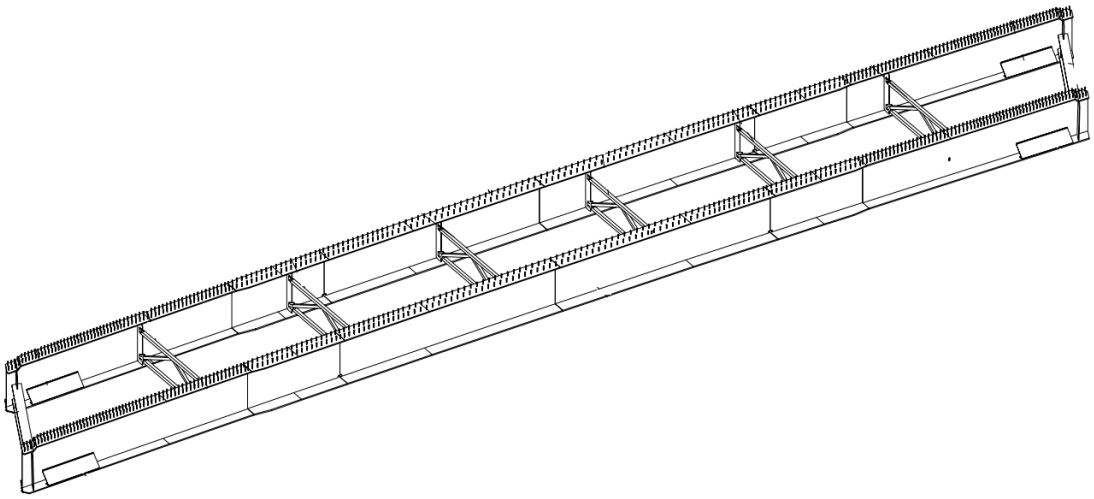


Figure 5.2: Steel parts of Bridge 100-262-1 over Delångersån in Näsviken.

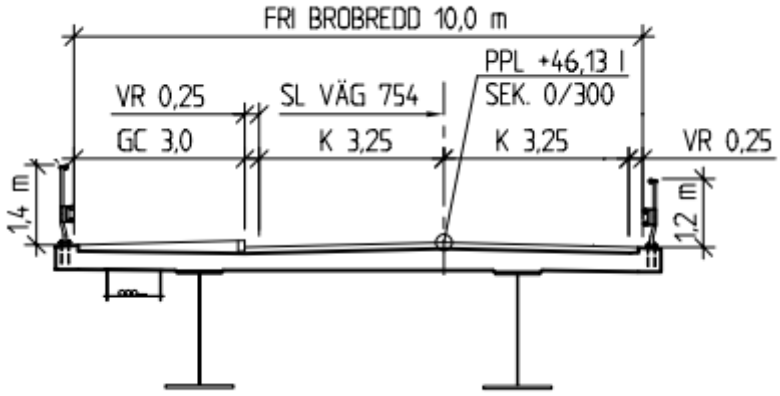


Figure 5.3: Cross-section of bridge 100-262-1 over Delångersån in Näsviken. Picture taken from BaTMan.

When running the parametric program and the Life Cycle Cost tool for the original design, the resulting weight was concluded to 117 tons and and the total life cycle cost 6.3 million SEK. Note that the cost considers the prices and amount of traffic assumed for the optimization and may not reflect the actual cost of when the bridge was originally built.

5.1.1 Input Data for the Optimization Program

The input parameters are chosen according to Tables B.1 to B.5 found in Appendix B and the algorithm settings are presented in Table B.6, also in Appendix B. The height of the concrete deck is taken as the height of the existing bridge deck. However, since the height of the concrete deck varies, as seen in Figure 5.3, the concrete deck height is set to the average height 320 mm, in the same way as in the previous master's thesis by Henrysson and Yman (2020) who studied the same bridge.

5.2 Life Cycle Cost Optimization

The optimization for life cycle cost is performed for duplex stainless steel girders with corrugated web and girders of carbon steel grade S355 with flat web. The optimization study is conducted in several parts. In the first part, the web height of the main girders is run in a domain between 1-2 meters and then in a 1-3 meters domain to study the cost related to the web height. In the second analysis, the number of observed vehicles, N_{obs}, and average daily traffic, ADT, is increased from 0.05 millions to 0.5 millions and from 5 thousands to 50 thousands to see how the usage of the bridge affects the total cost. In the third analysis, the material cost is divided by two to study how the material price affects the total cost. The used domain settings are stated in Table 4.11. The table does however differ considering the web height domain, where the case study domains are explained in the following section.

The Life Cycle Cost in this study is taken as the sum of the investment cost and the user cost with the resell profit deducted. The investment cost includes the material and production cost, where the production cost includes everything regarding the production of the bridge and the material cost includes everything regarding the material, such as the cost for welded plate sections (main girder and end crossbeams) and the hot-rolled sections (intermediate crossbeams) as well as the concrete material, the reinforcement bars, the shear studs, and the weld filler material. The user cost includes the cost for maintenance work and material, as well as the cost due to the consequence of closing the bridge, such as redirecting traffic. Finally, the resell profit means that you recycle the material by selling it.

5.2.1 Increased Web Height

The maximum allowed height of the girders is usually determined by case-specific requirements. Therefore, as the existing bridge over Delångersån has a web height of 1.96 meters, the domain for this case study is set to 1 to 2 meters. However, previous master's theses on the subject of composite bridges using stainless steel girders with corrugated webs have shown that increasing web height can reduce the material use (Henrysson and Yman, 2020) (Steffner and Öman, 2021). Therefore, an additional study with the web height domain 1 to 3 meters is performed.

The optimal solutions for the domain 1 to 2 meters gave web heights close to the upper limit: 1.89 meters for both alternatives. For the 1 to 3 meters domain however, the program choose a web height of only 2.47 meters for the carbon steel alternative while for the stainless steel alternative a web height of 2.85 meters was chosen. The full dimension of each solution is found in Appendix D.

The life cycle cost results from the study, listed in Table 5.1, show that for girders with webs limited to 2 meters, the carbon steel alternative is cheaper both in terms of investment and life cycle cost. The same is true regarding the investment cost for the webs limited to 3 meters, however, savings can be made in terms of the total

life cycle cost by using the stainless steel alternative.

Table 5.1: The resulting cost [SEK] of the run giving the smallest value of the life cycle cost for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
h_w domain:	1-2m	1-3m	1-2m	1-3m
Material	2 250 000	2 330 000	5 380 000	4 680 000
Production	1 920 000	1 990 000	1 130 000	1 060 000
Tot investment:	4 170 000	4 320 000	6 510 000	5 740 000
Maintenance:				
- Cost	1 700 000	1 910 000	0	0
- Consequence	50 000	50 000	0	0
Tot user cost:	1 750 000	1 960 000	0	0
Resell	-20 000	-20 000	-40 000	-40 000
Tot LCC:	**5 900 000**	**6 260 000**	**6 470 000**	**5 710 000**

For web height limited to 2 meters, material could be saved by using stainless steel, and, as seen in Table 5.2, even more material savings could be achieved if a web height of up to 3 meters is allowed for the stainless steel alternative. The material weight increases when increasing the web height domain for the carbon steel alternative. One reason for this is concluded to be that when increasing the web height, also the total painting area increases, which is the largest post in the maintenance budget. Therefore, when optimizing to minimize the total cost, the algorithm does not prioritize low weight in the same way as it does for the stainless steel alternative where the material cost is governing. Additionally, when the web height is increased the flat web thickness is also increased from 20 to 25 millimeters, while the web thickness of the corrugated alternative reduces with increased height from 10 to 8 millimeters. The reason for this is concluded to be that the program aims to to avoid effective height reduction due to risk of local buckling of the flat web.

Table 5.2: The resulting steel weight [kg] of the run giving the smallest value of the life cycle cost for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
h_w domain:	1-2m	1-3m	1-2m	1-3m
Weight	91 000	94 400	82 100	70 200

The result of the Life Cycle Analysis is summarized in Table 5.3. The total emissions increase with increased steel weight and are higher per kg for the stainless steel alternative compared to the carbon steel alternative. Therefore, even though the weight of the stainless steel alternative with a height domain of 1 to 2 meters is lower, the emissions are higher than for the corresponding carbon steel alternative. For the height domain 1 to 3 meters, where the stainless steel alternative is 25 % lower

in mass, the total emission is lower than the corresponding carbon steel alternative. Further correlations between weight, life cycle cost, and life cycle analysis for the different alternatives can be seen in the summarizing bar chart in Figure 5.4.

Table 5.3: The resulting Life Cycle Assessment [CO_2 eq.] of the run giving the smallest value of the life cycle cost for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
h_w domain:	1-2m	1-3m	1-2m	1-3m
LCA	231 000	237 000	240 000	219 000

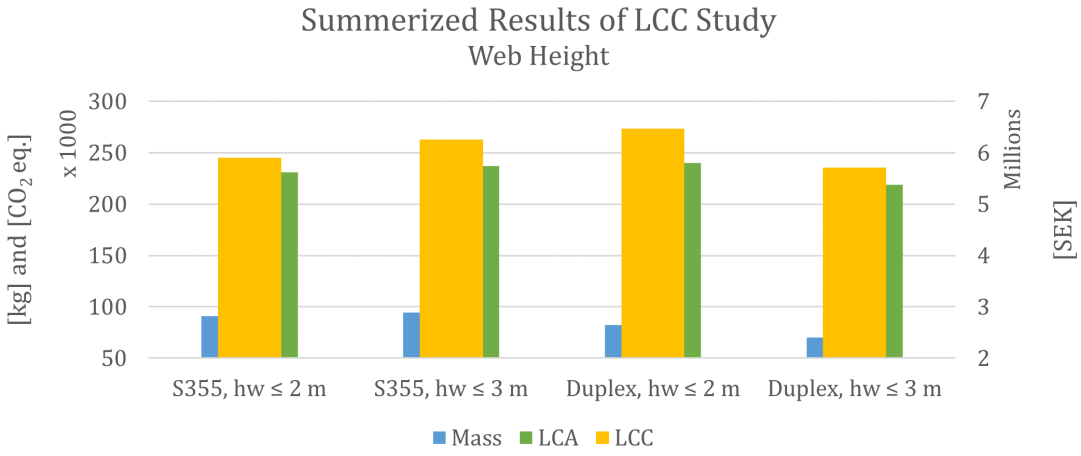


Figure 5.4: LCC Web Height Study: The resulting Life Cycle Cost [SEK], mass [kg] and CO_2 equivalent of the run giving the smallest life cycle cost for each optimization alternative.

5.2.2 Increased Average Daily Traffic

Depending on where the bridge is located, it will be loaded by different numbers of vehicles. This is regulated by using the parameters *Number of Observed Vehicles*, N_{obs} as well as the *Average Daily Traffic*, ADT. To compare the difference in total life cycle cost between a high used road and a less-loaded road, the variables N_{obs} and ADT is increased from $N_{obs} = 0.05 \cdot 10^6$ to $N_{obs} = 0.5 \cdot 10^6$ and from $ADT = 5 \cdot 10^3$ to $ADT = 50 \cdot 10^3$. The values for the number of observed vehicles are recommended values found in Eurocode 1991-2, Table 4.5 (Swedish Standards Institute, 2002d). The lower value represents a bridge with a smaller road while the higher value represents a bridge with a larger road. The average daily traffic is taken from *Vägars och gators utformning* (Trafikverket, 2022, Section 5.6). Note that, the increased values are not taken as the maximum values given in the documents as it would not be realistic on a the bridge with only one lane in each direction, as the one considered in this study.

In the Life Cycle Cost analysis, the average daily traffic affects the calculations of the maintenance consequence cost. With larger roads and higher amounts of traffic, the cost of closing roads for maintenance increases. In the design calculations, the number of observed vehicles affects the calculations of fatigue described in Section 3.6.8. The cost results from the study are summarised in Table 5.4.

When comparing the different alternatives, one can see that by increasing the parameters, the total life cycle cost for the carbon steel alternative increases as the maintenance consequence cost increases with increasing ADT. This is also seen in Table 5.5, showing that the weight of the S355 alternative stays approximately the same when increasing the parameters, as the increased cost is due to maintenance costs and not material costs.

Table 5.4: The resulting cost [SEK] of the run giving the smallest value of the life cycle cost for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
N_{obs}:	$0.05 \cdot 10^6$	$0.5 \cdot 10^6$	$0.05 \cdot 10^6$	$0.5 \cdot 10^6$
ADT:	$5 \cdot 10^3$	$50 \cdot 10^3$	$5 \cdot 10^3$	$50 \cdot 10^3$
Material	2 250 000	2 270 000	5 380 000	5 350 000
Production	1 920 000	1 880 000	1 130 000	1 130 000
Tot investment:	4 170 000	4 150 000	6 510 000	6 480 000
Maintenance:				
- Cost	1 700 000	1 660 000	0	0
- Consequence	50 000	510 000	0	0
Tot user cost:	1 750 000	2 170 000	0	0
Resell	-20 000	-20 000	-40 000	-30 000
Tot LCC:	**5 900 000**	**6 300 000**	**6 470 000**	**6 450 000**

Table 5.5: The resulting steel weight [kg] of the run giving the smallest value of the life cycle cost for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
N_{obs}:	$0.05 \cdot 10^6$	$0.5 \cdot 10^6$	$0.05 \cdot 10^6$	$0.5 \cdot 10^6$
ADT:	$5 \cdot 10^3$	$50 \cdot 10^3$	$5 \cdot 10^3$	$50 \cdot 10^3$
Weight	91 000	92 000	82 100	81 700

For the stainless steel alternative, the increased N_{obs} lead to fatigue governing the design with up to 98 % FAT utilization rates for Detail C (web principle stress), compared to the maximum 64 % FAT utilization rate for the alternative with the lower N_{obs} (see all utilization rates in Appendix D.6 and D.3). However, the same web thickness of 10 mm and web height of 1890 mm was chosen for both girders, resulting in no major design change. The reduction in weight and cost is concluded

to be due to the randomness of the algorithm rather than the parameters change. The result suggest however that if the N_{obs} would be taken as the highest value, used for multi-lane roads with heavy traffic, fatigue could influence the design.

From Table 5.6, the same pattern can be seen where the emissions follow the weight change. The higher total life cycle assessment for stainless steel alternatives depends on the higher emission for the material compared to carbon steel. A bar chart comparing the different alternatives in terms of life cycle cost, life cycle assessment, and weight is visualized in Figure 5.5.

Table 5.6: The resulting Life Cycle Assessment [CO_2 eq.] of the run giving the smallest value of the life cycle cost for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
N_{obs}:	$0.05 \cdot 10^6$	$0.5 \cdot 10^6$	$0.05 \cdot 10^6$	$0.5 \cdot 10^6$
ADT:	$5 \cdot 10^3$	$50 \cdot 10^3$	$5 \cdot 10^3$	$50 \cdot 10^3$
LCA	231 000	232 000	240 000	239 000

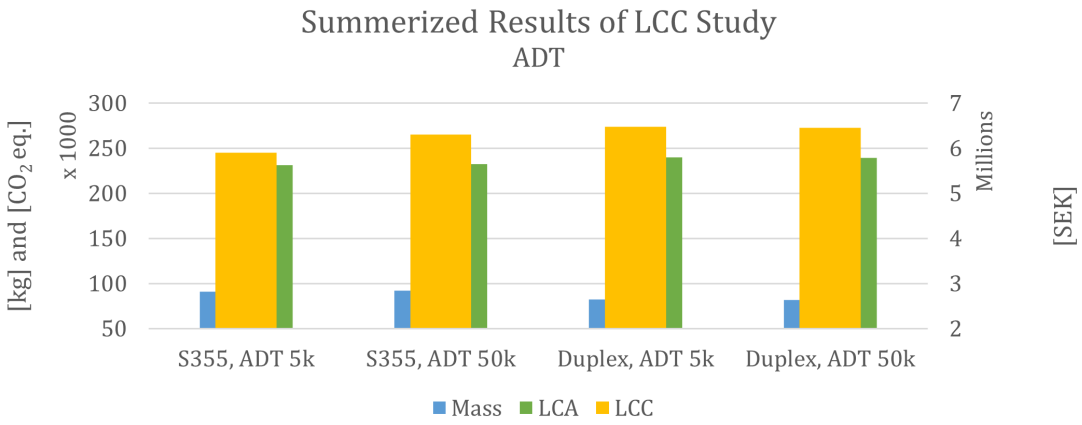


Figure 5.5: LCC ADT Study: The resulting Life Cycle Cost [SEK], mass [kg] and CO_2 equivalent of the run giving the smallest life cycle cost for each optimization alternative.

5.2.3 Decreased Material Cost

To see how the material price per kg affects the results, the total material cost in the Life Cycle Analysis was divided by two. The main reason for this is that the prices for the steel material have increased a lot while this master's thesis where written (January-June 2022). The steel prices that are used in this master's thesis are 20 SEK per kg for plates of steel grade S355, 25 SEK per kg for hot-rolled sections of steel grade S355, 60 SEK per kg for Duplex stainless steel plates, and 75 SEK per kg for Duplex hot-rolled sections. In the previous master's thesis by Henrysson and Yman (2020) investigating the same case, the stated material cost was 9 SEK per

kg for carbon steel S355, approximately half of today's price, and 20 SEK per kg for stainless steel, a third of today's price.

The results from the study are seen in Table 5.7 where X represents the results when today's material cost is considered and X/2 the results when half of today's material cost is considered. The results show clearly that by decreasing the material cost, the total life cycle cost also decreases. The alternative with Duplex stainless steel with corrugated webs is also the cheapest alternative in this study as the material cost is the highest post, while the production and maintenance are large contributes for the carbon steel alternative. The full dimension of each solution is found in Appendix D.

Table 5.7: The resulting cost [SEK] of the run giving the smallest value of the life cycle cost for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
Material cost:	X	X/2	X	X/2
Material	2 250 000	1 140 000	5 380 000	2 680 000
Production	1 920 000	1 860 000	1 130 000	1 130 000
Tot investment:	4 170 000	3 000 000	6 510 000	3 810 000
Maintenance:				
- Cost	1 700 000	1 690 000	0	0
- Consequence	50 000	50 000	0	0
Tot user cost:	1 750 000	1 740 000	0	0
Resell	-20 000	-20 000	-40 000	-40 000
Tot LCC:	**5 900 000**	**4 720 000**	**6 470 000**	**3 770 000**

Table 5.8: The resulting steel weight [kg] of the run giving the smallest value of the life cycle cost for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
Material cost:	X	X/2	X	X/2
Weight	91 000	92 300	82 100	81 700

Table 5.9: The resulting Life Cycle Assessment [CO_2 eq.] of the run giving the smallest value of the life cycle cost for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
Material cost:	X	X/2	X	X/2
LCA	231 000	233 000	240 000	239 000

The results from the weight and Life cycle analysis of the different alternatives are summarized in Table 5.8 and Table 5.9. As for increasing the average daily traffic and number of observed vehicles parameters, the weight and CO_2-eq is marginally affected by decreasing the material cost. The slight weight increase for the S355 alternative could be a result of the algorithm not prioritizing the steel mass as high with decreased price, but is more likely, as for the stainless steel alternative, a result of the randomness of the algorithm.

A summarizing bar chart comparing the different alternatives can be seen in Figure 5.6. One can notice that the cost reduction of the stainless steel alternative is larger than for the carbon steel alternative although the mass is constant for both alternatives, which is reasonable as the first is higher influenced by the material price and the latter equally by production and maintenance.

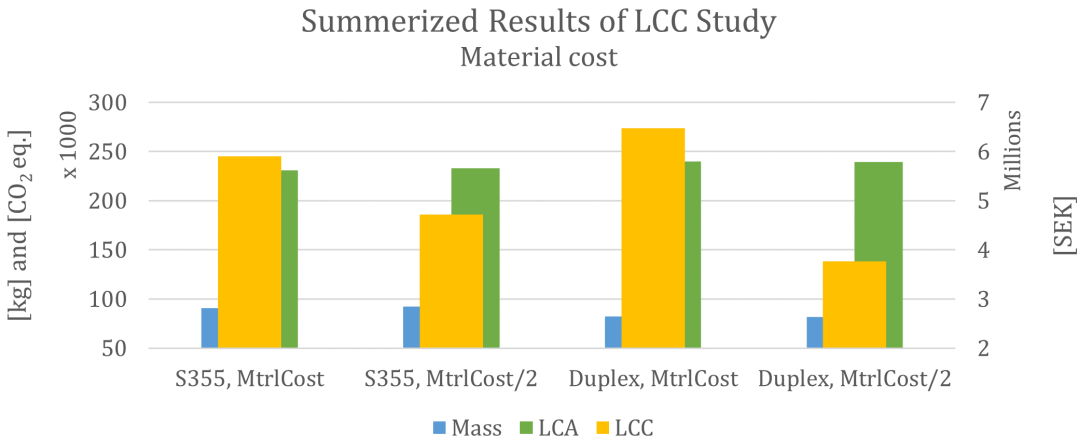


Figure 5.6: LCC Material Cost Study: The resulting Life Cycle Cost [SEK], mass [kg] and CO_2 equivalent of the run giving the smallest life cycle cost for each optimization alternative.

5.3 Mass Optimization

To understand the relation between the weight and cost of the bridge, an optimization is performed to minimize the weight. The study is conducted for web height domains 1 to 2 meters and 1 to 3 meters for both the carbon steel and the stainless steel alternatives. Similarly as for the life cycle cost optimization study, the run with the minimum weight was further analyzed in terms of life cycle cost and life cycle assessment. The results from the analysis are summarized in Table 5.10 to 5.12 and visualized in Figure 5.7. The full dimension of each solution is found in Appendix E. As for the life cycle cost study in Section 5.2.1, the life cycle cost analysis showed that when using a web height domain up to 2 meters, the carbon steel S355 alternative is the cheapest. However, when increasing the web height domain, the stainless steel alternative has the minimum total life cycle cost.

For the stainless steel alternatives the weights presented in this study are similar to what was seen when optimizing based on life cycle cost in Section 5.2.1, with variations concluded due to the randomness of the algorithm. The same is concluded for the carbon steel alternative with a web height less than 2 meters. For the 1 to 3 meter range, however, the weight of the carbon steel alternative is lower, while the total cost is higher. This is reasonable as for this alternative, the proportion of cost due to production and maintenance, dependent on the total steel surface area, is larger than the material cost. Likewise, it is noted, as for the web height life cycle cost study in Section 5.2.1, the total cost for the 1-3 m web alternative of carbon steel is higher than for the 1-2 meter alternative of the same material, although the weight is lower, also due to the increasing surface area. Further, the CO_2 emission are, as previously noted, highly dependent on the weight.

Table 5.10: The resulting cost [SEK] of the run giving the smallest mass for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
h_w domain:	1-2m	1-3m	1-2m	1-3m
Material	2 240 000	2 080 000	5 390 000	4 710 000
Production	2 010 000	2 220 000	1 140 000	1 170 000
Tot investment:	4 250 000	4 300 000	6 530 000	5 880 000
Maintenance:				
- Cost	1 870 000	2 090 000	0	0
- Consequence	50 000	50 000	0	0
Tot user cost:	1 920 000	2 140 000	0	0
Resell	-20 000	-20 000	-40 000	-30 000
Tot LCC:	**6 150 000**	**6 430 000**	**6 490 000**	**5 850 000**

Table 5.11: The resulting steel weight [kg] of the run giving the smallest mass for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
h_w domain:	1-2m	1-3m	1-2m	1-3m
Weight	90 000	81 700	82 400	70 300

Table 5.12: The resulting Life Cycle Assessment [CO_2 eq.] of the run giving the smallest mass for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
h_w domain:	1-2m	1-3m	1-2m	1-3m
LCA	230 000	221 000	240 000	222 000

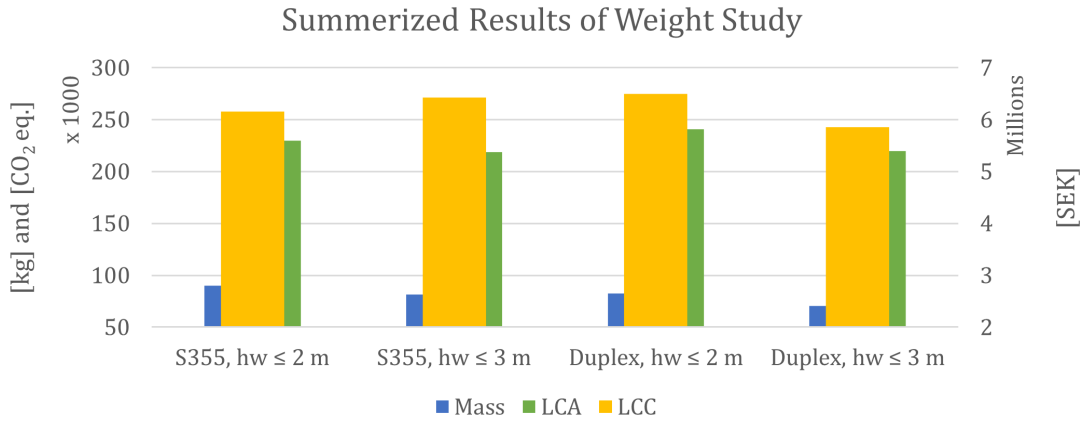


Figure 5.7: Mass Study: The resulting Life Cycle Cost [SEK], mass [kg] and CO_2 equivalent of the run giving the smallest mass for each optimization alternative.

5.4 Life Cycle Assessment Optimization

Lastly, an optimization based on life cycle assessment was performed and also analyzed in terms of life cycle cost and weight. The results can be seen in Table 5.13 to 5.15 and are visualized in Figure 5.8. The full dimension of each solution is found in Appendix F. The results of weight and emissions are close to the results from the mass optimisations, which makes sense as the CO_2 emissions are mostly dependent the material amount. There is noted some variations in the life cycle cost however, which is reasonable as neither of these studies aims to minimize the costs related to production and maintenance.

Table 5.13: The resulting cost [SEK] of the run giving the smallest LCA value for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
h_w domain:	1-2m	1-3m	1-2m	1-3m
Material	2 250 000	2 100 000	5 370 000	4 700 000
Production	2 030 000	2 220 000	1 130 000	1 110 000
Tot investment:	4 280 000	4 310 000	6 500 000	5 770 000
Maintenance:				
- Cost	1 910 000	2 200 000	0	0
- Consequence	50 000	50 000	0	0
Tot user cost:	1 960 000	2 240 000	0	0
Resell	-20 000	-20 000	-40 000	-30 000
Tot LCC:	**6 220 000**	**6 540 000**	**6 460 000**	**5 770 000**

Table 5.14: The resulting steel weight [kg] of the run giving the smallest LCA value for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
h_w domain:	1-2m	1-3m	1-2m	1-3m
Weight	90 300	82 700	81 900	70 400

Table 5.15: The resulting Life Cycle Assessment [CO_2 eq.] of the run giving the smallest LCA value for each optimization alternative.

	S355	**S355**	**Duplex**	**Duplex**
Shape:	Flat	Flat	Corrugated	Corrugated
h_w domain:	1-2m	1-3m	1-2m	1-3m
LCA	230 000	221 000	240 000	221 000

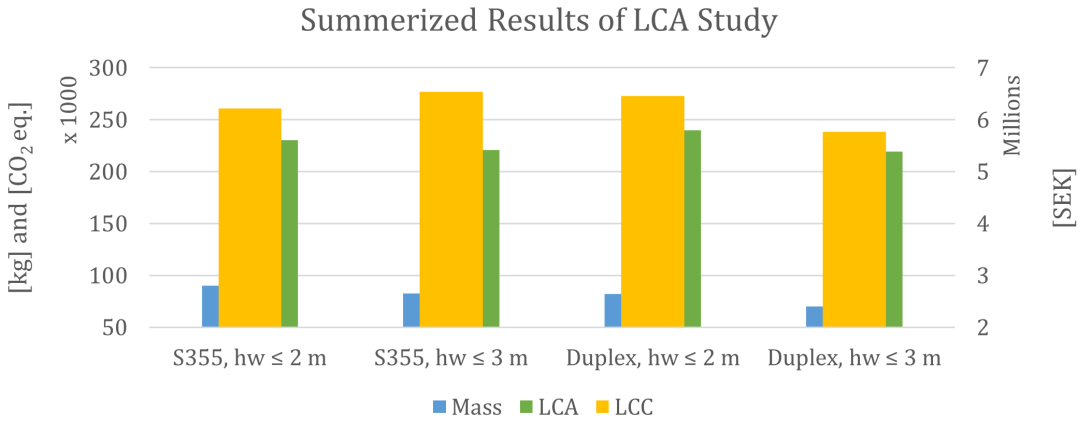


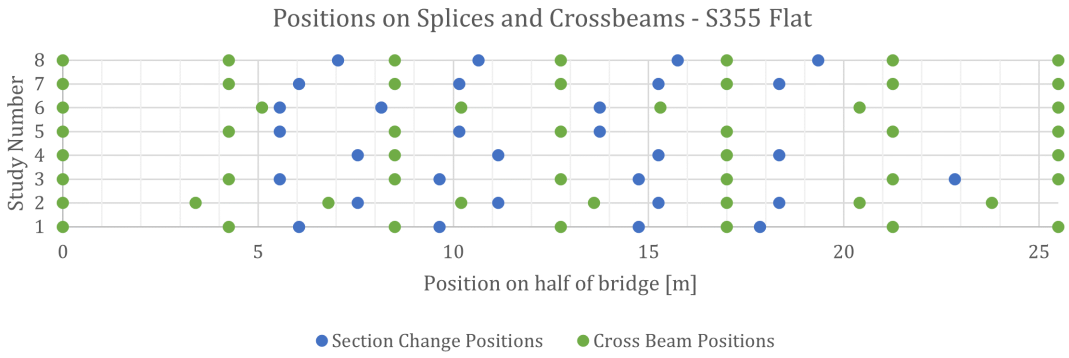
Figure 5.8: LCA Study: The resulting Life Cycle Cost [SEK], mass [kg] and CO_2 equivalent of the run giving the smallest mass for each optimization alternative.

5.5 Optimization Design Choices

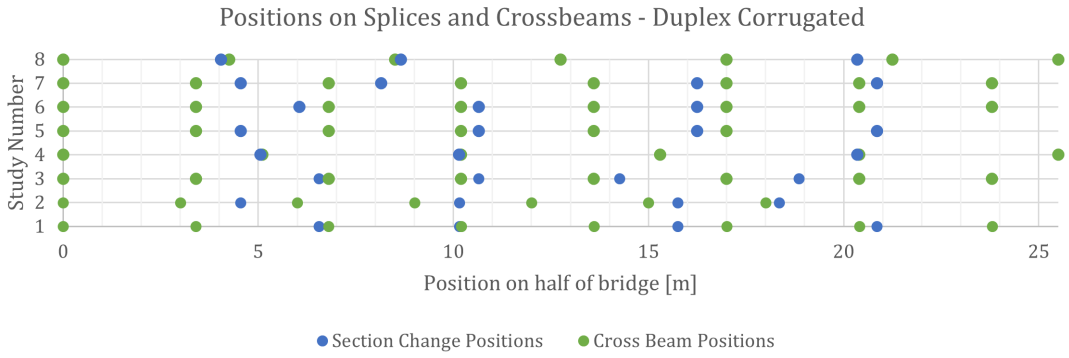
After performing all the optimization procedures, the results from all the optimizations were summarized to find trends and governing design choices. Comparisons were made for the position of the splices and crossbeams, flange width, web thickness, and corrugation parameters. The full results can be seen in Appendix D, E, and F. For the first study of the position of splices and crossbeams, only the run with the lowest objective were considered, while for the other studies all the performed runs were considered to get a wider set of data to analyze.

5.5.1 Positions of Splices and Crossbeams

As described in Section 4.1.2, the C-C distance between the crossbeams is a subject of the optimization and can be chosen within a domain of distances that are evenly divisible by the total span length. Likewise, the position of the longitudinal segment changes is subject to the optimization, restrained from choosing the same position as the crossbeams. The resulting positions of the splices and the crossbeams are plotted in Figure 5.9a for the S355 alternative with flat webs and in Figure 5.9b for the stainless alternative with corrugated webs. The study numbering is explained in Table 5.16.



(a) S355 flat web.



(b) Duplex corrugated web

Figure 5.9: Location of cross-section changes and crossbeams. Due to symmetry only half of the bridge is shown, meaning x-axis values from 0 to 25,5 meters.

Considering the crossbeams, some trends can be detected. For the carbon steel alternative, the C-C distance of 4.25 meters was chosen for a majority of the studies (5 of 8 times), while for the stainless steel alternative, the distance of 3.4 meters was chosen most often (5 of 8 times). The other studies had distances both larger and smaller than the most common distance. The choices did not seem to be governed by the lateral loads, as the same cross-section was chosen for all, but rather by the lateral-torsional buckling of the upper flanges and the shear buckling of the web, as the design was chosen so that no reduction due to buckling occurred. It was however noted, as small C-C distances was preferred, the program only needed the smallest

cross section available for the truss system and the second smallest for the simple beam. By this, the truss system turned out heavier due to more elements, leading the program to restrict also the total beam height to two meters as that is the set criteria for the use of simple beam.

Regarding the splices, for most cases, the length of each segment is rather even along the bridge. For the S355 flat web studies, it is noted that the program placed most splices within 5 and 20 meters from the support. Additionally, only four out of sixteen studies returned an optimal solution where only four splices instead of five was utilized. As seen in Figure 5.9a, for the S355 alternative the two runs with less section change was run 5 and 6, reduced material cost and increased ADT. As it is concluded that more section changes reduces the material usage, one explanation could be that for these runs the material cost was less important. For the stainless steel alternative, Figure 5.9b, run 4 and 8 resulted less section changes, LCC for web height up to 3 meters and LCA. Similarly, the reason could be that the material was already lowered largely by the increased web height. However, these conclusions would need to be supported by more data. Similarly, the resulting spice positions is scattered over a large domain, and therefore no trend could be noted.

Table 5.16: Numbering of the studies.

Study	Description	Numbering
Mass	hw <2m	1
	hw <3m	2
LCC	hw <2m	3
	hw <3m	4
	50e3 ADT	5
	0.5MtrlCost	6
LCA	hw <2m	7
	hw <3m	8

5.5.2 Flange Dimensions

For the flange design, it was chosen to look at the flange width rather than the flange thickness as the width is consistent along the whole bridge, while the flange thickness follows the moment distribution with increasing size closer to the mid-span. The values of the flange width parameters b_{fu} and b_{fo} (bottom flange and top flange) were collected for all runs made during the optimization study: in total 24 runs for each material. The domain from which the optimization program can choose the flange width is between 400 and 1500 millimeters with an interval of 50 millimeters.

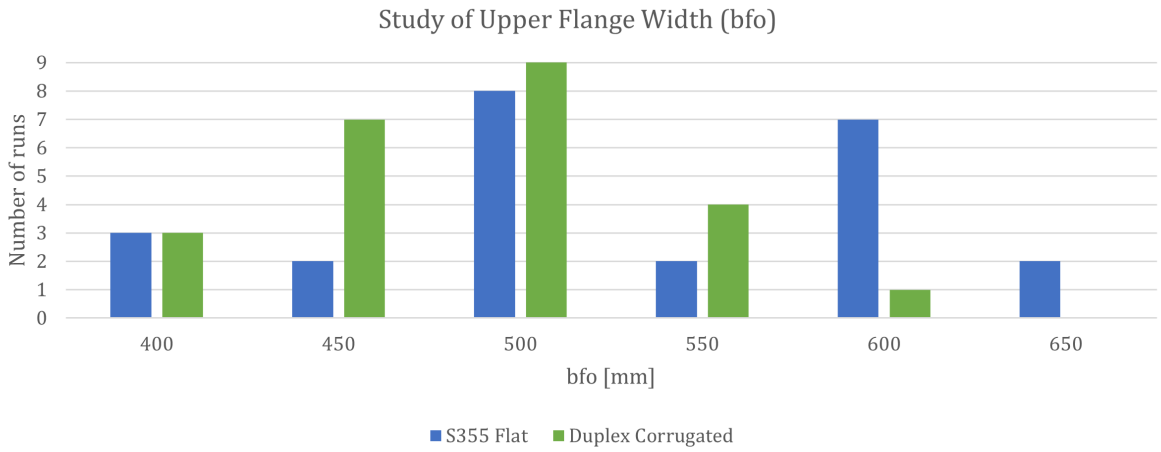


Figure 5.10: Summary of the optimized values for the width of the top flange from all of the runs of the optimization program.

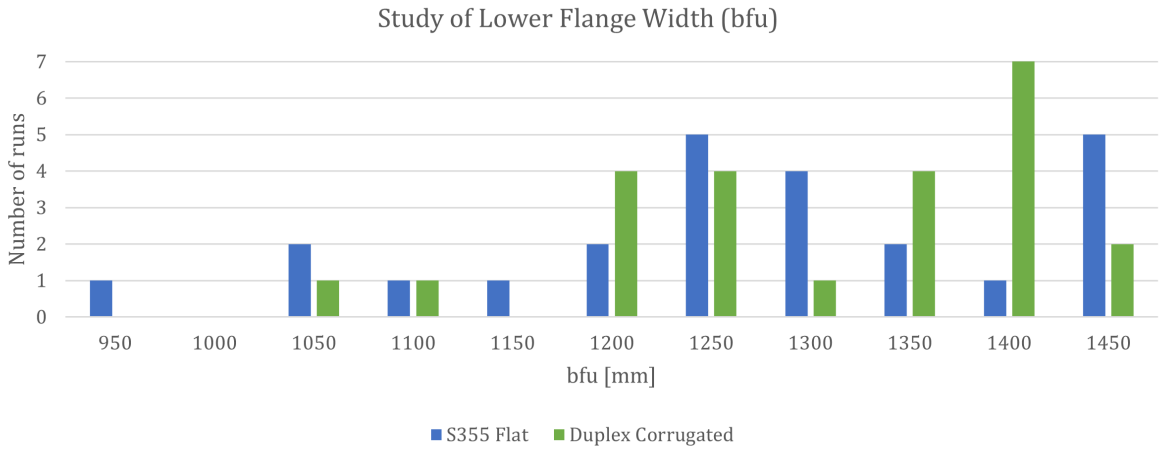


Figure 5.11: Summary of the optimized values for the width of the bottom flange from all of the runs of the optimization program.

Figure 5.10 show that the top flange chooses values between 400 and 650 millimeters for both S355 with flat web and duplex with corrugated web. For the duplex stainless steel alternative with corrugated web, a trend could be seen that it in most cases chooses the values of 500 millimeters, follow by 450 millimeters. For the alternative of carbon steel S355 with flat web, the values 500 and 600 millimeters are often chosen.

Figure and 5.11 show that the interval of the the width of the bottom flange is between 950 and 1450 millimeters. The general design chose for both stainless duplex steel with corrugated web and carbon steel of S355 with flat web is in the upper range of the domain, 1200 mm and up. The smaller widths are generally from the studies with increased web height which is reasonable as the distance between the flanges increases the moment of inertia and less material is therefore needed.

5.5.3 Web Dimensions

The web height was seen to be chosen close to the maximum allowed height also for the S355 flat web alternative even though it resulted in a higher total cost. For the S355 flat web alternative of maximum 2 meters, the program chose within the range of 1.89 to 1.99 meters and between 2.47 and 2.88 for the runs with the allowed maximum of 3 meters. For the duplex corrugated web alternative, the program chose 1.89 meters for all apart from one study in the 1-2 meter domain. For the 1-3 meter domain web height within the small range of 2.85 to 2.88 were chosen.

The web thickness was chosen from the domain [4, 6, 8, 10, 12, 14, 16, 18, 20, 25, 28, 30]. As the bar chart in Figure 5.12 visualizes, the duplex stainless steel alternative with corrugated web has lower web thicknesses than the carbon steel S355 alternative with flat web. This is expected as no reduction due to local buckling is needed, and is the main reason why the corrugated alternative is competitive in weight, especially for high girders. For the duplex stainless steel alternative, the web thickness is 8 millimeters when allowing a web height of up to 3 meters for all runs. When only allowing the web height of maximum 2 meters, the program choose 10 millimeters for all runs apart from two where 12 millimeters was used instead, however these two runs were never the most optimal within their study.

For the carbon steel S355 alternative with flat web, the most common value is 20 millimeters. This value is chosen for most of the alternatives where the web height is limited to 2 meters. However, when allowing a web height of up to 3 meters, there is a dissemination of the chosen thickness, both less and higher than 20 millimeters.

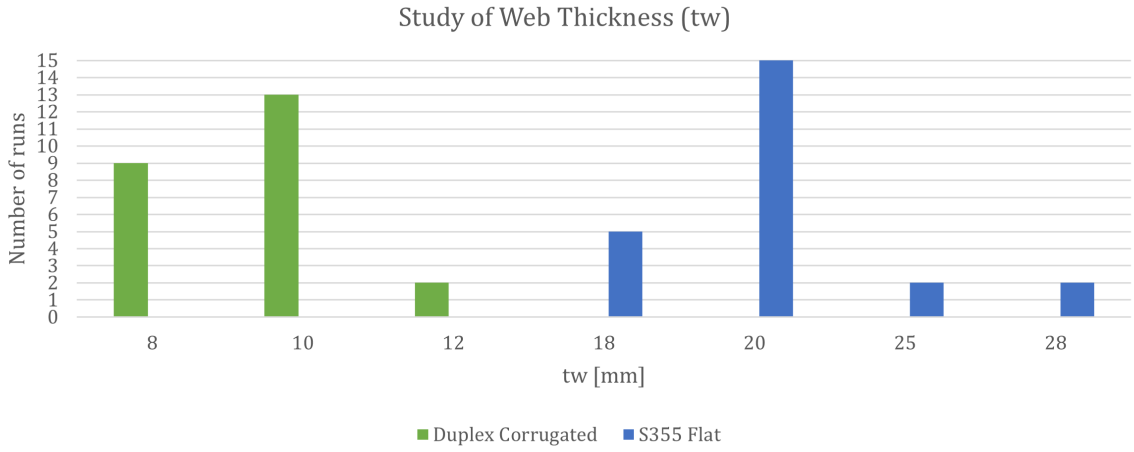


Figure 5.12: Summarize of the optimized values for the thickness of the web from all of the runs of the optimization program.

5.5.4 Corrugation Parameters

Lastly, a study was conducted on the chosen corrugation parameters for the stainless steel alternative with corrugated web, defined in Figure 5.13. The value of a_1, a_3 and α are chosen by the optimization program, while a_2 and a_4 are calculated based

on the chosen parameters and are therefore not studied. In the design calculations, the corrugation parameters mainly affect the shear buckling capacity described in Section 3.1.3, but also the flange outstand in Section 3.1.1 and the total length ratio, r_c, in Section 3.4.2.

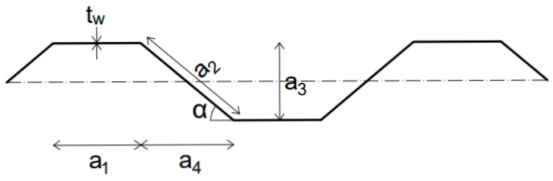


Figure 5.13: Trapezoidal corrugation parameters, previously shown in Figure 2.6b.

The results show that 30 degrees is the most common choice of the alpha-parameter (α) over all runs. Other values of the alpha parameter chosen were 31 and 32 degrees. However, when looking at the runs giving the best result in the optimization, all of them had the alpha value of 30 degrees.

The a_3 parameter was for all alternatives chosen to 50 millimeters when the web height was restricted to 2 meters, which is the lower bound value of the design domain. When increasing the allowed web height to 3 meters, 70 millimeters was selected every time. One reason for the program to try to minimize a_3 is to minimize the flange outstand to avoid width reduction due to risk of local buckling. Additionally, a higher value of a_3 increases the length ratio, r_c, an by it the gross web length of the girder and therefore the total steel weight.

The parameter a_1 had large dissemination from the extremely low value of 60 millimeters to the high value of 400 millimeters. For this parameter it is therefore hard to draw any conclusion without further studies.

5.6 Governing Design Modes

For the S355 flat web alternatives the shear utilization ratio rarely exceeded 90 %, while the moment utilization both in the construction and service phase was observed up to 100 % in most studies.

The stainless steel alternatives with corrugated webs on the other hand only returned one optimal solution with the web utilized less than 99 %. Likewise, the moment utilization ratio was up to 100 % in many studies. Additionally, as previously stated in 5.2.2, for the study with increased N_{obs} also the fatigue design was highly utilized (98 %) regarding the principle web stress.

6
Discussion

The discussion points and conclusions drawn from the master's thesis are all based on the result of the Sensibility Study (Section 4.3) and the Case Study (Chapter 5), which both studied a specific bridge. To get a more general overview and result, more studies of other bridges would be necessary. It is therefore important to remember that the conclusions may not be the same for a bridge with a different span length or loads. Further, in previous master's thesis studies on the subject, the comparison between the different alternatives made in carbon steel with flat webs and stainless steel with corrugated webs has been made by comparing the original bridge in carbon steel, which may not be as optimized, with an optimized alternative in stainless steel. In this master's thesis, however, the comparison between alternatives is made for different optimized alternatives, giving less difference between the alternatives. Additionally, in practice when designing a bridge, testing a wide range of solutions as the genetic algorithm does might not be possible to the same extent since the process is not often that automated. Therefore, the final design of the bridge may not be the most optimal one even though it was aimed for. Yet, it should be noted that the cost calculation might not include all practical considerations, where for example the high amount of section changes considered in these optimizations are only accounted for in the cost of welding, while potential material surplus due to limitations in the minimum quantity that can be purchased of a particular plate is not considered.

6.1 Optimization Program

The optimization program is developed so that different input data regarding the geometry, the location of the bridge, the construction specific data, and the fatigue parameters can be changed to be able to look at different bridges. However, there are currently some constraints in the program that could affect the results and some further studies that could improve both the running time and the optimal solution.

6.1.1 Design Limitations

The program only designs the steel part of the superstructure of the bridge and does not include the concrete deck, the substructure, or any specific connection details. It is also restricted to the steel grades S355, S460, and duplex 1.4162 and the concrete grades C30/37, C35/45, and C40/50. However, the program is arranged so that more materials can easily be added to the material module, as explained in Section

4.1.1. To make it more comprehensive it may be beneficial to add also the design of the concrete deck, the superstructure, and the different details. However, these design parts may be added outside of the optimization routine, or by doing another optimization for these parts since added calculations will increase the computational time.

Another aspect that could be beneficial to add, is to calculate the weld throat thickness, a, using the design stress in the weld instead of giving it as a fixed input. This would make the weld more optimized, like the rest of the structure, and would mainly reduce the welding cost as it is calculated based on the thickness and length of the weld. Furthermore, as discussed in Section 2.3, studies have showed that additional capacity could be achieved by corrugated webs, apart from what is currently included in Eurocode. As this is not, yet, proven and included it is neither in this optimization program.

Lastly, another limiting constraint in the program that may affect the result, is that the neutral axis always needs to be below the upper steel flange due to a simplification of the calculations. However, this may not always be the case, especially not for wider bridges with resulting thicker concrete decks. One possible development of the program could therefore be to add the calculation alternative of a structure with the neutral axis above the upper flange.

6.1.2 Limitations due to Computational Time

When optimizing against the life cycle cost and life cycle assessment, the database with the different values, by Nissan and Woldeyohannes (2022), was written in excel. The process of repeatedly reading the data from excel was very time-consuming, making the optimization routine take approximately two hours when using a population size of 150, 500 iterations, and an elite ratio of 0.3. However, when optimizing only against the mass, where an excel sheet was only loaded once, the time frame was half an hour using the same settings. To make the routine more flexible and to be able to increase the number of iterations and keep a reasonable time frame, the life cycle cost and life cycle assessment data could be accessed directly in a python module.

Further, to increase the convergence rate, the elite ratio parameter was increased to the highest value. The elite ratio determines which solutions of the current population that is kept in the next population. The higher the ratio is, the larger proportion of high-ranked solutions is kept for the next round. By increasing the elite ratio, the risk of finding a local optima is higher as specific variables not proving to be efficient in the first few iterations are disregarded, although they might give better results combined with other variables. One example of this could be seen in the results presented in Section 5.2.1, where the carbon steel alternative with a flat web returns a solution with increased cost when the height domain is broadened to also include webs in the 2-3 meters range. Here, the increased web height does

return a larger cost, however, it could have proven to be an initial good height as it decreases the moment utilization ratios. Due to the high elite ratio, this height might therefore have been kept in the future generations.

By decreasing the computational time, more runs and iterations could be done which is necessary to get more reliable results from the case study. In the case study, 500 iterations and three runs were made due to the time limitations explained above. Therefore, some of the conclusions could be a coincidence due to the randomness of the genetic algorithm and more runs could decrease the uncertainty in the results.

6.1.3 Suggested Additional Sensibility Studies

The sensibility study was somewhat limited due to the set time frame of the thesis work. The parameters were studied one by one, evaluated, and the specific setting was then decided. The studies regarding the crossover type, mutation probability, and the penalty constants were all made using a small population size and a small number of iterations. To make the study more reliable, these parameters would be needed to study again after the population size, the number of iterations, and the value of the elite ratio is determined to see if there are any changes or impact. Further, more values of the population size and number of iterations would be beneficial to study. Now, a population size of 50, 100, 150, 200, and 250 is examined, but it would be advantageous to also look at for example 75 and 125 to see if the size can decrease and thereby decrease the running time. The same reasoning is valid for the number of iterations and the elite ratio that more values of the parameters should be studied, and primarily, they should be studied together.

To make the optimization routine faster, the number of design parameters would also be beneficial to decrease. A study, similar to the study in Section 5.5, is therefore good to perform after several runs to determine if there are any parameters that the optimization algorithm often selects. By this, the parameter could be a set parameter instead of chosen by the algorithm. In the case study performed in the thesis, the alpha value determining the angle of the corrugation is for most cases chosen to 30 degrees. This parameter is a good example of when, instead of being a parameter chosen by the program, it could be an input parameter selected by the designer. Further, decreasing the domains would also help the program find an optimal solution in a shorter time. From the study of the web dimensions, it is obvious that the alternative with stainless steel with corrugated webs always chooses the values 8, 10, or 12 millimeters. The domain for this parameter could therefore be limited to these three values. The same conclusion could be drawn for the carbon steel alternative, however, here four alternatives are needed. Similar conclusions could be drawn from the summarized flange width results.

6.2 Optimization Results

Comparing the original design presented Section 5.1 with the result of the optimized design with the most similar constrains, the carbon steel alternative with web limited to two meters, the mass was reduced with 22 % and the total life cycle cost with 6 %, from which it is concluded that the optimization tool resulted in design improvements. Additionally, compared to the stainless steel alternative even larger material savings was seen, up to 30 %. However, due to the high material price of stainless steel, the total life cycle cost increased slightly. Following are discussions on the chosen parameters, methods and unintended programming effecting this result.

6.2.1 Web Design Choices

From the study of when the web height is increased, the results in Table 5.1 show that the stainless steel alternative with corrugated webs is more beneficial when allowing the web height domain of up to 3 meters. For bridges with a web height of up to 2 meters, the carbon steel alternative with flat webs is the cheapest. It was however noted that even though the corrugated web alternative benefit from increased height, the results did not show any webs very close to 2 meters. This is concluded due to the program choosing a lower web height to not get a total height larger than 2 meters. This, as earlier commented, to avoid the use of the truss bracing system. As an improvement, it is therefore suggested to set the bracing condition dependent on the web height and to chose the condition carefully in respect to the web height domain to get a fair result. Further, the total life cycle cost increases when increasing the web height domain for the carbon steel alternative, while for the stainless alternative the total life cycle cost decreases by increasing the web height domain. This is expected and also the reason why a corrugated web is beneficial to use compared to a flat web.

Additionally, as the corrugated web only depends on the shear, there could be an additional advantage of also allowing the web thickness vary between the segments for the stainless steel alternative with corrugated web. As the shear decreases towards the mid of the bridge, the web thickness is also expected to decrease. And as the web thickness has been seen to highly influence the total weight, this could possibly have a large effect on the cost of the stainless steel alternative as the material cost is the absolute largest part of the total cost. For the carbon steel flat web alternative however, the effect is suggested to not be as large. Partly as the flat web also contributes to the moment and deflection capacity and could then possibly result in the need of larger flanges, and partly as the weight is less important for the total cost due to the lower carbon steel price.

6.2.2 Effect of the Steel Prices

Other reasons for the increase in total cost when allowing a higher web height for the carbon steel alternative could be that each of the costs material, production, and maintenance has approximately the same proportion of the total price. There-

fore, the optimization program does not prioritize decreasing the material weight as much as it does for the stainless alternative where the material cost is the largest proportion of the total price.

In Table 5.1, it is also important to note the high material cost for the stainless steel alternatives compared to the carbon steel alternatives. The reason for this is because the accounted material price for stainless steel is three times higher than carbon steel, 60 SEK per kilo compared to 20 SEK per kilo. It is therefore of great interest to look at how the material price affects the optimization to gain knowledge of when each alternative is more beneficial. Due to current circumstances, the price of steel material has almost three doubled in the last years. A further study where therefore made when increasing the material price to half of the original price. The result in Table 5.7 shows that the stainless alternative then became cheaper even with a web height domain of 1-2 meters.

To conclude on this topic, the stainless steel alternative could in the future show to be the cheapest option, depending on how the material price develops. This study showed that if the material price of stainless steel is less or equal to three times the carbon steel price, a stainless bridge is economically reasonable from a life cycle perspective. It could also be suggested from the result in Section 5.2.3, that if the price of stainless steel would further reduce to double the carbon steel price, it would also prove to be reasonable from an investment point of view for girders with higher allowed web height.

6.2.3 Amount of Traffic

Further studies were also made on the average daily traffic, ADT and the number of observed vehicles, N_{Nobs}. When increasing both parameters, the total life cycle cost increased for the carbon steel alternative as the ADT only affects the maintenance cost included in the carbon steel alternative. It should however be noted that the ADT does not include any specific local constraints or demands that could further increase the maintenance cost for the carbon steel alternative.

The N_{Nobs} affect the fatigue calculations, but as the design for both alternatives were rather governed by moment and shear capacities, it did not affect the cost. However, it was seen to increase the fatigue utilization rate, and as the design procedure for fatigue does not consider the steel strength, this can be crucial for higher strength steel as Duplex.

6.2.4 Choice of Optimization Objective

When performing the mass optimization, the results show very similar results for the stainless steel alternatives regarding the weight when comparing them to the optimization with the goal to minimize life cycle cost, as seen in Figure 6.1. From Figure 6.2, also the cost is very similar for the lower height domain, however there is a difference when looking at the higher domain due to increased production costs

that is not considered in the mass optimization.

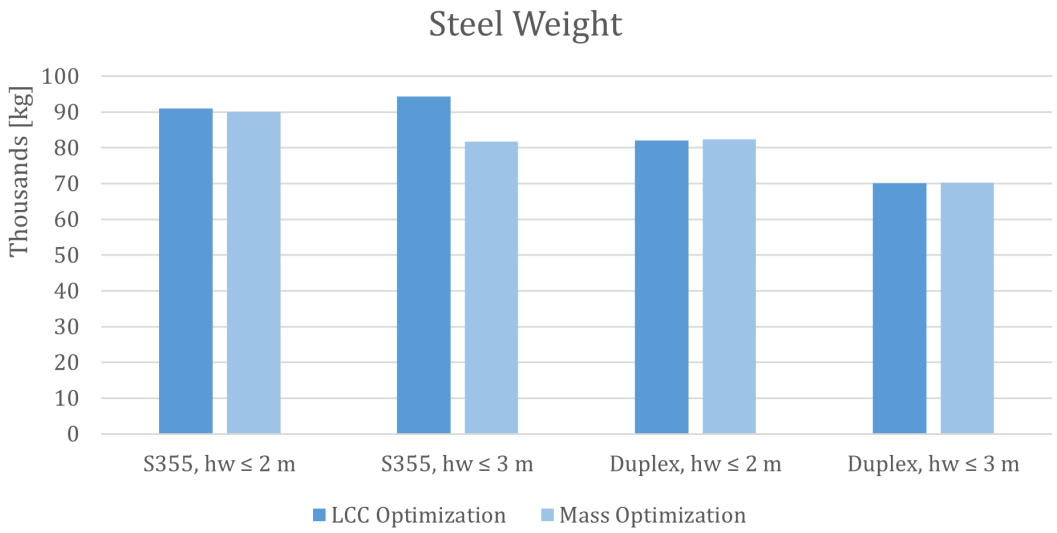


Figure 6.1: Summery of resulting weights of the LCC and mass optimizations, Table 5.2 and 5.11

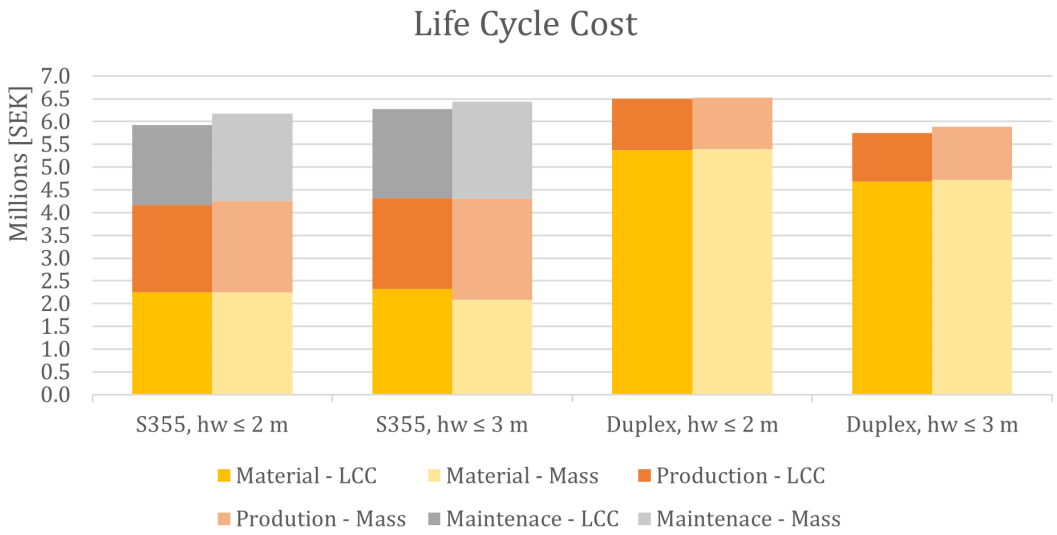


Figure 6.2: Summery of resulting costs of the LCC and mass optimizations, Table 5.1 and 5.10

For the carbon steel alternatives the weight decreased when optimizing targeting mass, especially for the case with higher web domain, as seen in Figure 6.1. However, Figure 6.2, show that the total cost was increases for both cases. As earlier discussed, this could be that the proportion of the material cost in the carbon steel study is approximately a third of the total price while for the stainless steel alternative it is the largest proportion of the total price. A conclusion from this study could therefore be that it is enough for the stainless steel alternative to optimize against

the weight for a lower height domain since it returns a similar total cost. However, for the carbon steel alternative and generally for higher girders, optimizing against the total life cycle cost will give a more optimized result in terms of a lower total cost.

As seen in Figure 6.3, The Life Cycle Assessment Study gave similar results as the mass study which is not surprising since the emission mainly depend on the amount of material, where the emissions for stainless steel is higher per kg than for carbon steel. As seen in Figure 6.4 the material cost is similar between the studies, especially for the carbon steel alternative. However the production and maintenance cost vary in some cases which is reasonable as neither of these studies aims to minimize these.

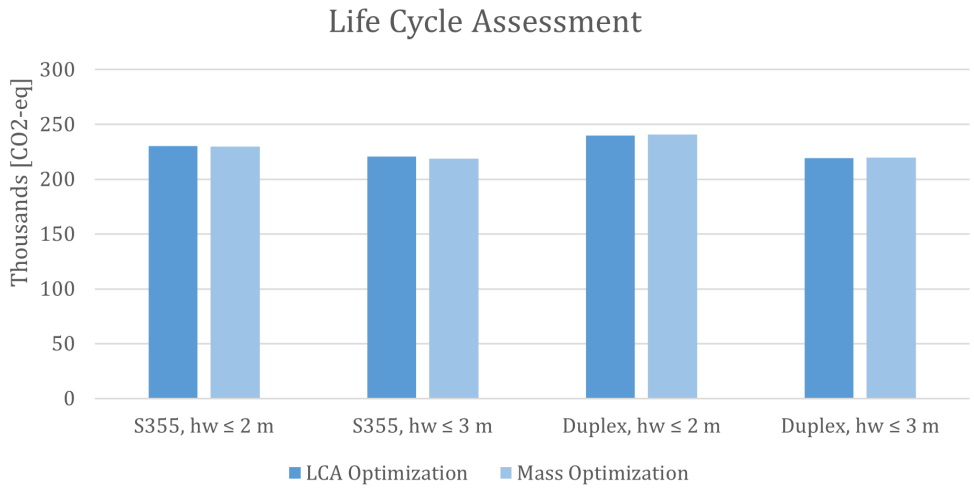


Figure 6.3: Summery of resulting emissions of the mass and LCA optimizations, Table 5.12 and 5.15

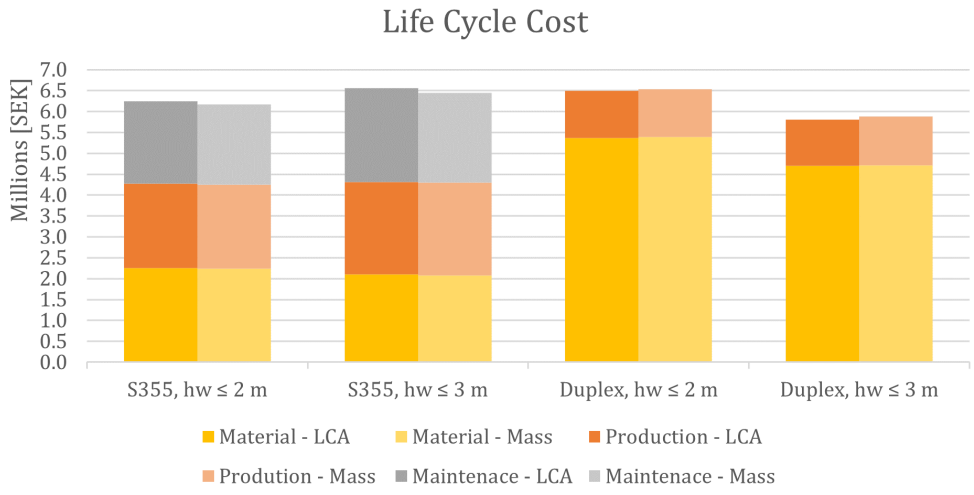


Figure 6.4: Summery of resulting costs of the mass and LCA optimizations, Table 5.10 and 5.13

7
Conclusion

The aim of the thesis was to develop a design optimization program to optimize the design of the intended bridge concepts from different aspects including material weight as well as Life Cycle Cost and Life Cycle Assessment and use it to compare two alternative composite road bridges: one with carbon steel girders with flat webs and one with stainless steel girders with corrugated webs. Some final concluding remarks on the efficiency of the optimization program are as follows:

- The program did not return one single optimal solution, rather a set of solutions close in cost and material mass. To help the program to find the true optimal it is suggested to reduce the domain range and fix variables that was proven to give a consistent value. Suggested variables to fix is the web height and the corrugation parameters α and a_3 according to the results in Section 5.5. Suggested domains to limit is the widths of both the upper and lower flange, as well as the web thickness, also after the results shown in Section 5.5. Note that these suggestions is only valid for this specific bridge. For other bridges a similar study as presented in Chapter 5 is recommended to start with.

- Reducing the number of design variables chosen by the program was also concluded to reduce the convergence time.

- Additionally, the algorithm parameter 'elite ratio' was proven to increase the convergence rate of the optimization program. However, it was also suggested to lead to local optima when the domain range increased.

Additionally, from the optimization results the following was concluded:

- The optimization program was concluded efficient, with 20 % less material use and 6 % less total life cycle cost considering the carbon steel alternative compared to the original design. The stainless steel alternative showed even larger material savings up to 30 %, however with to similar cost due to the high material price of stainless steel.

- The stainless steel alternative was seen to highly depend on the material cost since the same result was obtained regardless if the optimization was aimed to minimize the life cycle cost or the material mass. For the carbon steel alternative, the lowest cost was proven to be returned by the life cycle cost optimization rather than the mass optimization due to the production and

maintenance costs being dependent on other factors like the steel surface area.

- Increased traffic on the bridge resulted in increased cost for the carbon steel alternative while the cost for the stainless steel alternative remained unchanged since there are no maintenance costs.

- The cost efficiency for the stainless steel alternatives was seen strongly dependent on the material price of stainless steels in relation to the price of carbon steels. For the current prices (60 SEK/kg Duplex stainless steel and 20 SEK/kg S355 carbon steel), the stainless alternative is 9 % more expansive. While if the prices of both steels where reduced to half, the stainless steel alternative was proven to be 20 % cheaper.

- Also, by decreasing the total steel mass with the use of corrugated webs and higher girders (above 2 meters), the stainless steel bridge was proven a competitive alternative to the more traditionally used carbon steel alternative by 3 %.

- As a last note, it is important to remark that this conclusion may only be valid for the bridge chosen to study during the case study and not for all bridges. For a more general conclusion more bridges would need to be studied.

Bibliography

Blank, J. and Deb, K. (2020). Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509.

Bozorg-Haddad, O., Solgi, M., and Loáiciga, H. A. (2017). *Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization.* John Wiley & Sons, Incorporated.

Chisari, C. and Amadio, C. (2018). Tosca: a tool for optimisation in structural and civil engineering analyses. *International Journal of Advanced Structural Engineering*, 10:401–419.

Francis, R. and Byrne, G. (2021). Duplex stainless steels—alloys for the 21st century. *MDPI: Metals*, 11.

Górecki, M. and Śledziewski, K. (2021). Influence of corrugated web geometry on mechanical properties of i-beam: Laboratory tests. *Materials*, 15:277.

Henrysson, A. and Yman, E. (2020). *Design of Composite Steel-Concrete Bridges using Stainless Steel Girders with Corrugated Webs - A Study of the Applicability and Effectiveness of using Stain-less Steel Girders with Corrugated Webs in Bridge Design.* Master's thesis, Chalmers University of Technology, Gothenburg.

Hlal, F. and Mohra, N. (2021). *Shear behavior and imperfection sensitivity analysis of Stainless Steel girders with corrugated web plates.* Master's thesis, Chalmers University of Technology.

Holland, J. H. (1992). Genetic algorithms. *Scientific American*, 267:66–72.

Hosseinpour, E., Baharom, S., and Yadollahi, Y. (2015). Evaluation of steel shear walls behavior with sinusoidal and trapezoidal corrugated plates. *Advances in Civil Engineering*, 2015.

Hällmark, R. (2018). *Innovative ways of achieving composite action.* PhD thesis, Luleå University of Technology.

ISSF (2019). The stainless steel family. Retrieved January 2022 from https://www.worldstainless.org/Files/issf/non-image-files/PDF/TheStainlessSteelFamily.pdf.

Johansson, B., Maquoi, R., Sedlacek, G., Müller, C., and Beg, D. (2007). *Commentary and worked examples to en 1993-1-5 "plated structural elements" - Background documents in support to the implementation, harmonization and further development of the Eurocodes.* European Communities.

Karlsson, E. (2018). *Stainless Steel Bridge Girders with Corrugated Webs Efficiency, stability and life-cycle cost analysis*. Master's thesis, Chalmers University of Technology.

Larsson, M. and Persson, J. (2013). *Lateral-torsional buckling of steel girders with trapezoidally corrugated webs*. Master's thesis, Chalmers University of Technology.

Nissan, A. and Woldeyohannes, Y. (2022). *Life cycle assessment, LCA, and life cycle cost, LCC, for composite bridges - Corrugated web stainless steel girders vs. flat web carbon steel girders*. Master's thesis, Chalmers University of Technology.

Pasternak, H. and Kubieniec, G. (2010). Plate girders with corrugated webs. *Journal of Civil Engineering and Management*, 16:166–171.

Sarraf, R. E., Iles, D. C., Momtahan, A., Easey, D., Hicks, S. J., and Agency, N. T. (2013). *Steel-concrete composite bridge design guide*. NZ Transport Agency.

Sayed-Ahmed, E. Y. (2007). Design aspects of steel i-girders with corrugated steel webs. *Electronic Journal of Structural Engineering*.

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

Solgi, M. (2020). PyPI: geneticalgorithm. Retrieved March 2022 from https://pypi.org/project/geneticalgorithm/.

Steffner, J. and Öman, M. (2021). *Design of Continuous Composite Road Bridges - Bridge girders with corrugated webs in stainless steel*. Master's thesis, Chalmers University of Technology, Gothenburg.

Stålbyggnadsinstitutet (2017). *Dimensionering av konstruktioner i rostfritt stål*. SBI, Stockholm.

Swedish Standards Institute (1990). *Eurocode 0: Basis of structural design (SS-EN 1990)*. www.sis.se.

Swedish Standards Institute (2002a). *Eurocode 1: Actions on structures – Part 1-1: General actions – Densities, self-weight, imposed loads for buildings (SS-EN 1991-1-1)*. www.sis.se.

Swedish Standards Institute (2002b). *Eurocode 1: Actions on structures – Part 1-4: General actions – Wind actions (SS-EN 1991-1-4)*. www.sis.se.

Swedish Standards Institute (2002c). *Eurocode 1: Actions on structures – Part 1-5: General actions – Thermal actions (SS-EN 1991-1-5)*. www.sis.se.

Swedish Standards Institute (2002d). *Eurocode 1: Actions on structures – Part 2: Traffic loads on bridges (SS-EN 1991-1-5)*. www.sis.se.

Swedish Standards Institute (2002e). *Eurocode 2: Design of concrete structures - Part 1-1: General rules and rules for buildings (SS-EN 1992-1-1)*. www.sis.se.

Swedish Standards Institute (2003). *Buildings and constructed assets – Service life planning – Part 1: General principles (SS-ISO 15686-1)*. www.sis.se.

Swedish Standards Institute (2005a). *Eurocode 3: Design of steel structures – Part 1-1: General rules and rules for buildings (SS-EN 1993-1-1).* www.sis.se.

Swedish Standards Institute (2005b). *Eurocode 3: Design of steel structures – Part 1-4: General rules – Supplementary rules for stainless steels (SS-EN 1993-1-4).* www.sis.se.

Swedish Standards Institute (2005c). *Eurocode 3: Design of steel structures – Part 1-5: Plated structural elements (SS-EN 1993-1-5).* www.sis.se.

Swedish Standards Institute (2005d). *Eurocode 3: Design of steel structures – Part 1-9: Fatigue (SS-EN 1993-1-9).* www.sis.se.

Swedish Standards Institute (2005e). *Eurocode 3: Design of steel structures – Part 2: Steel bridges (SS-EN 1993-2).* www.sis.se.

Swedish Standards Institute (2005f). *Eurocode 3: Design of steel structures –Part 1-8: Design of joints (SS-EN 1993-1-8).* www.sis.se.

Swedish Standards Institute (2005g). *Eurocode 4: Design of composite steel and concrete structures – Part 1-1: General rules and rules for buildings (SS-EN 1994-1-1).* www.sis.se.

Swedish Standards Institute (2005h). *Eurocode 4 – Design of composite steel and concrete structures – Part 2: General rules and rules for bridges (SS-EN 1994-2).* www.sis.se.

Swedish Standards Institute (2006). *Environmental management – Life cycle assessment – Principles and framework (SS-EN ISO 14040:2006).* www.sis.se.

Swedish Standards Institute (2014). *Stailess steels - Part 1: List of stainless steels(SS-EN 10088-1:2014).* www.sis.se.

Swedish Standards Institute (2015). *Eurocode 3: Design of steel structures – Part 1-4: General rules – Supplementary rules for stainless steels (SS-EN 1993-1-4 A1:2015).* www.sis.se.

Trafikverket (2019). *Krav Brobyggande (TDOK 2016:0204).* www.sis.se.

Trafikverket (2022). *Vägars of gators utformning (2022:003).* www.sis.se.

Transportstyrelsen (2018). *Transportsstyrelsens föreskrifter och almänna råd om tillämpning av eurokoder (TSFS-2018:57).* www.sis.se.

Vayas, I. and Iliopoulos, A. (2013). *Design of Steel-Concrete Composite Bridges to Eurocodes.* CRC Press, Taylor & Francis group.

Yang, X. (2010). *Engineering optimization : An introduction with metaheuristic applications.* John Wiley & Sons, Incorporated.

A

Load Combinations

The design loads for the different states are determined by load combination and the following tables describe the factors to use for Ultimate Limit State and Serviceable Limit State. For Ultimate Limit State, two different combinations are used, 6.10a and 6.10b, which determines the design load for STR/GEO (material failure). In Serviceable Limit State, two combinations are used. The first one, Equation 5.15b is used for reversible limit states for calculations of deflection while 6.16b is used for long-term effects.

Self-weight includes all the different parts of the structure except the concrete cover. This is because in Section 5.2.3 (4) in SS-EN 1991-1-1 there should be a deviation between the upper and lower value of the self-weight of the concrete cover which is regulated to ± 10 % according to TSFS 2018:57. Therefore the upper value, *sup*, is set to 1.1 and the lower value, *inf*, to 0.9. The recommended value for ψ_0 for temperature loads can in many cases be set to zero in the Ultimate Limit State (except fatigue) (SS-EN 1990, Table A.2.1) for cross-sections in class 1 and 2 (SS-EN 1994-1-1, Section 5.4.2.5 (2)). The factor γ_{sh} has a recommended value of 1.0 (SS-EN 1992-1-1, Section 2.4.2.1). The variable loads only include unfavorable actions since favorable actions are set to zero.

Table A.1: Ultimate limit state, STR/GEO. Equation 6.10a (Transportstyrelsen, 2018, Table 4.4)

Permanent	**sup**	**inf**	**Unfavourable**	**Favourable**	**Reference**
Self-weight	1.0	1.0	$\gamma_d 1.35 G_{k,j,sup}$	$1.0 G_{k,j,inf}$	SS-EN 1990 Section 4.1.2 (5)
Concrete cover	1.1	0.9	$\gamma_d 1.35 G_{k,j,sup}$	$1.0 G_{k,j,inf}$	SS-EN 1991-1-1 Section 5.2.3 (4) TSFS 2018:57 5 Ch 3 §
Shrinkage	1.0	1.0	$\gamma_{sh} G_{k,j,sup}$	$\gamma_{sh} G_{k,j,sup}$	SS-EN 1990 Section 4.1.2 (3) SS-EN 1992-1-1 Section 2.4.2.1

.

Variable	ψ_0	**Main load**	**Other load**	**Reference**
Traffic load				
- Distributed	0.75	$\gamma_d 1.5 \psi_{0,1} Q_{K,1}$	$\gamma_d 1.5 \psi_{0,i} Q_{K,i}$	TSFS 2018:57
- Point load	0.4			Table 4.2
Acceleration	0.75	$\gamma_d 1.5 \psi_{0,1} 0.6 Q_{K,1}$	$\gamma_d 1.5 \psi_{0,i} 0.6 Q_{K,i}$	TSFS 2018:57 Table 4.2 TSFS 2018:57 11 Ch 5 §
Temperature	0.6	$\gamma_d 1.5 \psi_{0,1} Q_{K,1}$	$\gamma_d 1.5 \psi_{0,i} Q_{K,i}$	SS-EN 1990 Table A.2.1

Table A.2: Ultimate limit state, STR/GEO. Equation 6.10b (Transportstyrelsen, 2018, Table 4.4)

Permanent	**sup**	**inf**	**Unfavourable**	**Favourable**	**Reference**
Self-weight	1.0	1.0	$1.35\gamma_d 0.89 G_{k,j,sup}$	$1.0 G_{k,j,inf}$	SS-EN 1990, Section 4.1.2 (5)
Concrete cover	1.1	0.9	$1.35\gamma_d 0.89 G_{k,j,sup}$	$1.0 G_{k,j,inf}$	SS-EN 1991-1-1 Section 5.2.3 (4) TSFS 2018:57 5 Ch 3 §
Shrinkage	1.0	1.0	$\gamma_{sh} G_{k,j,sup}$	$\gamma_{sh} G_{k,j,sup}$	SS-EN 1990 Section 4.1.2 (3) SS-EN 1992-1-1 Section 2.4.2.1

.

Variable	ψ_0	**Main load**	**Other load**	**Reference**
Traffic load - Distributed - Point load	 0.75 0.4	$\gamma_d 1.5 Q_{K,1}$	$\gamma_d 1.5 \psi_{0,i} Q_{K,i}$	TSFS 2018:57 Table 4.2
Acceleration	0.75	$1.5\gamma_d 0.6 Q_{K,1}$	$1.5\gamma_d 0.6 \psi_{0,i} Q_{K,i}$	TSFS 2018:57 Table 4.2 TSFS 2018:57 11 Ch 5 §
Temperature	0.6	$\gamma_d 1.5 Q_{K,1}$	$\gamma_d 1.5 \psi_{0,i} Q_{K,i}$	SS-EN 1990 Table A.2.1

Table A.3: Servicable limit state, frequent load combination. Equation 6.15b.

Permanent	**sup**	**inf**	**Unfavourable**	**Favourable**	**Reference**
Self-weight	1.0	1.0	$1.0G_{k,j,sup}$	$1.0G_{k,j,inf}$	SS-EN 1990 Section 4.1.2 (5)
Concrete cover	1.1	0.9	$1.0G_{k,j,sup}$	$1.0G_{k,j,inf}$	SS-EN 1991-1-1 Section 5.2.3 (4) TSFS 2018:57 5 Ch 3 §
Shrinkage	1.0	1.0	$\gamma_{sh}G_{k,j,sup}$	$\gamma_{sh}G_{k,j,sup}$	SS-EN 1990& Section 4.1.2 (3) SS-EN 1992-1-1 Section 2.4.2.1
Variable	ψ_1	ψ_2	**Main load**	**Other load**	**Reference**
Traffic load					
- Distributed	0.75	0	$\psi_1 Q_{K,1}$	$\psi_2 Q_{K,i}$	TSFS 2018:57
- Point load	0.4	0			Table 4.2
Acceleration	0.75	0	$\psi_1 0.6 Q_{K,1}$	$\psi_2 0.6 Q_{K,i}$	TSFS 2018:57 Table 4.2 TSFS 2018:57 11 Ch 5 §
Temperature	0.6	0.5	$\psi_1 Q_{K,1}$	$\psi_2 Q_{K,i}$	SS-EN 1990 Table A.2.1

Table A.4: Serviceable limit state, quasi-permanent load combination. Equation 6.16b.

Permanent	**sup**	**inf**	**Unfavourable**	**Favourable**	**Reference**
Self-weight	1.0	1.0	$1.0G_{k,j,sup}$	$1.0G_{k,j,inf}$	SS-EN 1990 Section 4.1.2 (5)
Concrete cover	1.1	0.9	$1.0G_{k,j,sup}$	$1.0G_{k,j,inf}$	SS-EN 1991-1-1 Section 5.2.3 (4) TSFS 2018:57 5 Ch 3 §
Shrinkage	1.0	1.0	$\gamma_{sh}G_{k,j,sup}$	$\gamma_{sh}G_{k,j,sup}$	SS-EN 1990& Section 4.1.2 (3) SS-EN 1992-1-1 Section 2.4.2.1
Variable	ψ_1	ψ_2	**Main load**	**Other load**	**Reference**
Traffic load					
- Distributed	0.75	0	$\psi_2 Q_{K,1}$	$\psi_2 Q_{K,i}$	TSFS 2018:57
- Point load	0.4	0			[Table 4.2]
Acceleration	0.75	0	$\psi_2 0.6 Q_{K,1}$	$\psi_2 0.6 Q_{K,i}$	TSFS 2018:57 Table 4.2 TSFS 2018:57 11 Ch 5 §
Temperature	0.6	0.5	$\psi_2 Q_{K,1}$	$\psi_2 Q_{K,i}$	SS-EN 1990 Table A.2.1

B

Case Study Input Data

Input data for the case study presented in Chapter 5: Bridge over Delångersån in Näsviken.

Table B.1: Bridge specific geometrical fixed design parameters.

Parameter	**Value**
L	51000 mm
SubDiv	500 mm
w	3250 mm
Nw	2
PC1	3000 mm
PC2	500 mm
a	5 mm
b0	330 mm

Table B.2: Bridge specific material fixed design parameters.

Parameter	**Value**
SteelGrade	'S355', 'S460' or 'Duplex'
Shape	'Flat' or 'Corrugated'
ConcreteGrade	'C35_45'
PavementType	'AsphaltConcrete'
hp	50 mm
ReinforcementGrade	'B500B'
CementClass	'S'
d_re	20 mm
d_stud	22 mm
h_stud	200 mm

Table B.3: Bridge specific environmental fixed design parameters.

Parameter	**Unit**
RH	80 %
T_max	35 C°
T_min	-36 C°
vb	23 m/s
qp	0.75 kN/m^2
ADT	5000 or 50000

Table B.4: Bridge specific construction fixed design parameters.

Parameter	**Unit**
Delta_T_cs	15 C°
T0	10 days
ts	1 days
t0	7 days
t0_cs	1 days
SafetyClass	3

Table B.5: Bridge specific fatigue fixed design parameters.

Parameter	**Definition**
ServiceLife	120 years
Method	'SafeLife'
Consequence	'High'
Nobs	$0.05 \cdot 10^6$ or $0.5 \cdot 10^6$

Table B.6: Genetic algorithm parameters used in the case study.

Parameter	**Value**
a_{pen} (Mass)	10^9
a_{pen} (LCC)	10^{12}
a_{pen} (LCA)	10^{10}
b_{pen}	1
C_{pen}	a_{pen}
Population	150
Iterations	500
Mutation probability	0.2
Elite ratio	0.3
Crossover portion	0.3
Crossover type	'uniform'
Max iteration without improv	None

C

Python Code

The appendix includes the python code written for this thesis: the optimization program (parametric program with the genetic algorithm applied) and the different modules. It does however not include the Life Cycle Cost and Life Cycle Analysis functions developed by Nissan and Woldeyohannes (2022).

C.1 Optimization Program

```
# Master's Thesis
#
# Design Optimization of Composite Road Bridges using Genetic Algorithms
# -Corrugated Web Stainless Steel Girders versus Flat Web Carbon Steel Girders
#
# Cecilia Hallgren & Vilma Johansson
# June 2022
#
#-------------------------------------------------------------------------------
#---------------------------------- CONTENTS -----------------------------------
# 1. INPUT DATA
#   1.1 Geometrical Input Data - not Subjected to the Optimization
#   1.2 Width of the Bridge
#   1.3 Number of Longitudinal Bridge Segments and Splices
#   1.4 Material Input
#   1.5 Environmental
#   1.6 Construction
#   1.7 Fatigue
#   1.8 Imported Modules
# 2. CONCRETE DESIGN
#   2.1 Material Parameters
#   2.2 Height of the Concrete Deck [mm]
#   2.4 Modular Ratio
#   2.4 Shrinkage Force
#   2.5 Shear Stud Capacity
# 3. OPTIMIZATION INPUT
#   3.1 Design Domains
# 4. MAIN GIRDERS
#   4.1 Design Variables
#   4.2 Constraints
#   4.3 Corrugation Parameters [class]
#   4.4 Segment Coordinates and Position [mm], [mm], [-]
#   4.5 Material Parameters (web)
#   4.6 Loads (mean value)
#   4.7 Vectors for SLS Calculations (average value)
#   4.8 Design Procedure per Segment
#       4.8.1 Material Parameters (flanges)
#       4.8.2 Cross-section Class
#   4.9 CONSTRUCTION PHASE
#       4.9.1 Sectional Properties
#       4.9.2 Structural Analysis - ULS
#       4.9.3 Shear Capacity
#       4.9.4 Bending Moment Capacity
#       4.9.5 Verification
#       4.9.6 Assumption Check: Compressed Upper Flange
#   4.10 SERVICE PHASE
#       4.10.1 Concrete Effective Width
#       4.10.2 Sectional Properties
#       4.10.3 Structural Analysis: ULS
#       4.10.4 Shear Capacity
#       4.10.5 Bending Moment Capacity
#       4.10.6 Verification
#       4.10.7 Assumption Check: Compressed Upper Flange
#   4.11 SHEAR STUDS
#   4.12 WELDS
#   4.13 SLS Sectional Constants
#   4.14 FATIGUE
#   4.15 DEFLECTION
# 5. CROSSBEAMS: Span
# 6. CROSSBEAMS: Ends
# 7. Minimum Reinforcement
# 8. Connections
# 9. Quantites
# 10. OBJECTIVE AND FITNESS FUNCTION
#   10.1 Objective Function
#   10.2 Penalty Function
#   10.3 Fitness Function
```

```
# 11. GENETIC ALGORITHM
# 12. RESULTS
#   12.1 Convergence Plots

#-------------------------------------------------------------------------------
#-------------------------------- 1. INPUT DATA --------------------------------
OptimizationTarget = 'LCC'  # 'Mass', 'LCC', 'LCA' or 'Invest'

#-----------------------------------------------------
# 1.1 Geometrical Input Data - not Subjected to the Optimization
#-----------------------------------------------------
L = 51e3                    # Length of the bridge [mm], 25 m <= L <= 75 m
SubDiv = 500                # Subdivison/mesh of the bridge [mm]
w = 3250                    # Width of the traffic lanes [mm]
Nw = 2                      # Number of traffic lanes [-]
PC1 = 3e3                   # Length of pedestrian/cyclist lane, left side
PC2 = 500                   # and right side. Note! Minimum PC = 500 [mm]
a = 5                       # Weld throat thickness [mm]
b0 = 330.                   # Distance between shear studs [mm]

#-----------------------------------------------------
# 1.2 Width of the Bridge ------------------------------
#-----------------------------------------------------
B = w*Nw+PC1+PC2            # Total width of the bridge deck [mm]
Bs = 5600                   # C-C distance between steel giders [mm]

#-----------------------------------------------------
# 1.3 Number of Longitudinal Bridge Segments and Splices
#-----------------------------------------------------
if L >=25.0e3 and L <= 40.0e3:
    Nseg = 3
    Nsp = 2
if L > 40.0e3 and L <= 60.0e3:
    Nseg = 5
    Nsp = 2
if L > 60.0e3 and L <= 75.0e3:
    Nseg = 7
    Nsp = 3

#-----------------------------------------------------
# 1.4 Material Input -----------------------------------
#-----------------------------------------------------
SteelGrade = 'S355'                 # Steel grade of girders
Shape = 'Flat'                      # Web shape, 'Flat' or 'Corrugated'
ConcreteGrade = 'C35_45'            # Concrete grade of deck
PavementType = 'AsphaltConcrete'    # Pavement type
hp = 50.                            # Height of pavement cladding [mm]
ReinforcementGrade = 'B500B'        # Reinforcement grade
CementClass = 'S'                   # Cement class, 'S' or 'N'
d_re = 20.                          # Reinforcement diameter [mm]
d_stud = 22.                        # Diameter of shear stud, 16 to 25 [mm]
h_stud = 200.                       # Height of shear stud

#-----------------------------------------------------
# 1.5 Environmental ------------------------------------
#-----------------------------------------------------
RH = 80.                    # Relative humidity [%]
T_max = 35.                 # Maximum ambient temperature [degrees C]
T_min = -36.                # Minimum ambient temperature [degrees C]
vb = 23.                    # Reference wind velocity [m/s]
qp = 0.75                   # Characteristic velocitypressure [kN/m^2]
ADT = 5e3                   # Average Daily Traffic

#-----------------------------------------------------
# 1.6 Construction -------------------------------------
#-----------------------------------------------------
Delta_T_cs = 15.            # Temperature difference between parts [degrees C]
T0 = 10.                    # Initial bridge temperature [degrees C]
ts = 1.                     # Age of concrete at beginning of drying [days]
t0 = 7.                     # Age of concrete at loading [days]
```

```
t0_cs = 1.                  # Age of concrete at loading, shrinkage [days]
SafetyClass = 3             # Safety Factor

#-----------------------------------------------------
# 1.7 Fatigue ------------------------------------------
#-----------------------------------------------------
ServiceLife = 120           # Design service life [years]
Method = 'SafeLife'         # Fatigue method
Consequence = 'High'        # Fatigue consequence
Nobs = 0.05*10**6           # Table 4.5 in SS-EN 1991-2
DetailCategoryA = 80        # Detail Category Mode A [MPa]
DetailCategoryB1 = 100      # Detail Category Mode B1 [MPa]
DetailCategoryB2 = 100      # Detail Category Mode B2 [MPa]
DetailCategoryC = 80        # Detail Category Mode C [MPa]
DetailCategoryD = 80        # Detail Category Mode D [MPa]
DetailCategoryE = 100       # Detail Category Mode E [MPa]
DetailCategoryF = 112       # Detail Category Mode F [MPa]

#-----------------------------------------------------
# 1.8 Imported Modules ---------------------------------
#-----------------------------------------------------
# Developed by this thesis
import MaterialClass as MTRL
import GeometricalClassFunctions as GEO
import LoadFunctions as LF
import StructuralAnalysisFunctions as SAF
import DesignFunctions as DF
import GetResults as GR

# Developed by parallell thesis
import LCC
import LCA

# Shared on PyPI
import numpy as np
import pandas as pd
import math
from geneticalgorithm import geneticalgorithm as ga
import matplotlib.pyplot as plt

#-------------------------------------------------------------------------------
#------------------------------- 2. CONCRETE DESIGN ----------------------------

#-----------------------------------------------------
# 2.1 Material Parameters ------------------------------
#-----------------------------------------------------
MtrlPar_c = MTRL.Concrete(ConcreteGrade)
MtrlPar_p = MTRL.Pavement(PavementType)
MtrlPar_re = MTRL.Reinforcement(ReinforcementGrade, d_re)

#-----------------------------------------------------
# 2.2 Height of the Concrete Deck [mm] -----------------
#-----------------------------------------------------
hc = 320.                   # Average height from Case Study [mm]

#-----------------------------------------------------
# 2.3 Modular Ratio ------------------------------------
#-----------------------------------------------------
Ac = B*hc/2                 # Area half bridge [mm^2]
u = (2*hc+2*B)/2            # Length exposed to drying, half bridge [mm]

# Loading Elastic Modulus of steel: OBS! thickness not relevant here.
MtrlPar_s = MTRL.Steel(SteelGrade, 20.)

# Modular ratios, short term, long term and shrinkage:
nL_short = DF.ModularRatio('ShortTerm', MtrlPar_c, MtrlPar_s, RH, Ac, u, t0)
nL_long = DF.ModularRatio('LongTerm', MtrlPar_c, MtrlPar_s, RH, Ac, u, t0)
nL_cs = DF.ModularRatio('Shrinkage', MtrlPar_c, MtrlPar_s, RH, Ac, u, t0_cs)

#-----------------------------------------------------
```

```
# 2.4 Shrinkage Force ----------------------------------
#-------------------------------------------------------
Fcs = LF.ShrinkageLoad(CementClass, MtrlPar_s, MtrlPar_c, Ac,u, RH, nL_cs)

#-------------------------------------------------------
# 2.5 Shear Stud Capacity ------------------------------
#-------------------------------------------------------
PRd = DF.ShearStuds(MtrlPar_s, MtrlPar_c, d_stud, h_stud)

#-------------------------------------------------------------------------------
#------------------------ 3. OPTIMIZATION INPUT --------------------------------

#-------------------------------------------------------
# 3.1 Design Domains -----------------------------------
#-------------------------------------------------------
# Plate thickness (flanges)
domain_tf = np.array([16, 18, 20, 25, 28, 30, 35, 40, 45, 50, 55, 60],
                     dtype=np.float64)

# Plate thickness (web)
domain_tw = np.array([4, 6, 8, 10, 12, 14, 16, 18, 20, 25, 28, 30],
                     dtype=np.float64)

# Plate width (flanges) [mm]
bmin = 400
bmax = 1500
bstep = 50
domain_b = np.arange(bmin, bmax+bstep, bstep, dtype=np.float64)

# Plate height (web) [mm]
hmin = 1000
hmax = 2000
hstep = 10
domain_h = np.arange(hmin, hmax+hstep, hstep, dtype=np.float64)

# Corrugation length (a1, a3) [mm]
amin = 50
amax = 400
astep = 10
domain_aCorr = np.arange(amin, amax+astep, astep, dtype=np.float64)

# Corrugation angle (alpha) [degrees]
alphamin = 30
alphamax = 60
alphastep = 1
domain_alphaCorr = np.arange(alphamin, alphamax+alphastep, alphastep,
                             dtype=np.float64)

# Crossbeams C-C distance
Cmin = 2000
Cmax = 10e3
Cstep = 50
domain_Ccb = np.arange(Cmin, Cmax+Cstep, Cstep, dtype=np.float64)
for j in range (0,len(domain_Ccb)):
    if L/domain_Ccb[j] != round(L/domain_Ccb[j]):
        domain_Ccb[j] = 0
domain_Ccb = domain_Ccb[domain_Ccb != 0]

# Distance to change section
mmin = -L/4/Nseg
mmax = L/4/Nseg
mstep = 500
domain_m = np.arange(mmin, mmax, mstep, dtype=np.float64)

#-------------------------------------------------------
# 3.2 Design Function ----------------------------------
#-------------------------------------------------------
def f(X):
    #---------------------------------------------------------------------------
    # -------------------------- 4. MAIN GIRDERS ----------------------------
```

```
    #-------------------------------------------------------
    # 4.1 Design Variables ---------------------------------
    #-------------------------------------------------------
    X = X.astype(int)
    hw = domain_h[(X[0])]            # Height of web [mm]
    tw = domain_tw[(X[1])]           # Tickness of web [mm]
    a1 = domain_aCorr[(X[2])]        # Corrugation flat fold lenght [mm]
    a3 = domain_aCorr[(X[3])]        # Corrugation depth [mm]
    alpha = domain_alphaCorr[X[4]]   # Corrugation angle [mm]
    Ccb = domain_Ccb[X[5]]           # C-C distance of crossbeams [mm]
    m = np.empty(Nseg-1)             # Distances to move each segment change
    for i in range (0, Nseg-1):
        m[i] = domain_m[X[6+i]]
    bfo_v = np.empty(Nseg)       # Upper flange width per segment [mm]
    bfu_v = np.empty(Nseg)       # Lower flange width per segment [mm]
    for i in range (0, Nseg):
        bfo_v[i] = domain_b[X[6+(Nseg-1)]]
        bfu_v[i] = domain_b[X[7+(Nseg-1)]]
    tfo_v = np.empty(Nseg)       # Upper flange thickness per segment [mm]
    tfu_v = np.empty(Nseg)       # Lower flange thickness per segment [mm]
    for i in range (0, Nseg):
        tfo_v[i] = domain_tf[X[8+(Nseg-1)+2*i]]
        tfu_v[i] = domain_tf[X[9+(Nseg-1)+2*i]]

    #-------------------------------------------------------
    # 4.2 Constraints --------------------------------------
    #-------------------------------------------------------
    ConstrainUR = np.zeros((19,Nseg))
# Utilization rate constraints. Plots how far from 1 each segment is.
# Each column represents segment i with i = 0...(Nseg-1), apart from SLS that
# is plotted in column 0.
# Each row represents each constraint in the following order:
    # --- ULS, Construction Phase:
    # 0. Shear
    # 1. Bending moment
    # 2. Interaction shear/moment
    # --- ULS, Service Phase:
    # 3. Shear
    # 4. Bending stresses in steel
    # 5. Bending stresses in concrete (compressive)
    # 6. Bending stresses in reinforcement (tensile)
    # 7. Interaction shear/moment
    # --- ULS, Welds:
    # 8. i1
    # 9. 2
    # 10. i2
    # --- FAT:
    # 11. Detail A
    # 12. Detail B1
    # 13. Detail B2
    # 14. Detail C
    # 15. Detail D
    # 16. Detail E
    # 17. Detail F
    # --- SLS:
    # 18. Deflection
    #       18.0 Max deflection (mid)

    ConstrainOther = np.zeros((11,Nseg))
# Other constraints that needs to be fulfilled due to computational and
# functional reasons.
# Each column is one segment i with i = 0...(Nseg-1), apart from the
# last three rows.
# Each row represents each constraint in the following order:
    # --- Segment x-coordinates:
    # 0. Interfearance of segment start x-coord. with crossbeam x-coord.
    # 1. Interfearance of segment end x-coord. with crossbeam x-coord.
    # --- Construction Phase:
    # 2. Convergence criteria for sectional parameter calculations
    # 3. Check of assumption that upper flange is in compression
```

```
# --- Service Phase:
# 4. Convergence critera for sectional parameter calculations (long term)
# 5. Convergence critera for sectional parameter calculations (short term)
# 6. Convergence critera for sectional parameter calculations (shrinkage)
# 7. Check of assumption that concrete and upper flange is in compression
# --- Crossbeams:
# 8. Crossbeams utilization criteria fulfilled by allowed selection:
#       8.0 Crossbeams in span (HEA section selected from sheet)
#       8.1 Thickness of end flanges
#       8.2 Thickness of end webs
# --- Segment change:
# 9. Ratio criteria of flange thickness between segments:
#       9.0 Sum of ratios for upper flanges
#       10.1 Sum of ratios for lower flanges
# 10. Ratio criteria of flange width between segments:
#       10.0 Sum of ratios for upper flanges
#       10.1 Sum of ratios for lower flanges

#-------------------------------------------------------
# 4.3 Corrugation Parameters [class] -------------------
#-------------------------------------------------------
CorrPar = GEO.CorrugationParameters(Shape, a1, a3, alpha)

#-------------------------------------------------------
# 4.4 Segment Coordinates and Positions [mm], [mm], [-]
#-------------------------------------------------------
[xstart, xend, SegPos]= GEO.SegCoord(L,Nseg,m)

# Penalty if joint interferes with crossbeam coordinates
numCB = int(L/Ccb)+1
CBcoord = np.empty(numCB)
for i in range (0,numCB):
    CBcoord[i] = Ccb*i

for i in range(0, len(xstart)):
    for k in range (0,numCB):
        if xstart[i] == CBcoord[k] and xstart[i] != 0:
            ConstrainOther[0,i] = 1.
        if xend[i] == CBcoord[k] and xend[len(xend)-1] != L/2:
            ConstrainOther[1,i] = 1.

#-------------------------------------------------------
# 4.5 Material Parameters (web) ------------------------
#-------------------------------------------------------
MtrlPar_w = MTRL.Steel(SteelGrade, tw)

#-------------------------------------------------------
# 4.6 Loads (mean value) -------------------------------
#-------------------------------------------------------
# Selfweight [N/mm]
bfo_mean = np.mean(bfo_v)
tfo_mean = np.mean(tfo_v)
bfu_mean = np.mean(bfu_v)
tfu_mean = np.mean(tfu_v)
[G_sgirders, G_concrete, G_unconcrete, G_pave, G_cb, G_fw] = \
LF.SelfWeight(Shape, MtrlPar_s, MtrlPar_s, MtrlPar_s, MtrlPar_c, MtrlPar_p,
              L, B, hw, tw, bfo_mean, tfo_mean, bfu_mean, tfu_mean, a,
              CorrPar, hc, hp)
GSelfWeight_con = G_sgirders+G_unconcrete+G_cb+G_fw
GSelfWeight_ser = G_sgirders+G_concrete+G_cb

# Temperature [N]
As_mean = bfo_mean*tfo_mean+hw*tw+bfu_mean*tfu_mean
[FTemp_t,  FTemp_c] = LF.TemperatureLoad(MtrlPar_w, MtrlPar_c, Delta_T_cs,
                                         T_max, T_min, T0, As_mean)

#-------------------------------------------------------
# 4.7 Vectors for SLS Calculations (average value) -----
#-------------------------------------------------------
I_SLS = np.empty((Nseg,3))          # Moment of inertia (SLS)
```

```
tp_SLS = np.empty((Nseg,3))          # Neutral axis (SLS)
tp_s_SLS = np.empty((Nseg,3))        # Neutral axis, steel section (SLS)

#-----------------------------------------------------
# 4.8 Design Procedure per Segment ---------------------
#-----------------------------------------------------
nStuds = np.empty(Nseg)              # Collection of studs
ccStuds = np.empty(Nseg)             # Collection of C-C distance of studs

for i in range (0,Nseg):
    bfo = bfo_v[i]
    tfo = tfo_v[i]
    bfu = bfu_v[i]
    tfu = tfu_v[i]

    #-------------------------------------------------
    # 4.8.1 Material Parameters (flanges) -------------
    #-------------------------------------------------
    MtrlPar_fo = MTRL.Steel(SteelGrade, tfo)
    MtrlPar_fu = MTRL.Steel(SteelGrade, tfu)

    #-------------------------------------------------
    # 4.8.2 Cross-Section Class -----------------------
    #-------------------------------------------------
    [CSC_fo, CSC_w] = DF.CrossSectionClass(MtrlPar_w, MtrlPar_fo, Shape,
                                           CorrPar, hw, tw, bfo, tfo, a)

    #-------------------------------------------------------------------------
    # ---------------------- 4.9 CONSTRUCTION PHASE ----------------------

    #-------------------------------------------------
    # 4.9.1 Sectional Properties ----------------------
    #-------------------------------------------------
    tpw = hw/2                  # Neutral axis from top of web [mm],
    e = 1                       # Initial guess of tpw and initial error e
    j = 0
    while e > 1e-5:
        [psi_w, psi_fo] = SAF.CSC4psi(CSC_w, CSC_fo, hw, tpw)
        [hwo_eff, bfo_eff] = \
        DF.CSC4EffectiveWidth(MtrlPar_w, MtrlPar_fo, hw, tw, bfo, tfo, a,
                              CSC_w, CSC_fo, psi_w, psi_fo)
        SecPar = GEO.SectionalProperties(hw, hwo_eff, tw, bfo_eff, tfo,
                                         bfu, tfu, 0, 0, 1, Shape)
        tpw_new = SecPar.tp-tfo
        e = abs(tpw-tpw_new)
        tpw = tpw_new
        j = j+1
        if j == 30:
            ConstrainOther[2,i] = 1.
            break

    #-------------------------------------------------
    # 4.9.2 Structural Analysis - ULS -----------------
    #-------------------------------------------------
    H = hw+tfo+tfu+hc+hp     # Total height of bridge [mm]

    # Units: [MPa], [MPa], [MPa], [N], [N/mm], [Nmm].
    # OBS! no nL -> same SecPar for all loads
    [sigmaEd_ct, sigmaEd_cc, sigmaEd_o, sigmaEd_u, VEd, VEd_studs, MEd] = \
    SAF.ULS(SafetyClass, GSelfWeight_con, G_pave, 0, 0, 0, 'no', L, SubDiv,
            xstart[i], xend[i], B, Bs, H, 1, SecPar, SecPar, SecPar)
    if i == Nseg-1:
        MEd_con = MEd

    #-------------------------------------------------
    # 4.9.3 Shear Capacity ----------------------------
    #-------------------------------------------------
    chi_w = DF.ShearBuckling(MtrlPar_w, Shape, hw, tw, CorrPar, Ccb)
    [VRd, VRd_w] = DF.ShearResistance(Shape, MtrlPar_w, MtrlPar_fo,
                                      MtrlPar_fu, hw, tw, bfo_eff, tfo,
```

```
                                    bfu, tfu, chi_w, MEd, Ccb, SecPar.tp)

#-------------------------------------------------
# 4.9.4 Bending Moment Capacity --------------------
#-------------------------------------------------
chi_LT = DF.LTBuckling(MtrlPar_fo, Ccb, bfo, bfo_eff, tfo, hw, tfu)
fyd = DF.SteelMomentCapacity(MtrlPar_fu, SecPar, Shape, bfo_eff,
                             tfo, bfu, tfu, hw, chi_LT)

#-------------------------------------------------
# 4.9.5 Verification -------------------------------
#-------------------------------------------------
# Shear
URshear = VEd/VRd
if URshear > 1:
    ConstrainUR[0,i] = (URshear-1)

# Bending moment
sigma_M = max(abs(sigmaEd_o), abs(sigmaEd_u))
URmoment = sigma_M/fyd
if URmoment > 1:
    ConstrainUR[1,i] = (URmoment-1)

# Interaction
URint = DF.Interaction(Shape, MtrlPar_w, VEd, VRd, VRd_w, MEd, hw, tw,
                       bfo_eff, tfo, bfu, tfu, 0, 0, 1)
if URint > 1:
    ConstrainUR[2,i] = (URint-1)

#-------------------------------------------------
# 4.9.6 Assumption Check: Compressed Upper Flange --
#-------------------------------------------------
if SecPar.tp < tfo:
    ConstrainOther[3,i] = 1.

#-------------------------------------------------------------------------
#------------------------- 4.10 SERVICE PHASE ----------------------------

#-------------------------------------------------
# 4.10.1 Concrete Effective Width ------------------
#-------------------------------------------------
Be = (B-Bs)/2
bc_eff = DF.ShearLagConcrete(b0, Bs, Be, Be, L, SegPos[i])

#-------------------------------------------------
# 4.10.2 Sectional Properties ----------------------
#-------------------------------------------------
# Long term load
tpw = hw/2              # Neutral axis from top - initial guess [mm]
e = 1                   # Error
j = 0
while e > 1e-5:
    [psi_w, psi_fo] = SAF.CSC4psi(CSC_w,CSC_fo,hw,tpw)
    [hwo_eff, bfo_eff] = \
    DF.CSC4EffectiveWidth(MtrlPar_w, MtrlPar_fo, hw, tw, bfo, tfo, a,
                          CSC_w, CSC_fo, psi_w, psi_fo)
    SecPar_long = \
    GEO.SectionalProperties(hw, hwo_eff, tw, bfo_eff, tfo, bfu, tfu,
                            hc, bc_eff, nL_long, Shape)
    tpw_new = SecPar_long.tp-hc-tfo
    e = abs(tpw-tpw_new)
    tpw = tpw_new
    j = j+1
    if j == 30:
        ConstrainOther[4,i] = 1.
        break

# Short term load
e = 1                   # Error
j = 0
```

```
while e > 1e-5:
    [psi_w, psi_fo] = SAF.CSC4psi(CSC_w,CSC_fo,hw,tpw)
    [hwo_eff, bfo_eff] = \
    DF.CSC4EffectiveWidth(MtrlPar_w, MtrlPar_fo, hw, tw, bfo, tfo, a,
                          CSC_w, CSC_fo, psi_w, psi_fo)
    SecPar_short = \
    GEO.SectionalProperties(hw, hwo_eff, tw, bfo_eff,tfo, bfu, tfu,
                            hc, bc_eff, nL_short, Shape)
    tpw_new = SecPar_short.tp-hc-tfo
    e = abs(tpw-tpw_new)
    tpw = tpw_new
    j = j+1
    if j == 30:
        ConstrainOther[5,i] = 1.
        break

# Shrinkage
e = 1                    # Error
j = 0
while e > 1e-5:
    [psi_w, psi_fo] = SAF.CSC4psi(CSC_w,CSC_fo,hw,tpw)
    [hwo_eff, bfo_eff] = \
    DF.CSC4EffectiveWidth(MtrlPar_w, MtrlPar_fo, hw, tw, bfo, tfo, a,
                          CSC_w, CSC_fo, psi_w, psi_fo)
    SecPar_cs = \
    GEO.SectionalProperties(hw, hwo_eff, tw, bfo_eff, tfo, bfu, tfu,
                            hc, bc_eff, nL_cs, Shape)
    tpw_new = SecPar_cs.tp-hc-tfo
    e = abs(tpw-tpw_new)
    tpw = tpw_new
    j = j+1
    if j == 30:
        ConstrainOther[6,i] = 1.
        break

#-------------------------------------------------
# 4.10.3 Structural Analysis: ULS ----------------
#-------------------------------------------------
H = hw+tfo+tfu+hc+hp     # Total height of bridge [mm]

# Units: [MPa], [MPa], [MPa], [N], [N/mm], [Nmm]
[sigmaEd_ct, sigmaEd_cc, sigmaEd_o, sigmaEd_u,
 VEd, VEd_studs, MEd] = SAF.ULS(SafetyClass, GSelfWeight_ser, G_pave,
                                Fcs, FTemp_t, FTemp_c, 'yes',
                                L, SubDiv, xstart[i], xend[i],
                                B, Bs, H, Ac,
                                SecPar_long, SecPar_short, SecPar_cs)

#-------------------------------------------------
# 4.10.4 Shear Capacity --------------------------
#-------------------------------------------------
# Neutral axis from top of steel [mm]:
tp_s = SecPar_long.tp_s-hc

# OBS! chi_w same as for construction phase
[VRd, VRd_w] = DF.ShearResistance(Shape, MtrlPar_w, MtrlPar_fo,
                                  MtrlPar_fu, hw, tw, bfo_eff, tfo,
                                  bfu, tfu, chi_w, MEd, Ccb, tp_s)

#-------------------------------------------------
# 4.10.5 Bending Moment Capacity and Interaction ---
#-------------------------------------------------
[fyd, fcd, fctd, fsd] = DF.CompositeMomentCapacity(MtrlPar_c,
                                                   MtrlPar_s,
                                                   MtrlPar_re)

#-------------------------------------------------
# 4.10.6 Verification ----------------------------
#-------------------------------------------------
# Shear
```

```
        URshear = VEd/VRd
        if URshear > 1:
            ConstrainUR[3,i] = (URshear-1)

        # Bending
        sigma_M = max(abs(sigmaEd_o), abs(sigmaEd_u))
        URmoment_s = sigma_M/fyd
        if URmoment_s > 1:
            ConstrainUR[4,i] = (URmoment_s-1)

        URmoment_cc = abs(sigmaEd_cc)/fcd
        if URmoment_cc > 1:
            ConstrainUR[5,i] = (URmoment_cc-1)

        if sigmaEd_ct <= 0:
            URmoment_ct = 0
        else:
            URmoment_ct = sigmaEd_ct/fsd
        if URmoment_ct > 1:
            ConstrainUR[6,i] = (URmoment_ct-1)

        # Interaction
        URint = DF.Interaction(Shape, MtrlPar_w, VEd, VRd, VRd_w, MEd,
                               hw, tw, bfo_eff, tfo, bfu, tfu,
                               hc, bc_eff, nL_long)
        if URint > 1:
            ConstrainUR[7,i] = (URint-1)

        #-------------------------------------------------
        # 4.10.7 Assumption Check: Compressed Upper Flange -
        #-------------------------------------------------
        if SecPar_long.tp < hc+tfo or SecPar_short.tp < hc+tfo \
        or SecPar_cs.tp < hc+tfo:
            ConstrainOther[7,i] = 1

        #-------------------------------------------------------------------------
        # ------------------------ 4.11 SHEAR STUDS --------------------------
        Lseg = xend[i]-xstart[i]
        nStuds[i] = VEd_studs*Lseg/PRd
        nStuds[i] = int(nStuds[i])+1
        ccStuds[i] = Lseg/(nStuds[i]/2)

        #-------------------------------------------------------------------------
        # --------------------------- 4.12 WELDS -----------------------------
        # Design stresses [MPa]:
        (TauPar1, sigmai1, SigmaPer2, sigmai2) = \
        SAF.ULS_welds(G_cb, G_concrete, G_pave, hc, hp, a, tw, SecPar_short,
                      VEd, SegPos[i])

        # Design resistance [MPa]:
        (sigma_perp, sigma_i) = DF.WeldResistance(MtrlPar_s)

        # Verification:
        URweldi1 = sigmai1 / sigma_i
        if URweldi1 > 1:
            ConstrainUR[8,i] = (URweldi1-1)

        URweld2 = SigmaPer2 / sigma_perp
        if URweld2 > 1:
            ConstrainUR[9,i] = (URweld2-1)

        URweldi2 = sigmai2 / sigma_i
        if URweldi2 > 1:
            ConstrainUR[10,i] = (URweldi2-1)

    #-------------------------------------------------------------------------
    # ------------------- 4.13 SLS Sectional Constants ----------------------
        # Save values from all section iterations to calculate mean value:
        bfo_eff = DF.ShearLagSteel(bfo, L, SegPos[i])
        SecPar_long = GEO.SectionalProperties(hw, 0, tw, bfo_eff, tfo, bfu,
```

```
                                                tfu, hc, bc_eff, nL_long, Shape)
        SecPar_short = GEO.SectionalProperties(hw, 0, tw, bfo_eff, tfo, bfu,
                                                tfu, hc, bc_eff, nL_short,
                                                Shape)
        SecPar_cs = GEO.SectionalProperties(hw, 0, tw, bfo_eff, tfo, bfu, tfu,
                                            hc, bc_eff, nL_cs, Shape)
        I_SLS[i,:] = [SecPar_long.I, SecPar_short.I, SecPar_cs.I]
        tp_SLS[i,:] = [SecPar_long.tp, SecPar_short.tp, SecPar_cs.tp]
        tp_s_SLS[i,:] = [SecPar_long.tp_s, SecPar_short.tp_s, SecPar_cs.tp_s]

    #-------------------------------------------------------------------------
    # ------------------------------ 4.14 FATIGUE -----------------------------
        # Fatigue load:
        (VEd_FAT, MEd_FAT) = SAF.FAT(L, SubDiv, xstart[i], xend[i])

        # Lambda-method:
        (lambd_span_s, lambda_span_m, lambd_support, lambda_2, lambda_3,
         lambda_4, lambda_1_s, lambda_1_m) = SAF.LambdaMethod(L, Nobs,
                                                              ServiceLife)

        # Design Stress in Details:
        (A, B1, B2, C, D, E, F) = SAF.Details_FAT(lambd_span_s, lambda_span_m,
                                                  lambd_support, SecPar_short,
                                                  a, VEd_FAT, MEd_FAT,
                                                  tw, hw, tfo, tfu, hc,
                                                  SegPos[i])

        # Design Resistance of Details:
        # Mode A
        ks = 1.0                            # [-]
        FatigueResistanceA = DF.Fatigue(DetailCategoryA, ks, Method,
                                        Consequence)

        # Mode B1
        ks = 1.0                            # [-]
        FatigueResistanceB1 = DF.Fatigue(DetailCategoryB1, ks, Method,
                                         Consequence)

        # Mode B2
        ks = 1.0                            # [-]
        FatigueResistanceB2 = DF.Fatigue(DetailCategoryB2, ks, Method,
                                         Consequence)

        # Mode C
        ks = 1.0                            # [-]
        FatigueResistanceC = DF.Fatigue(DetailCategoryC, ks, Method,
                                        Consequence)

        # Mode D
        ks = 1.0                            # [-]
        FatigueResistanceD = DF.Fatigue(DetailCategoryD, ks, Method,
                                        Consequence)

        # Mode E
        ks = 1.0                            # [-]
        FatigueResistanceE = DF.Fatigue(DetailCategoryE, ks, Method,
                                        Consequence)

        # Mode F
        if tfu > 25:
            ksF = (25/tfu)**0.2             # [-]
        if tfu <= 25:
            ksF = 1.0
        FatigueResistanceF = DF.Fatigue(DetailCategoryF, ksF, Method,
                                        Consequence)

        # Verification:
        # MODE A:
        URfatA = A / FatigueResistanceA
        if URfatA > 1:
```

```
            ConstrainUR[11,i] = (URfatA-1)

        # MODE B1:
        URfatB1 = B1 / FatigueResistanceB1
        if URfatB1 > 1:
            ConstrainUR[12,i] = (URfatB1-1)

        # MODE B2:
        URfatB2 = B2 / FatigueResistanceB2
        if URfatB2 > 1:
            ConstrainUR[13,i] = (URfatB2-1)

        # MODE C:
        URfatC = C / FatigueResistanceC
        if URfatC > 1:
            ConstrainUR[14,i] = (URfatC-1)

        # MODE D:
        URfatD = D / FatigueResistanceD
        if URfatD > 1:
            ConstrainUR[15,i] = (URfatD-1)

        # MODE E:
        URfatE = E / FatigueResistanceE
        if URfatE > 1:
            ConstrainUR[16,i] = (URfatE-1)

        # MODE F:
        URfatF = F / FatigueResistanceF
        if URfatF > 1:
            ConstrainUR[17,i] = (URfatF-1)

#-------------------------------------------------------------------------
# ----------------------------- 4.15 DEFLECTION ---------------------------
# Mean value of Moment of Inertia
I_long = np.mean(I_SLS[:,0])
I_short = np.mean(I_SLS[:,1])
I_cs = np.mean(I_SLS[:,2])

# Mean values of the neutral axis
tp_long = np.mean(tp_SLS[:,0])
tp_short = np.mean(tp_SLS[:,1])
tp_cs = np.mean(tp_SLS[:,2])
tp_s_short = np.mean(tp_s_SLS[:,1])
tp_c = hc/2

# Sectional modulus:
Wo_short = I_short / -tp_short
Wu_short = I_short/(hc+tfo+hw+tfu-tp_short)
Wo_long = I_long / -tp_long
Wu_long = I_long/(hc+tfo+hw+tfu-tp_long)
Wo_cs = I_cs / -tp_cs
Wu_cs = I_cs/(hc+tfo+hw+tfu-tp_cs)

# Structural analysis, SLS, maximum deflection [mm]
(defTra, defPerm) = SAF.SLS(MtrlPar_s, GSelfWeight_ser, G_pave, Fcs,
                            FTemp_t, FTemp_c,
                            'yes', L, PC1, I_long, I_short, I_cs,
                            SubDiv, B, Bs, Ac,
                            tp_cs, tp_c, tp_short, tp_s_short,
                            Wo_long, Wu_long, Wo_cs,
                            Wu_cs, Wo_short, Wu_short)

# Verification, SLS
UR_SLS = defTra/(L/400)
if UR_SLS > 1:
    ConstrainUR[18,0] = (UR_SLS-1)

#-------------------------------------------------------------------------
# ----------------------- 5. CROSSBEAMS: Span ---------------------------
```

```
# Loads
MEd_max = MEd_con                           # Mid section moment [Nmm]
H = tfo_v[Nseg-1]+hw+tfu_v[Nseg-1]+hc+hp    # Mid section height [mm]
Fwind = LF.WindLoad(qp,B,H,'no')            # [N/mm]
[NEd_cb, MEd_cb] = SAF.ULS_horizontal(Fwind, MEd_max, L, hc, hw,
                                      tfo, tfu, Ccb)

# Design: beam or truss
DataHEA = pd.read_excel('BALK HEA.xls', sheet_name='HEA2python')
MtrlPar_cb = MTRL.Steel(SteelGrade, 16.5) # Material and maximum thickness
gammaM1 = 1.1
gammaM0 = 1.
Hs = tfo_v[Nseg-1]+hw+tfu_v[Nseg-1]
if Hs > 2000:
    CrossBeamType = 'Truss'
    URcb = 2
    i = -1
    while URcb > 1:
        A = DataHEA.values[i+1,2]
        h = DataHEA.values[i+1,4]
        b = DataHEA.values[i+1,5]
        Iz = DataHEA.values[i+1,12]
        chi = DF.Buckling(MtrlPar_cb, Bs, A, h, b, Iz)
        if chi == 1.:
            NRd_cb = MtrlPar_cb.fy*A/gammaM0
        else:
            NRd_cb = chi*MtrlPar_cb.fy*A/gammaM1
        URcb = NEd_cb/NRd_cb
        i = i+1
        if i == len(DataHEA):
            ConstrainOther[8,0] = 1
            break
    HEA_cb = DataHEA.values[i,0]
    weight_cb = DataHEA.values[i,1]*1e3   # [kg/mm]
    surfacearea_cb = DataHEA.values[i,22] # [mm^2/mm]
else:
    CrossBeamType = 'Beam'
    URcb = 2
    i = 0
    while URcb > 1:
        # Axial
        A = DataHEA.values[i,2]
        h = DataHEA.values[i,4]
        b = DataHEA.values[i,5]
        Iz = DataHEA.values[i,12]
        chi = DF.Buckling(MtrlPar_cb, Bs, A, h, b, Iz)
        if chi == 1.:
            NRd_cb = MtrlPar_cb.fy*A/gammaM0
        else:
            NRd_cb = chi*MtrlPar_cb.fy*A/gammaM1
        # Moment
        tf = DataHEA.values[i,6]
        Wely = DataHEA.values[i,9]
        hw_cb = h-2*tf
        chiLT_cb = DF.LTBuckling(MtrlPar_cb, Bs, b, b, tf, hw_cb, tf)
        if chiLT_cb == 1:
            MRd_cb = MtrlPar_cb.fy*Wely/gammaM0
        else:
            MRd_cb = chiLT_cb*MtrlPar_cb.fy*Wely/gammaM1
        URcb = NEd_cb/NRd_cb+MEd_cb/MRd_cb
        i = i+1
        if i == len(DataHEA):
            ConstrainOther[8,0] = 1
            break
    HEA_cb = DataHEA.values[i,0]
    weight_cb = DataHEA.values[i,1]*1e3   # [kg/mm]
    surfacearea_cb = DataHEA.values[i,22] # [mm^2/mm]

#-------------------------------------------------------------------------
# ------------------------ 6. CROSSBEAMS: Ends --------------------------
```

```
# Loads
# Safety factors:
if SafetyClass == 1:
   gamma_d= 0.83
if SafetyClass == 2:
   gamma_d=0.91
if SafetyClass == 3:
   gamma_d=1.0
else:
   gamma_d=1.1
sup_cc=1.1
gamma_G=1.35
TotSelfWeight = gamma_d*gamma_G*(np.mean(GSelfWeight_ser)+sup_cc*G_pave)*L
VEd_cbe = TotSelfWeight/4
MEd_cbe = VEd_cbe*800
# Assume support location 800 mm in from main girders connection

# Design
bf_cbe = 400.                # [mm]
hw_cbe = min(hw-200., 660.) # [mm],
                             # 100 mm distances top/bottom or max height

# Moment (design of flange thickness)
URmoment_cbe = 2
i = 0
while URmoment_cbe > 1:
    tf_cbe = domain_tf[i]

    # Material
    MtrlPar_fcbe = MTRL.Steel(SteelGrade, tf_cbe)

    # Cross-section class (flange)
    [CSC_fcbe, none] = DF.CrossSectionClass(MtrlPar_fcbe, MtrlPar_fcbe,
                                            'Flat', 1, hw_cbe, 10, bf_cbe,
                                            tf_cbe, a)

    # Only Cross-section class 3 or less
    if CSC_fcbe < 4:
        # Bending moment capacity
        chi_LT_cbe = DF.LTBuckling(MtrlPar_fcbe, Bs, bf_cbe, bf_cbe,
                                   tf_cbe, hw_cbe, tf_cbe)
        fyd_cbe = DF.SteelMomentCapacity(MtrlPar_fcbe, 0, 'Flat', bf_cbe,
                                         tf_cbe, bf_cbe, tf_cbe, hw_cbe,
                                         chi_LT_cbe)
        tp_cbe = (hw_cbe+2*tf_cbe)/2

        # Section modulus only considering flange contribution
        W_cbe = 2*(bf_cbe*tf_cbe**3/12+bf_cbe*tf_cbe*(tp_cbe-tf_cbe/2)**2)\
                /tp_cbe

        # Moment verification
        sigmaEd_cbe = MEd_cbe/W_cbe
        URmoment_cbe = sigmaEd_cbe/fyd_cbe
    else:
        URmoment_cbe = 2
    i = i+1
    if i == len(domain_tf):
        ConstrainOther[8,1] = 1
        break

# Shear (design of web thickness)
URshear_cbe = 2
URint_cbe = 2
i = 0
while URshear_cbe > 1 or URint_cbe > 1:
    tw_cbe = domain_tw[i]

    # Material
    MtrlPar_wcbe = MTRL.Steel(SteelGrade, tw_cbe)
```

```
        # Cross-section class (web)
        [none, CSC_wcbe] = DF.CrossSectionClass(MtrlPar_fcbe, MtrlPar_fcbe,
                                                'Flat', 1, hw_cbe, tw_cbe,
                                                bf_cbe, tf_cbe, a)

        # Only Cross-section class 3 or less
        if CSC_wcbe < 4:
            # Section properties
            SecPar_cbe = GEO.SectionalProperties(hw_cbe, 0, tw_cbe, bf_cbe,
                                                 tf_cbe, bf_cbe, tf_cbe,
                                                 0, 0, 1, 'Flat')

            # Shear capacity
            chi_w = DF.ShearBuckling(MtrlPar_wcbe, 'Flat', hw_cbe, tw_cbe,
                                     0, Bs)
            [VRd_cbe, VRd_w_cbe] = DF.ShearResistance('Flat',
                                                      MtrlPar_wcbe,
                                                      MtrlPar_fcbe,
                                                      MtrlPar_fcbe,
                                                      hw_cbe, tw_cbe, bf_cbe,
                                                      tf_cbe, bf_cbe, tf_cbe,
                                                      chi_w, MEd_cbe, Bs,
                                                      SecPar_cbe.tp)
            # Verification
            URshear_cbe = VEd_cbe/VRd_cbe
            URint_cbe = DF.Interaction('Flat', MtrlPar_wcbe, VEd_cbe, VRd_cbe,
                                       VRd_w_cbe, MEd_cbe, hw_cbe, tw_cbe,
                                       bf_cbe, tf_cbe, bf_cbe, tf_cbe, 0, 0, 1)
        else:
            URshear_cbe = 2
            URint_cbe = 2
        i = i+1
        if i == len(domain_tw):
            ConstrainOther[8,2] = 1
            break

#-------------------------------------------------------------------------
# ------------------------ 7. Minimum Reinforcement ------------------------
Ac_eff = hc * bc_eff
numRe = DF.MinReinforcement(Ac_eff, hc, MtrlPar_c, MtrlPar_re, tp_short,
                            tp_c, d_re)

#-------------------------------------------------------------------------
# ---------------------------- 8. Connections -----------------------------
if Shape == 'Flat':
    Lcon = L
if Shape == 'Corrugated':
    rc = (CorrPar.a1+CorrPar.a2)/(CorrPar.a1+CorrPar.a4)
    Lcon = L * rc
nStuds_tot = np.sum(nStuds)*2
Connection = GEO.Connection(SteelGrade, Lcon, Nseg, Nsp, numCB, a, hw, tw,
                            bfo_v, tfo_v, bfu_v, tfu_v, hw_cbe, xstart,
                            xend, CBcoord, 10, d_stud, nStuds_tot)

#-------------------------------------------------------------------------
# ----------------------------- 9. Quantities -----------------------------
# Web main girders
if Shape == 'Flat':
    weight_plate = 2*L*hw*tw*1e-9*MtrlPar_s.rho
    paint_area =  4*L*hw*1e-6
if Shape == 'Corrugated':
    rc = (CorrPar.a1+CorrPar.a2)/(CorrPar.a1+CorrPar.a4)
    weight_plate = 2*L*rc*hw*tw*1e-9*MtrlPar_s.rho
    paint_area = 4*L*rc*hw*1e-6

# Flanges
for i in range (0,Nseg):
    Lseg = xend[i]-xstart[i]
    weight_plate = weight_plate +\
                   4*Lseg*(bfo_v[i]*tfo_v[i]+bfu_v[i]*tfu_v[i])*1e-9*\
```

```
                    MtrlPar_s.rho
    paint_area = paint_area + 8*Lseg*(bfo_v[i]+bfu_v[i]+tfo_v[i]+tfu_v[i])\
                              *1e-6

# End crossbeams
weight_plate = weight_plate + 2*Bs*\
                (2*bf_cbe*tf_cbe+hw_cbe*tw_cbe)*1e-9*MtrlPar_s.rho
paint_area = paint_area + 4*Bs*(hw_cbe+2*bf_cbe+2*tf_cbe)*1e-6

# Vertical stiffeners (assume thickness 10 mm)
weight_plate = weight_plate + \
                2*(numCB+2)*hw*np.mean(bfo_v)*10*1e-9*MtrlPar_s.rho
paint_area = paint_area + 4*(numCB+2)*hw*np.mean(bfo_v)*1e-6

# Intermediate crossbeams
if CrossBeamType == 'Truss':
    Hs = H-hc-hp
    weight_hotrolled = (numCB-2)*(2*Bs+2*math.sqrt((Bs/2)**2+(Hs/2)**2))*\
                       weight_cb
    paint_area = (numCB-2)*(2*Bs+2*math.sqrt((Bs/2)**2+(Hs/2)**2))*\
                 surfacearea_cb*1e-6 + paint_area

if CrossBeamType == 'Beam':
    weight_hotrolled = (numCB-2)*Bs*weight_cb
    paint_area = (numCB-2)*Bs*surfacearea_cb*1e-6 + paint_area

# Concrete
volume_concrete = L*hc*B*1e-9

# Reinforcement
Asi = (d_re/2)**2*math.pi
weight_rebars = (numRe*L)*Asi*1e-9*MtrlPar_re.rho

# Weight of welds:
weight_weld = 0
for i in range(1,Nw):
        if Connection[i,0] == 'Fillet':
            weight_weld = weight_weld + \
                          Connection[i,2]**2*1e-6*Connection[i,3]*7800
        if Connection[i,0] == 'Butt':
            weight_weld = weight_weld +\
                          Connection[i,2]*5*(1+1/math.sqrt(3))*1e-6*\
                          Connection[i,3]*7800

Quantities = GEO.Quantities(SteelGrade, Shape, weight_plate,
                            weight_hotrolled, 0, weight_weld, paint_area,
                            volume_concrete, weight_rebars)

Painting = GEO.Painting(MtrlPar_s.SteelType, paint_area, Nsp, numCB, hw)

if Shape == 'Flat':
    LenCorr = 0
if Shape == 'Corrugated':
    LenCorr = L*rc/1e3

if MtrlPar_s.SteelType == 'Carbon':
    pickling_area = 0
    grinding_len = 8*L/1e3

if MtrlPar_s.SteelType == 'Stainless':
    pickling_area = paint_area
    grinding_len = 0

#-----------------------------------------------------------------------
# --------------- 10. OBJECTIVE AND FITNESS FUNCTION ---------------------

#-------------------------------------------------------
# 10.1 Objective Function  ----------------------------
#-------------------------------------------------------
if OptimizationTarget == 'Mass':
```

```
        objective = weight_plate+weight_hotrolled+weight_weld
        a_pen = 1e9

    if OptimizationTarget == 'LCC' or OptimizationTarget == 'Invest':
        objective = LCC.LCC(OptimizationTarget, Quantities, Connection,
                            Painting, LenCorr, ServiceLife, pickling_area,
                            grinding_len, Nsp+1, numCB-2, ADT)
        a_pen = 1e12

    if OptimizationTarget == 'LCA':
        objective = LCA.LCA(OptimizationTarget,Painting, Quantities,
                            Connection, ServiceLife)
        a_pen = 1e10

    #-----------------------------------------------------
    # 10.2 Penalty Function  ------------------------------
    #-----------------------------------------------------
    # Parameters
    a_pen = a_pen           # Slope
    b_pen = 1               # Linear or exponetial
    C_pen = a_pen           # Constant

    # Reduced matrices
    ConstrainUR_red = ConstrainUR[ConstrainUR != 0]
    penaltyUR = 0
    for i in range(0,len(ConstrainUR_red)):
        penaltyUR = penaltyUR + a_pen*ConstrainUR_red[i]**b_pen +\
                    ConstrainUR_red.size*C_pen

    ConstrainOther_red = ConstrainOther[ConstrainOther != 0]
    penaltyOther = 0
    for i in range(0,len(ConstrainOther_red)):
        penaltyOther = penaltyOther + ConstrainOther_red[i]*C_pen

    penalty = penaltyUR + penaltyOther

    #-----------------------------------------------------
    # 10.3 Fitness Function  ------------------------------
    #-----------------------------------------------------
    fitness = objective + penalty

    return(fitness)

#-------------------------------------------------------------------------------
# ------------------------ 11. GENETIC ALGORITHM -------------------------------

N = 6+(Nseg-1)+2+2*Nseg   # Number of design variables [-]

# Variable boundaries
varbound1 = np.array([[0,len(domain_h)-1],\
                      [0,len(domain_tw)-1],\
                      [0,len(domain_aCorr)-1],\
                      [0,len(domain_aCorr)-1],\
                      [0,len(domain_alphaCorr)-1],\
                      [0,len(domain_Ccb)-1]])

varbound2 = np.array([[0,len(domain_m)-1]]*(Nseg-1))

varbound3 = np.array([[0,len(domain_b)-1],[0,len(domain_b)-1]])

varbound4 = np.array([[0,len(domain_tf)-1],[0,len(domain_tf)-1]]*Nseg)

varbound = np.concatenate((varbound1,varbound2,varbound3,varbound4),axis=0)

# Algorithm parameters
algorithm_param = {'max_num_iteration': 500,\
                   'population_size': 150,\
                   'mutation_probability': 0.2,\
                   'elit_ratio': 0.3,\
                   'crossover_probability': 0.5,\
```

```
                    'parents_portion': 0.3,\
                    'crossover_type':'uniform',\
                    'max_iteration_without_improv':None}

# Assemble GA model
model = ga(function=f,\
            dimension=N,\
            variable_type='int',\
            variable_boundaries=varbound,\
            function_timeout=10.,\
            algorithm_parameters=algorithm_param
            )

# Run GA
model.run()

# Report
convergence=model.report
solution = model.output_dict

#%%-------------------------------------------------------------------------------
# -------------------------- 12. RESULTS -------------------------------------
FinalObjective = solution['function']
FinalDesignVector = solution['variable']

# GetResults includes the same calculations as Section 1 to 10 of this
# document, but returns more details than the function f(X) included here.
# Note! that if any changes in input data are made in this document (apart
# from SteelGrade, Shape and OptimizationTarget) these changes also need to
# be made in GetResults.

[fitness, objective, penalty,
 ConstrainUR, ConstrainOther, UR,
 hw, tw, bfo_v, tfo_v, bfu_v, tfu_v,
 CBcoord, xstart, xend, SteelWeight,
 nStuds, ccStuds]  = GR.GetResults(FinalDesignVector, SteelGrade, Shape,
                                   OptimizationTarget)

#-----------------------------------------------------------
# 12.1 Convergence Plots  -------------------------------
#-----------------------------------------------------------
# Settings
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['left'].set_position('center')
ax.spines['bottom'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

# Full plot
x = np.linspace(0,len(convergence)-1,len(convergence))
plt.plot(x,convergence, 'r')
plt.show()

# Plot of last n-itterations
n = 495
xn = np.linspace(len(convergence)-n,len(convergence)-1,n)
plt.plot(xn,convergence[len(convergence)-n:len(convergence)], 'r')
plt.show()
```

C.2 Module: MaterialClass

```
# Master's Thesis
#
# Design Optimization of Composite Road Bridges using Genetic Algorithms
# -Corrugated Web Stainless Steel Girders versus Flat Web Carbon Steel Girders
#
# Cecilia Hallgren & Vilma Johansson
# June 2022
#
# ------------------------------------------------------------------------------
# MODULE OF MATERIAL CLASSES
#
# Contents
#   1. Steel Material Parameters
#   2. Concrete Material Parameters
#   3. Pavement Material Parameters
#   4. Reinforcement Material Parameters
#
# ------------------------------------------------------------------------------
# 1. Steel Material Parameters
#
# Input:
#   SteelGrade  Material grade: 'Duplex', 'S355' or 'S460'      [object, -]
#   t           Thickness of plate                              [float, mm]
#
# Output:
#   MatPar      Material class [SteelType, Es, fy, fu, rho, nu, alpha_s]:
#               With:
#                   SteelType   'Stainless' or 'Carbon'         [object, -]
#                   Es          Elastic modulus                 [float, GPa]
#                   fy          Yield strength                  [float, MPa]
#                   fu          Ultimate strength               [float, MPa]
#                   rho         Steel density                   [float, kg/m^3]
#                   nu          Poisson's ratio                 [float, -]
#                   alpha_s     Thermal expansion coefficent    [float, -]
#
# ------------------------------------------------------------------------------

class Steel:
    def __init__(self, SteelGrade, t):

        # Duplex 1.4462 parameters (SS-EN 1993-1-4: 2006/A1:2015):
        if SteelGrade == 'Duplex':
            self.SteelType = 'Stainless'

            # Elastic modulus:
            self.Es = 200.

            if t <= 8:
                # Yield strength:
                self.fy = 530.

                # Ultimate strength:
                self.fu = 700.

            if t > 8 and t <= 13:
                # Yield strength:
                self.fy = 480.

                # Ultimate strength:
                self.fu = 680.

            if t > 13 and t <= 75:
                # Yield strength:
                self.fy = 450.

                # Ultimate strength:
                self.fu = 650.
```

```
        # Density [kg/m^3] SS-EN 10088-1:2014 Table E.2
        self.rho = 7800.

        # Poisson's ratio:
        self.nu = 0.3

        # Linear thermal expansion coefficient (Second draft EN 1993-1)
        self.alpha_s = 13.*10**-6

    # S355 N/NL parameters (SS-EN 1993-1-1 Table 3.1 EN 10025-3):
    if SteelGrade == 'S355':
        self.SteelType = 'Carbon'

        if t > 80.:
            raise Exception('Not supported, t is too large.')

        # Elastic modulus:
        self.Es = 210.

        # Yield strength:
        if t <= 40.:
            self.fy = 355.
        if t > 40. and t <= 80.:
            self.fy = 335.

        # Ultimate strength:
        if t <= 40.:
            self.fu = 490.
        if t > 40. and t <= 80.:
            self.fu = 470.

        # Density:
        self.rho = 7800.

        # Poisson's ratio:
        self.nu = 0.3

        # Linear thermal expansion coefficient (SS-EN 1991-1-5 Table C.1):
        self.alpha_s = 10.*10**-6

        # Correlation factor [-] SS-En 1993-1-8 Table 4.1:
        self.beta_w = 0.9

    # S460 N/NL parameters (SS-EN 1993-1-1 Table 3.1 EN 10025-3):
    if SteelGrade == 'S460':
        self.SteelType = 'Carbon'

        if t > 80.:
            raise Exception('Not supported, t is too large.')

        # Elastic modulus:
        self.Es = 210.

        # Yield strength:
        if t <= 40.:
            self.fy = 460.
        if t > 40. and t <= 80.:
            self.fy = 430.

        # Ultimate strength:
        self.fu = 540.

        # Density:
        self.rho = 7800.

        # Poisson's ratio:
        self.nu = 0.3

        # Linear thermal expansion coefficient (SS-EN 1991-1-5 Table C.1)
```

```
            self.alpha_s = 10.*10**-6

            # Correlation factor [-] SS-EN 1993-1-8 Table 4.1:
            self.beta_w = 1.0

        # Material types supported:
        if SteelGrade == 'Duplex' or SteelGrade == 'S355' or SteelGrade == '
    S460':
            return
        else:
            raise Exception('Material grade not supported.')

# ------------------------------------------------------------------------
# 2. Concrete Material Parameters
#
# Input:
#   ConcreteGrade   Material grade: 'C30_37', 'C35_45' or 'C40_50' [object, -]
#
# Output:
#   MatPar          Material class [Ecm, fck, fckcube, fcm, fctm, fctk_005,
#                                   fctk_095, fsk, rho_c, rho_unc, alpha_c,
#                                   alpha, beta]:
#   With:
#       Ecm         Elastic modulus                              [float, GPa]
#       fck         Characteristic compressive strength          [float, MPa]
#       fckcube     Characteristic cube compressive strength     [float, MPa]
#       fcm         Mean value of concrete compressive strength[float, MPa]
#       fctm        Mean value of tensile strength               [float, MPa]
#       fctk_005    Characteristic tensile strength 0.05%        [float, MPa]
#       fctk_095    Characteristic tensile strength 0.95%        [float, MPa]
#       fsk         Characteristic yield strength of
#                   reinforcement                                [float, MPa]
#       rho_c       Reinforced concrete density                  [float, kN/m^3]
#       rho_unc     Unhardened reinforced concrete density       [float, kN/m^3]
#       alpha_c     Linear thermal expansion coefficient         [float, 1/dC]
#       alpha       Coefficient used in concrete design          [float, -]
#       beta        Coefficient used in concrete design          [float, -]
# ------------------------------------------------------------------------

class Concrete:
    def __init__(self, ConcreteGrade):

        # Concrete grade C30/37 (SS-EN 1992-1-1 Table 3.1):
        if ConcreteGrade == 'C30_37':

            # Elastic modulus:
            self.Ecm = 33.

            # Characteristic compressive strength:
            self.fck = 30.

            # Characteristic cube compressive strength:
            self.fckcube = 37.

            # Mean value of concrete compressive strength:
            self.fcm = 38.

            # Mean value of tensile strength:
            self.fctm = 2.9

            # Characteristic tensile strength 0.05%:
            self.fctk_005 = 2.0

            # Characteristic tensile strength 0.95%:
            self.fctk_095 = 3.8

            # Characteristic yield strength of reinforcement:
            self.fsk = 500.

            # Reinforced concrete density [kN/m^3]
```

```
        self.rho_c = 25.

        # Unhardened reinforced concrete density [kN/m^3]
        self.rho_unc = 26.

        # Linear thermal expansion coefficient (SS-EN 1991-1-5 Table C.1)
        self.alpha_c = 10.*10**-6

        # Factors for concrete design:
        self.alpha = 0.81
        self.beta = 0.416

    # Concrete grade C35/45 (SS-EN 1992-1-1 Table 3.1):
    if ConcreteGrade == 'C35_45':

        # Elastic modulus:
        self.Ecm = 34

        # Characteristic compressive strength:
        self.fck = 35.

        # Characteristic cube compressive strength:
        self.fckcube = 45.

        # Mean value of concrete compressive strength:
        self.fcm = 43.

        # Mean value of tensile strength:
        self.fctm = 3.2

        # Characteristic tensile strength 0.05%:
        self.fctk_005 = 2.2

        # Characteristic tensile strength 0.95%:
        self.fctk_095 = 4.2

        # Characteristic yield strength of reinforcement:
        self.fsk = 500.

        # Reinforced concrete density [kN/m^3]
        self.rho_c = 25.

        # Unhardened reinforced concrete density [kN/m^3]
        self.rho_unc = 26.

        # Linear thermal expansion coefficient (SS-EN 1991-1-5 Table C.1)
        self.alpha_c = 10.*10**-6

        # Factors for concrete design:
        self.alpha = 0.81
        self.beta = 0.416

    # Concrete grade C40/50 (SS-EN 1992-1-1 Table 3.1):
    if ConcreteGrade == 'C40_50':

        # Elastic modulus:
        self.Ecm = 35

        # Characteristic compressive strength:
        self.fck = 40.

        # Characteristic cube compressive strength:
        self.fckcube = 50.

        # Mean value of concrete compressive strength:
        self.fcm = 48.

        # Mean value of tensile strength:
        self.fctm = 3.5
```

```
            # Characteristic tensile strength 0.05%:
            self.fctk_005 = 2.5

            # Characteristic tensile strength 0.95%:
            self.fctk_095 = 4.6

            # Characteristic yield strength of reinforcement:
            self.fsk = 500.

            # Reinforced concrete density [kN/m^3]
            self.rho_c = 25.

            # Unhardened reinforced concrete density [kN/m^3]
            self.rho_unc = 26.

            # Linear thermal expansion coefficient (SS-EN 1991-1-5 Table C.1)
            self.alpha_c = 10.*10**-6

            # Factors for concrete design:
            self.alpha = 0.81
            self.beta = 0.416

        # Material types supported:
        if ConcreteGrade == 'C30_37' or ConcreteGrade == 'C35_45' \
        or ConcreteGrade == 'C40_50':
            return
        else:
            raise Exception('Concrete grade not supported.')

# ------------------------------------------------------------------------------
# 3. Pavement Material Parameter
#
# Input:
#   PavementType    Material types: 'AsphaltConcrete', 'AsphaltCover',
#                    'Epoxi_Akrylat', 'MasticAsphalt' or 'ProofMat'
#
# Output:
#   MatPar          Material class [rho_p]:
#
#   With:
#       rho         Material density                    [float, kN/m^3]
# ------------------------------------------------------------------------------

class Pavement:
    def __init__(self, PavementType):

# (Krav Brobyggande TDOK 2016:0204, Section B.3.1.1)
        # Asphalt Concrete:
        if PavementType == 'AsphaltConcrete':
            # Density
            self.rho = 23.

        # Asphalt Cover:
        if PavementType == 'AsphaltCover':
            # Density
            self.rho = 22.

        # Epoxi and Akrylat:
        if PavementType == 'Epoxi_Akrylat':
            # Density
            self.rho = 22.

        # Mastic Asphalt:
        if PavementType == 'MasticAsphalt':
            # Density
            self.rho = 24.

        # Water proof Mat:
        if PavementType == 'ProofMat':
            # Density
```

```
            self.rho = 22.

        # Material types supported:
        if PavementType == 'AsphaltConcrete' or PavementType == 'AsphaltCover'\
        or PavementType == 'Epoxi_Akrylat' or PavementType=='MasticAsphalt' or\
        PavementType=='ProofMat':
            return
        else:
            raise Exception('Pavement type not supported.')

# -----------------------------------------------------------------------------
# 4. Reinforcement Material Parameters
#
# Input:
#   ReinforcGrade   Material grade: 'SS_260S', 'B500B' & 'Ks_600S'      [-]
#   Dim             Diameter of bar                                    [float, mm]
#
# Output:
#   MatPar          Material class [Es, fyk]:
#   With:
#       Es          Elastic modulus                                    [float, GPa]
#       fyk         Characteristic yield strength                      [float, MPa]
#       Dim         Diameter of the bar                                [float, mm]
#       rho         Density                                            [float, kN/m^3]
# -----------------------------------------------------------------------------

class Reinforcement:
    def __init__(self, ReinforceGrade, Dim):

        # Reinforcement grade 'Ss_260S':
        if ReinforceGrade == 'Ss_260S':

            # Elastic modulus:
            self.Es = 200.

            # Characteristic yield strength:
            self.fyk = 260.

            # Diameter of the bar
            self.Dim = Dim

            # Density
            self.rho = 7800.

        # Reinforcement grade 'B500B':
        if ReinforceGrade == 'B500B':

            # Elastic modulus:
            self.Es = 200.

            # Characteristic yield strength:
            self.fyk = 500.

            # Diameter of the bar
            self.Dim = Dim

            # Selfweight
            self.rho = 7800.

        # Reinforcement grade 'Ks_600S':
        if ReinforceGrade == 'Ks_600S':

            # Elastic modulus:
            self.Es = 200.

            # Characteristic yield strength:
            self.fyk = 600.

            # Diameter of the bar
            self.Dim=Dim
```

```
        # Selfweight
        self.rho = 7800.

    # Material types supported:
    if ReinforceGrade == 'Ss_260S' or ReinforceGrade == 'B500B' \
    or ReinforceGrade == 'Ks_600S':
        return
    else:
        raise Exception('Reinforcement grade not supported.')
```

C.3 Module: GeometricalClassFunctions

```
# Master's Thesis
#
# Design Optimization of Composite Road Bridges using Genetic Algorithms
# -Corrugated Web Stainless Steel Girders versus Flat Web Carbon Steel Girders
#
# Cecilia Hallgren & Vilma Johansson
# June 2022
#
# ------------------------------------------------------------------------------
# MODULE OF GEOMETRICAL CLASSES AND FUNCTIONS
#
# Contents
#   0. Input Modules
#   1. Corrugation Parameters [class]
#   2. Coordinates of Longitudinal Segments
#   3. Sectional Properties
#   4. Connections (LCC analysis)
#   5. Quantities (LCC analysis)
#   6. Painting (LCC analysis)
#
# ------------------------------------------------------------------------------
# 0. Input Modules
import numpy as np
import math

# ------------------------------------------------------------------------------
# 1. Corrugation Parameters
#
# Input:
#   Shape       Web shape, 'Corrugated' or 'Flat'           [object, -]
#   a1          Flat fold length                            [float, mm]
#   a3          Corrugation depth                           [float, mm]
#   alpha       Corrugation angle                           [float, mm]
#
# Output:
#   CorrPar     Corrugation class [a1, a2, a3, a4, alpha]
#               With:
#                   a1      Flat fold length                [float, mm]
#                   a2      Inclined length                 [float, mm]
#                   a3      Corrugation depth               [float, mm]
#                   a4      Corrugation length              [float, mm]
#                   alpha   Corrugation angle               [float, degrees]
# ------------------------------------------------------------------------------

class CorrugationParameters:
    def __init__(self, Shape, a1, a3, alpha):
        if Shape == 'Corrugated':
            self.a1 = a1
            self.a2 = a3/math.sin(math.radians(alpha))
            self.a3 = a3
            self.a4 = a3/math.tan(math.radians(alpha))
            self.alpha = alpha

# ------------------------------------------------------------------------------
# 2. Coordinates of Longitudinal Segments
# Input:
#   L           Length of bridge                                [float, mm]
#   N           Number of segments                              [integer, -]
#   m           Vector of variable x-positions, [Y x Y]         [float, mm]
#
# Output:
#   xStart      Vector of start positions of longitudinal
#               segments, Nx1 [Y x Y]                           [float, mm]
#   xEnd        Vector of end positions of longitudinal
#               segments, Nx1 [Y x Y]                           [float, mm]
#   SegPos      'End' or 'Span' segment                         [object, -]
#
```

```
# ------------------------------------------------------------------------

def SegCoord(L, N, m):
    # Variation vector
    l0 = (L/2)/N # initial length of each segment
    p = np.empty(N-1)
    for i in range (0, N-1):
        p[i] = (i+1)*l0 + m[i]

    # Coordinate vectors
    x_start = np.empty(N)
    x_end = np.empty(N)
    SegPos = np.empty(N, object)

    # Start segment
    x_start[0] = 0
    x_end[0] = p[0]
    SegPos[0] = 'End'

    # End segment
    x_start[N-1] = p[N-2]
    x_end[N-1] = L/2
    SegPos[N-1] = 'Span'

    # Span segments
    for i in range (1,N-1):
        x_start[i] = p[i-1]
        x_end[i] = p[i]
        SegPos[i] = 'Span'

    return[x_start, x_end, SegPos]

# ------------------------------------------------------------------------
# 3. Sectional Properties
#
# Input:
#   hw              Height of web                                [float, mm]
#   hwo_eff         Effective height of compressed part of web, [float, mm]
#                   set to 0 if CSC < 4
#   tw              Thickness of web                             [float, mm]
#   bfo             Upper flange (effective) width               [float, mm]
#   tfo             Upper flange thickness                       [float, mm]
#   bfu             Lower flange (effective) width               [float, mm]
#   tfu             Lower flange thickness                       [float, mm]
#   hc              Height of concrete deck                      [float, mm]
#   bc_eff          Effective concrete width                     [float, mm]
#   nL              Modular ratio (if no nL, set to 1)           [float, -]
#   Shape           Web shape, 'Corrugated' or 'Flat'            [object, -]
#
# Output:
#   Class SecPar(tp, tp_s, tp_c, I, Su, So, Wo, Wu, nL)
#       with    tp      Neutral axis distance                    [float, mm]
#               tp_s    Neutral axis distance, steel section     [float, mm]
#               tp_c    Neutral axis distance, concrete section [float, mm]
#               I       Moment of inertia                        [float, mm^4]
#               Su      Static moment, lower flange              [float, mm^3]
#               So      Static moment, upper flange              [float, mm^3]
#               Wo      Sectional modulus, upper fibre           [float, mm^3]
#               Wu      Sectional modulus, lower fibre           [float, mm^3]
#               nL      Modular ratio                            [float, -]
#
#   OBS!:
#   Due to elastic analysis, I and W are elastic also for parts in CSC 1 and 2.
#   Assumption: Same steel material for all parts of the steel section.
#   Neutral axis distance, tp, is defined from top of section and down.
# ------------------------------------------------------------------------

class SectionalProperties:
    def __init__(self, hw, hwo_eff, tw, bfo, tfo, bfu, tfu, hc, bc_eff, nL,
                 Shape):
```

```
if Shape == 'Corrugated':
    tw = 0

# Area [mm^2]
Afo = bfo*tfo
Afu = bfu*tfu
Ac = bc_eff/nL*hc # concrete section transferred to steel section
Aw = hw*tw

# z-distances from top of section [mm]
dc = hc/2
dfo = hc+tfo/2
dfu = hc+tfo+hw+tfu/2
dw = hc+tfo+hw/2

# Neutral axis (z-distance from top) [mm]
tp = (Afo*dfo+Afu*dfu+Aw*dw+Ac*dc)/(Afo+Afu+Aw+Ac)
tp_s = (Afo*(dfo-hc)+Afu*(dfu-hc)+Aw*(dw-hc))/(Afo+Afu+Aw)

# Web in CSC 4 (flat webs)
if Shape == 'Flat' and hwo_eff != 0:
    e = 1
    i = 0
    while e > 1e-5:
        i = i+1
        Awo1 = 0.4*hwo_eff*tw
        Awo2 = 0.6*hwo_eff*tw
        Awu = (hw-(tp-hc-tfo))*tw

        dwo1 = hc+tfo+(0.4*hwo_eff/2)
        dwo2 = tp-(0.6*hwo_eff/2)
        dwu = tp+(hc+tfo+hw-tp)/2

        tp_new = (Afo*dfo+Afu*dfu+Awo1*dwo1+Awo2*dwo2+Awu*dwu+\
                  Ac*dc)/(Afo+Afu+Awo1+Awo2+Awu+Ac)
        e = abs(tp-tp_new)
        tp = tp_new
        if i == 30:
            break
    tp_s = (Afo*(dfo-hc)+Afu*(dfu-hc)+Awo1*(dwo1-hc)+Awo2*(dwo2-hc)+\
            Awu*(dwu-hc))/(Afo+Afu+Aw)

# Neutral axis
self.tp = tp
self.tp_s = tp_s+hc # defined from top of concrete section
self.tp_c = hc/2
# tp is defined from top of section and down

# Composite area
if Shape == 'Flat' and hwo_eff != 0:
    self.A = Afo + Afu + Ac + Awo1+Awo2+Awu
else:
    self.A = Afo + Afu + Ac + hw*tw

# Inertia [mm^4]
Ic = bc_eff/nL*hc**3/12+Ac*(tp-dc)**2
Ifo = bfo*tfo**3/12+Afo*(tp-dfo)**2
Ifu = bfu*tfu**3/12+Afu*(tp-dfu)**2
if Shape == 'Flat':
    if hwo_eff == 0:
        Iw = tw*hw**3/12+Aw*(tp-dw)**2
        I = Ic+Ifo+Ifu+Iw
    else:
        Iwo1 = tw*(0.4*hwo_eff)**3/12+Awo1*(tp-dwo1)**2
        Iwo2 = tw*(0.6*hwo_eff)**3/12+Awo2*(tp-dwo2)**2
        Iwu = tw*(hw-(tp-hc-tfo))**3/12+Awu*(tp-dwu)**2
        I = Ic+Ifo+Ifu+Iwo1+Iwo2+Iwu
if Shape == 'Corrugated':
    I = Ic+Ifo+Ifu
```

```
        self.I = I

        # Static moment [mm^3]
        self.Su = tfu * bfu * abs(tp-dfu)
        self.So = tfo * bfo * abs(tp-dfo)
        self.Sc = hc * bc_eff/nL * abs(tp-dc)

        # Sectional modulus (elastic!) [mm^3]
        self.Wo = I/-(tp-hc) # negative as there will be compression at top
        self.Wu = I/(hc+tfo+hw+tfu-tp)  # positive --> tension at bottom
        self.Wc = I/-tp # negative as there will be compression at top

        # Modular ratio [-]
        self.nL = nL

# ------------------------------------------------------------------------
# 4. Connection (LCC analysis)
#
# Input:
#   SteelGrade      'S355', 'S460' or 'Duplex'          [object, -]
#   L               Total length of web                 [float, mm]
#   Nseg            Number of segments                  [integer, -]
#   Nsp             Number of splices                   [integer, -]
#   numCB           Number of crossbeams                [integer, -]
#   a               Weld throat thickness               [float, mm]
#   hw              Height of web                       [float, mm]
#   tw              Thickness of web                    [float, mm]
#   bfo_v           Vector of upper flange widths       [float, mm]
#   tfo_v           Vector of upper flange thicknesses  [float, mm]
#   bfu_v           Vector of lower flange widths       [float, mm]
#   tfu_v           Vector of lower flange thicknesses  [float, mm]
#   hw_cbe          Height of end beam web height       [float, mm]
#   xstart          Vector of segment start coord.      [float, mm]
#   xend            Vector of segment end coord.        [float, mm]
#   CBcoord         Vector of crossbeam coord. pos.     [float, mm]
#
# Output:
#   Connection      Matrix [Nwx6]                       [objects and floats]
# ------------------------------------------------------------------------

def Connection(SteelGrade, L, Nseg, Nsp, numCB, a, hw, tw, bfo_v, tfo_v,
               bfu_v, tfu_v, hw_cbe, xstart, xend, CBcoord, d_bolt, d_stud,
               nStuds):
# Note that L for corrugated webs is L*rc

    Nw = 1 + 3 + 2*Nseg + 2 + 2*(Nseg-1) + 3 + 1 + 1
    Connection = np.empty((Nw,4), dtype=object)
    # --- Row 0: Column titles
    Connection[0,:] = ['Connection_type', 'Working_place', 'Thickness',
                       'Length']
    # Stiffeners
    # --- Row 1: Web to vertical stiffener
    Connection[1,:] = ['Fillet', 'Workshop', a, 4*numCB*hw*1e-3]

    # --- Row 2: Span crossbeams to vertical stiffener
    Connection[2,:] = ['Bolted', 'On site', d_bolt, 12*(numCB-2)]

    # --- Row 3: End crossbeams to vertical stiffener
    Connection[3,:] = ['Fillet', 'On site', a, 8*hw_cbe*1e-3]

    # --- Row 4 to (3+Nseg): Upper flange to vertical stiffener
    for i in range(0,Nseg):
        n = 0
        for j in range(0, len(CBcoord)):
            if CBcoord[j] >= xstart[i] and CBcoord[j] <= xend[i]:
                n = n + 1
        n = 2*n     # segment coordinates only half bridge
        Connection[4+i,:] = ['Butt', 'Workshop', tfo_v[i],
                             2*n*(bfo_v[i]-tw)/2*1e-3]
```

```
    # --- Row (4+Nseg) to (3+2*Nseg): Lower flange to vertical stiffener
    for i in range(0,Nseg):
        n = 0
        for j in range(0, len(CBcoord)):
            if CBcoord[j] >= xstart[i] and CBcoord[j] <= xend[i]:
                n = n + 1
        n = 2*n      # segment coordinates only half bridge
        Connection[4+Nseg+i,:] = ['Fillet', 'Workshop', a,
                                  4*n*(bfu_v[i]-tw)/2*1e-3]

    # I-beam
    # --- Row (4+2*Nseg) to (5+2*Nseg): Flanges to web
    Connection[4+2*Nseg,:] = ['Butt', 'Workshop', tw, 2*L*1e-3]
    Connection[5+2*Nseg,:] = ['Fillet', 'Workshop', a, 4*L*1e-3]

    # Segment change
    Nseg_change = 0
    # --- Row (6+2*Nseg) to (4+3*Nseg): Upper flange change
    # --- Row (5+3*Nseg) to (3+4*Nseg): Lower flange change
    for i in range(0,Nseg-1):
        if bfo_v[i] == bfo_v[i+1] and tfo_v[i] == tfo_v[i+1] and \
        bfu_v[i] == bfu_v[i+1] and tfu_v[i] == tfu_v[i+1]:
            Connection[6+2*Nseg+i,:] = ['Butt', 'Workshop', 0, 0]
            Connection[5+3*Nseg+i,:] = ['Butt', 'Workshop', 0, 0]
        else:
            Connection[6+2*Nseg+i,:] = ['Butt', 'Workshop',
                                        np.mean(np.array([tfo_v[i],tfo_v[i+1]])),
                                        2*np.mean(np.array([bfo_v[i],bfo_v[i+1]]))*\
                                        1e-3]
            Connection[5+3*Nseg+i,:] = ['Butt', 'Workshop',
                                        np.mean(np.array([tfu_v[i],tfu_v[i+1]])),
                                        2*np.mean(np.array([bfu_v[i],bfu_v[i+1]]))*\
                                        1e-3]
            Nseg_change = Nseg_change + 1

    # --- Row (4+4*Nseg): Web
    Connection[4+4*Nseg,:] = ['Butt', 'Workshop', tw, hw*Nseg_change*1e-3]

    # Splices
    # --- Row (5+4*Nseg): Web splices
    Connection[5+4*Nseg,:] = ['Butt', 'On site', tw, Nsp*hw*1e-3]

    # --- Row (6+4*Nseg): Upper flange splices
    Connection[6+4*Nseg,:] = ['Butt', 'On site', np.mean(tfo_v),
                              2*np.mean(bfo_v)*1e-3]

    # --- Row (7+4*Nseg): Lower flange splices
    Connection[7+4*Nseg,:] = ['Butt', 'On site', np.mean(tfu_v),
                              2*np.mean(bfu_v)*1e-3]

    # Shear studs
    # --- Row (8+4*Nseg): Number of shear studs
    Connection[8+4*Nseg,:] = ['ShearStuds', 'Workshop', d_stud, 2*nStuds]


    return Connection

# ------------------------------------------------------------------------------
# 5. Quantities (LCC analysis)
#
# Input:
#   SteelGrade          'S355', 'S460' or 'Duplex'          [object, -]
#   Shape               Web shape, 'Flat' or 'Corrugated'   [object, -]
#   weight_plate        Weight of all welded plates         [float, kg]
#   weight_hotrolled    Weight of all hotrolled sections    [float, kg]
#   weight_hollow       Weight of all hollow sections       [float, kg]
#   paint_area          Total area to be painted            [float, m2]
#   volume_concrete     Volume of concrete deck             [float, m3]
#   weight_rebars       Weight of all reinforcement         [float, kg]
#
```

```
# Output:
#   Quantities      Matrix [2x9]                    [objects and floats]
# ------------------------------------------------------------------------------

def Quantities(SteelGrade, Shape, weight_plate, weight_hotrolled,
               weight_hollow, weight_weld, paint_area, volume_concrete,
               weight_rebars):
    if SteelGrade == 'S355':
        name_plate = 'steel_s355_plate'
        name_hotrolled = 'steel_s355_hot_rolled'
        name_hollow = 'steel_s355_hollow_section'
        name_weld = 'welds_steel_s355'

    if SteelGrade == 'S460':
        name_plate = 'steel_s460_plate'
        name_hotrolled = 'steel_s460_hot_rolled'
        name_hollow = 'steel_s460_hollow_section'
        name_weld = 'welds_steel_s460'

    if SteelGrade == 'Duplex':
        name_plate = 'stainless_steel_plate'
        name_hotrolled = 'stainless_steel_hot_rolled'
        name_hollow = 'stainless_steel_hollow_section'
        name_weld = 'welds_stainless_steel'

    Quantities = np.empty((2,7), dtype=object)
    # --- Row 0: Column titels
    Quantities[0,:] = [name_plate, name_hotrolled, name_hollow, name_weld,
                       'paint', 'concrete_plant', 'steel_reinforcement_bars']
    # --- Row 1: Input data
    Quantities[1,:] = [weight_plate, weight_hotrolled, weight_hollow,
                       weight_weld, paint_area, volume_concrete,
                       weight_rebars]

    return Quantities

# ------------------------------------------------------------------------------
# 6. Painting (LCC analysis)
#
# Input:
#   SteelType       'Carbon' or 'Stainless'             [objective, -]
#   paint_area      Total area to be painted            [float, m^2]
#   Nsp             Number of splices                   [integer, -]
#   numCB           Number of crossbeams                [integer, -]
#   hw              Height of web                       [float, mm]
#
# Output:
#   Painting        Matrix [Nwx6]                       [objects and floats]
# ------------------------------------------------------------------------------
def Painting(SteelType, paint_area, Nsp, numCB, hw):
    Painting = np.empty((3,2), dtype=object)
    # --- Row 0: Column titels
    Painting[0,:] = ['Paint_area', 'Workin_place']
    if SteelType == 'Carbon':
        # --- Row 1: Initial painting
        Painting[1,:] = [paint_area, 'Workshop']
        # --- Row 2: On site painting
        Painting[2,:] = [2*hw*(numCB+Nsp)*1e-3, 'On site']
    if SteelType == 'Stainless':
        # --- Row 1: Initial painting
        Painting[1,:] = [0, 'Workshop']
        # --- Row 2: On site painting
        Painting[2,:] = [0, 'On site']
    return Painting
```

C.4 Module: LoadFunctions

```
# Master's Thesis
#
# Design Optimization of Composite Road Bridges using Genetic Algorithms
# -Corrugated Web Stainless Steel Girders versus Flat Web Carbon Steel Girders
#
# Cecilia Hallgren & Vilma Johansson
# June 2022
#
# ------------------------------------------------------------------------------
# MODULE OF LOAD FUNCTIONS
#
# Contents
#   0. Input Modules
# 1. Self-weight
# 2. Traffic Load: Load Model 1 (LM1)
#   3. Traffic Load: Load Model 2 (LM2)
#   4. Acceleration/Braking Load
#   5. Fatigue Traffic Load: Load Model 3 (FLM3)
#   6. Wind Load
#   7. Temperature Load
#   8. Shrinkage Load
#
# ------------------------------------------------------------------------------

# 0. Input Modules
import numpy as np
import math

# ------------------------------------------------------------------------------
#
# 1. Self-weight
# Input:
#    Shape            Web shape, 'Corrugated' or 'Flat'          [object, -]
#    MtrlPar_w        Material parameters of web                 [class]
#    MtrlPar_fo       Material parameters of upper flange        [class]
#    MtrlPar_fu       Material parameters of lower flange        [class]
#    MtrlPar_c        Material parameters of concrete            [class]
#    MtrlPar_p        Material parameters of pavement            [class]
#    L                Length of bridge                           [float, mm]
#    B                Width of bridge                            [float, mm]
#    hw               Height of web of main girder               [float, mm]
#    tw               Thickness of web of main girder            [float, mm]
#    bfo              Width of upper flange of main girder       [float, mm]
#    tfo              Thickness of upper flange of main girder   [float, mm]
#    bfu              Width of lower flange of main girder       [float, mm]
#    tfu              Thickness of lower flange of main girder   [float, mm]
#    a                a-distance for welds                       [float, mm]
#    CorrPar          Corrugation parameters                     [class]
#    hc               Height of concrete                         [float, mm]
#    hp               Height of pavement                         [float, mm]
#
# Output:
#    G_sgirders       Self-weight of main girders                [float, N/mm]
#    G_concrete       Self-weight of concrete after hardeing     [float, N/mm]
#    G_unconcrete     Self-weight of unhardened concrete         [float, N/mm]
#    G_pave           Self-weight of pavement                    [float, N/mm]
#    G_cb             Self-weight of crash barriers              [float, N/mm]
#    G_fw             Self-weight of form-work                   [float, N/mm]
#
# ------------------------------------------------------------------------------

def SelfWeight(Shape, MtrlPar_w, MtrlPar_fo, MtrlPar_fu, MtrlPar_c, MtrlPar_p,
               L, B, hw, tw, bfo, tfo, bfu, tfu, a, CorrPar, hc, hp):
 # Gravitational Constant [m/s^2]
    g = 9.82

 # Self-weight of steel girders [N/mm]
```

```
    if Shape == 'Corrugated':
        rc = (CorrPar.a1+CorrPar.a2)/(CorrPar.a1+CorrPar.a4)
        A_rho = tfu*bfu*10**(-6)*MtrlPar_fu.rho+\
        tfo*bfo*10**(-6)*MtrlPar_fo.rho+hw*tw*10**(-6)*rc*MtrlPar_w.rho
        G_sgirders = A_rho*1e-3*g
    if Shape == 'Flat':
        A_rho = tfu*bfu*10**(-6)*MtrlPar_fu.rho+\
        tfo*bfo*10**(-6)*MtrlPar_fo.rho+hw*tw*10**(-6)*MtrlPar_w.rho
        G_sgirders = A_rho*1e-3*g

 # Self-weight of concrete [N/mm]
    Ac = B*hc*10**(-6)
    G_concrete = (Ac*MtrlPar_c.rho_c)/2
    G_unconcrete = (Ac*MtrlPar_c.rho_unc)/2

 # Self-weight of pavement [N/mm]
    Ap=B*hp*10**(-6)
    G_pave=(Ap*MtrlPar_p.rho)/2/1000

 # Self-weight of crash barriers [N/mm]
    G_cb=(0.5*10**3/2)/1000

 # Self-weight of form-work [N/mm]
    G_fw=(0.5*(B/1000+2*hc/1000))/2

    return (G_sgirders, G_concrete, G_unconcrete, G_pave, G_cb, G_fw)

# ------------------------------------------------------------------------------
#
# 2. Traffic Load: Load Model 1 (LM1)
# Output:
#    qk      Vector of distributed traffic load, 1Xi [1 X 4]    [float, N/mm^2]
#    Qk      Vector of point loads                              [float, N]
#    PC      Minimum distance from edge to application of axle  [float, mm]
#    w       Width of lanes                                     [float, mm]
#    C       C-C distance between axle loads                    [float, mm]
#
# ------------------------------------------------------------------------------

def TrafficLoadLM1():
    # Distances [mm]:
    PC=500              # From bridge edge to application of axle load
    C=2000              # C-C distance between axle loads
    w=3000              # Width of lanes

    # Defined values of loads (Table 4.2 SS-EN 1991-2)
    #   qi in kN/m^2 and Qi in kN
    # Lane 1
    q1k_=9
    Q1k_=300
    # Lane 2
    q2k_=2.5
    Q2k_=200
    # Lane 3
    q3k_=2.5
    Q3k_=100
    # Other:
    qik_=2.5
    Qik_=0

    # Definition of the adjustment factors (TSFS 2018:57 Section 11 and
    # Krav Brobyggande Section B.3.2.1.3)
    # Lane 1
    alphaQ1=0.9
    alphaq1=0.8
    # Lane 2
    alphaQ2=0.9
    alphaq2=1
    # Lane 3
    alphaQ3=0
```

```
    alphaq3=1
    # Other:
    alphaQi=0
    alphaqi=1

    # Loads [N] and [N/mm^2]:
    # Lane 1
    q1k=q1k_*alphaq1*1e-3
    Q1k=Q1k_*alphaQ1*1e3
    # Lane 2
    q2k=q2k_*alphaq2*1e-3
    Q2k=Q2k_*alphaQ2*1e3
    # Lane 3
    q3k=q3k_*alphaq3*1e-3
    Q3k=Q3k_*alphaQ3*1e3
    # Other:
    qik=qik_*alphaqi*1e-3
    Qik=Qik_*alphaQi*1e3

    qk = np.array([q1k, q2k, q3k, qik])
    Qk = np.array([Q1k, Q2k, Q3k, Qik])

    return(qk, Qk, PC, w, C)

# ------------------------------------------------------------------------------
#
# 3. Traffic Load: Load Model 2 (LM2)
# Output:
#    Qk          Axle load                                          [float, N]
#    C           C-C distance between axle loads                    [float, mm]
#    b_wheels    Length/width of squares representing the wheels    [float, mm]
#
# ------------------------------------------------------------------------------

def TrafficLoadLM2():

    C = 2000            # [mm] C-C distance between axle loads
    b_wheels = 350      # [mm] Length/width of squares representing the wheels
    alpha_Q1 = 0.9      # [-]  Adjustment factor
    Qak = 400*10**3     # [N]  Axle Load

    Qk = Qak * alpha_Q1

    return (Qk, C, b_wheels)
# ------------------------------------------------------------------------------
#
# 4. Acceleration/Braking Load
# Input:
#    L           Length of bridge                    [float, mm]
#
# Output:
#    Qkacc       Acceleration/braking load           [float, N]
#
# ------------------------------------------------------------------------------

def AccBraLoad(L):

    w1 = 3000       # [mm] Width of lane
    alpha_Q1=0.9    # [-] Adjustment factor
    alpha_q1=0.8    # [-] Adjustment factor
    Q1k=300.0e3     # [N] Axle Load
    q1k=9.0e-3      # [N/mm^2] Distributed load

    Qkacc = 0.6*alpha_Q1*2*Q1k+0.1*alpha_q1*q1k*w1*L

    if Qkacc <= 180*alpha_Q1*1e3:
        Qkacc = 180*alpha_Q1*1e3

    elif Qkacc >= 900*1e3:
        Qkacc = 900*1e3
```

```
    else:
        Qkacc = 0.6*alpha_Q1*2*Q1k+0.1*alpha_q1*q1k*w1*L

    return Qkacc

# ------------------------------------------------------------------------------
#
# 5. Fatigue Traffic Load: Load Model 3 (FLM3)
# Output:
#    q       Fatigue Traffic Load                                  [float, N]
#    Ctran   C-C distance between axle loads in transverse dir     [float, mm]
#    Clong   C-C distance between acle loads in longitudinal dir   [float, mm]
#    Cvehic  C-C distance between vehicles                         [float, mm]
#    w       Width of lanes                                        [float, mm]
#
# ------------------------------------------------------------------------------

def FatigueTrafficLoadLM3():

    # C-C distances [mm]:
    Ctran=2000          # Between axle loads in transverse direction
    Clong=1200          # Between axle loads in longitudinal direction
    Cvehic=6000         # Between vehicles

    # Parameters:
    w=3000              # [mm] Width of lanes
    q=120.0e3           # [N] Axle loads

    return (q, Ctran, Clong, Cvehic, w)

# ------------------------------------------------------------------------------
#
# 6. Wind Load
# Input:
#    qp          Characteristic velocity pressure            [float, kN/m^2]
#    B           Width of bridge                             [float, mm]
#    h           Total height of bridge                      [float, mm]
#    Vehicle     If vehicle is included or not,'Yes' or 'No' [object, -]
#
# Output:
#    Fwind       Wind load                                   [float, N/mm]
#
# ------------------------------------------------------------------------------

def WindLoad(qp, B, h, Vehicle):
    h_vehicle=2000  # Vehicle height [mm] - Assumed
    d1=300          # Height of crash barriers according to Table 8.1 in
                    # SS-EN 1991-1-4 [mm]

# If vehicles are included or not
    if Vehicle == 'yes':

    # Total height of bridge and vehicle [mm]:
       dtot = h+h_vehicle

    # Calculation of the force coefficient c_fx0:
       if B/dtot > 0 and B/dtot < 4:
          c_fx0 = -1.1/4*(B/dtot)+2.4
       else:
          c_fx0 = 1.3
       C = qp*c_fx0             # kN/m^2

    # Calculation of wind force:
       Fwind = C*dtot*1e-3

    if Vehicle == 'no':

    # Total height of bridge and crash barriers [mm]:
       dtot = h+2*d1
```

```
    # Calculation of the force coefficient c_fx0:
       if B/dtot > 0 and B/dtot < 4:
          c_fx0 = -1.1/4*(B/dtot)+2.4
       else:
          c_fx0 = 1.3
       C = qp*c_fx0*1e-3

    # Calculation of wind force:
       Fwind = C*dtot

    return (Fwind)

# ------------------------------------------------------------------------------
#
# 7. Temperature Load
# Input:
#    MtrlPar_s      Steel material parameters                    [class]
#    MtrlPar_c      Concrete material parameters                 [class]
#    Delta_T_cs     Difference in the uniform temperature
#                   component between different parts            [float, dC]
#   T_max           Maximum ambient temperature            [float, dC]
#   T_min           Minimum ambient temperature          [float, dC]
#   T0              Initial bridge temperature             [float, dC]
#   As              Steel area                                   [float, mm^2]
#
# Output:           F_temp_t,  F_temp_c
#    F_temp_t       Temperature load, tension          [float, N]
#    F_temp_c       Temperature load, compression        [float, N]
#
# ------------------------------------------------------------------------------
# OBS! Only dublex stainless steel may be supported. For use of other stainless
# steel material, check if Es is the same for whole cross-section.

def TemperatureLoad(MtrlPar_s, MtrlPar_c, Delta_T_cs, T_max, T_min, T0, As):

    if MtrlPar_s.SteelType == 'Stainless':

        T_emin=T_min+4
        T_emax=T_max+4

        Delta_Tn_sho=T0-T_emin
        Delta_Tn_ext=T_emax-T0

        Epsilon_max_t=(MtrlPar_s.alpha_s-MtrlPar_c.alpha_c)*Delta_Tn_ext\
            +MtrlPar_s.alpha_s*Delta_T_cs
        Epsilon_max_c=-(MtrlPar_s.alpha_s-MtrlPar_c.alpha_c)*Delta_Tn_sho\
            -MtrlPar_s.alpha_s*Delta_T_cs

        F_temp_t=Epsilon_max_t*MtrlPar_s.Es*As*1e3
        F_temp_c=Epsilon_max_c*MtrlPar_s.Es*As*1e3

    else:
        F_temp_t=0
        F_temp_c=0

    return (F_temp_t,  F_temp_c)

# ------------------------------------------------------------------------------
#
# 8. Shrinkage Load
# Input:
#    SteelClass      Reinforcement steel class, 'S' or 'N'       [object, -]
#    MtrlPar_w       Material parameters of steel                [class]
#    MtrlPar_c       Material parameters of concrete             [class]
#    Ac            Concrete Area                              [float, mm^2]
#  u           Length exposed to drying           [float, mm]
#  RH          Relative humidity                  [float, %]
#    nL              Modular Ratio                               [float, days]
#
```

```
# Output:
#    Fsc        Shrinkage load                                  [float, N]
#
# ------------------------------------------------------------------------------

def ShrinkageLoad(SteelClass, MtrlPar_w, MtrlPar_c, Ac, u, RH, nL):

# Drying shrinkage strain

  # k_h - coefficient dependent on the notional size h_0. Defined in
  # SS_EN1992-1-1 Section 3.1.4
    h0=2*Ac/u
    if h0 < 100:
        kh = 1.0
    elif h0 >= 100 and h0<200:
       kh = -0.15/100*h0+1.15
    elif h0 >= 200 and h0<300:
        kh = -0.1/100*h0+0.65
    elif h0 >= 300 and h0<500:
        kh =- 0.05/200*h0+0.825
    else:
        kh=0.7
  # Beta_ds - SS-EN 1992-1-1 Equation 3.10 - Long term loading
    beta_ds = 1

  # epsilon_cd0 - unrestrained drying shrinkage values
  # Defined in 1992-1-1 Annex B Equation B.11
    if SteelClass=='S':
        alpha_ds1=3.
        alpha_ds2=0.13
    elif SteelClass=='N':
        alpha_ds1=4.
        alpha_ds2=0.12
    else:
        alpha_ds1=6.
        alpha_ds2=0.11
    beta_RH=1.55*(1-(RH/100))
    epsilon_cd0=0.85*\
        (220+110*alpha_ds1*math.exp(-alpha_ds2*(MtrlPar_c.fcm/10)))*1e-6*
    beta_RH

    epsilon_cd=beta_ds*kh*epsilon_cd0
# Autogenous shrinkage strain
    # beta_as - SS-EN 1992-1-1 Equation 3.13 - Equal to 1 since long term
    beta_as=1.0

    # epsilon_ca_inf - Defined in SS-EN 1992-1-1 Equation 3.12
    epsilon_ca_inf=2.5*(MtrlPar_c.fck-10)*10**(-6)
    epsilon_ca=beta_as*epsilon_ca_inf

# Total shrinkage strain - SS-EN 1992-1-1 Equation 3.8
    epsilon_cs=epsilon_cd+epsilon_ca

# Shrinkage force
    n0=MtrlPar_w.Es/MtrlPar_c.Ecm
    E_ceff=(n0/nL)*MtrlPar_c.Ecm
    Fcs=epsilon_cs*E_ceff*Ac*1e3

    return Fcs
```

C.5 Module: DesignFunctions

```
# Master's Thesis
#
# Design Optimization of Composite Road Bridges using Genetic Algorithms
# -Corrugated Web Stainless Steel Girders versus Flat Web Carbon Steel Girders
#
# Cecilia Hallgren & Vilma Johansson
# June 2022
#
# ------------------------------------------------------------------------------
# MODULE OF DESIGN FUNCTIONS
#
# Contents
#   0. Input Modules
#   1. Cross-Section Class
#   2. Cross-Section Reduction of Parts in Cross-Sectional Class 4 (CSC4)
#   3. Lateral Torsional Buckling
#   4. Bending Moment Capacity of Steel Girders
#   5. Shear Buckling
#   6. Shear Resistance
#   7. Moment and Shear Interaction
#   8. Shear Lag Steel
#   9. Shear Lag Concrete
#   10. Modular Ratio
#   11. Weld Resistance
#   12. Shear Stud Resistance
#   13. Fatigue
#   14. Bending Moment Capacity of Composite Beam
#   15. Buckling (general)
#   16. Minimum Reinforcement In Longitudinal Direction
#
# ------------------------------------------------------------------------------
# 0. Input Modules
import math

# ------------------------------------------------------------------------------
# 1. Cross-Sectional Class
#
# Input:
#   MtrlPar_w   Material parameters of web                  [class]
#   MtrlPar_fo  Material parameters of upper flange         [class]
#   Shape       Shape of web, 'Corrugated' or 'Flat'        [object, -]
#   CorrPar     Corrugation parameters                      [class]
#   hw          Height of web                               [float, mm]
#   tw          Thickness of web                            [float, mm]
#   bfo         Width of top flange                         [float, mm]
#   a           a-distance for welds                        [float, mm]
#   tfo         Thickness of top flange                     [float, mm]
#
# Output:
#   CSC         Cross-section class [c_fo, c_web]           [object, -]
#               With:
#                   CSC_fo: CSC for upper flange, outstand part
#                   CSC_web: CSC for web
# ------------------------------------------------------------------------------
def CrossSectionClass(MtrlPar_w, MtrlPar_fo, Shape, CorrPar, hw, tw,
                      bfo, tfo, a):

 # Type: Carbon steel - flat web
    if MtrlPar_w.SteelType=='Carbon' and Shape=='Flat':
        cweb=hw-math.sqrt(2)*a
        cfo=bfo/2-tw/2

    # Web of girder - Table 5.2 in SS-EN 1993-1-1
        ct_web=cweb/tw
        epsilon_cw=math.sqrt(235/MtrlPar_w.fy)
        if ct_web <= 72*epsilon_cw:
            CSC_web = 1.
```

```
        elif ct_web <= 83*epsilon_cw and ct_web > 72*epsilon_cw:
            CSC_web = 2.
        elif ct_web <= 124*epsilon_cw and ct_web > 83*epsilon_cw:
            CSC_web = 3.
        else:
            CSC_web = 4.

    # Upper flange, outstand part - Table 5.2 in SS-EN 1993-1-1
        ct_fo=cfo/tfo
        epsilon_cf=math.sqrt(235/MtrlPar_fo.fy)
        if ct_fo <= 9*epsilon_cf:
            CSC_fo = 1.
        elif ct_fo <= 10*epsilon_cf and ct_fo > 9*epsilon_cf:
            CSC_fo = 2.
        elif ct_fo <= 14*epsilon_cf and ct_fo > 10*epsilon_cf:
            CSC_fo = 3.
        else:
            CSC_fo = 4.

# Type: Stainless steel - flat web
    elif MtrlPar_w.SteelType == 'Stainless' and Shape == 'Flat':
        cweb=hw-math.sqrt(2)*a
        cfo=bfo/2-tw/2

    # Web of girder - Table 5.2 in SS-EN 1993-1-4
    # with replacements according to SS-EN 1993 1-4 2006/A1:2015
        ct_web=cweb/tw
        epsilon_sw=math.sqrt((235/MtrlPar_w.fy)*(MtrlPar_w.Es/210))
        if ct_web <= 72*epsilon_sw:
            CSC_web = 1.
        elif ct_web <= 76*epsilon_sw and ct_web > 56*epsilon_sw:
            CSC_web = 2.
        elif  ct_web <= 90*epsilon_sw and ct_web > 58.2*epsilon_sw:
            CSC_web = 3.
        else:
            CSC_web = 4.

    # Upper flange, outstand part - Table 5.2 in SS-EN 1993-1-4
    # with replacements according to SS-EN 1993 1-4 2006/A1:2015
        ct_fo=cfo/tfo
        epsilon_sf=math.sqrt((235/MtrlPar_fo.fy)*(MtrlPar_fo.Es/210))
        if ct_fo  <= 9*epsilon_sf:
            CSC_fo = 1.
        elif ct_fo <= 10*epsilon_sf and ct_fo > 9*epsilon_sf:
            CSC_fo = 2.
        elif  ct_fo <= 14*epsilon_sf and ct_fo > 9.4*epsilon_sf:
            CSC_fo = 3.
        else:
            CSC_fo = 4.

# Type: Stainless steel - corrugated web
    elif MtrlPar_w.SteelType == 'Stainless' and Shape == 'Corrugated':
        CSC_web=1.
        cfo_corr=bfo/2+CorrPar.a3/2-tw/2

    # Upper flange, outstand part - Table 5.2 in SS-EN 1993-1-4
    # with replacements according to SS-EN 1993 1-4 2006/A1:2015
        ct_fo=cfo_corr/tfo
        epsilon_sf=math.sqrt((235/MtrlPar_fo.fy)*(MtrlPar_fo.Es/210))
        if ct_fo  <= 9*epsilon_sf:
            CSC_fo = 1.
        elif ct_fo <= 10*epsilon_sf and ct_fo > 9*epsilon_sf:
            CSC_fo = 2.
        elif  ct_fo <= 14*epsilon_sf and ct_fo > 9.4*epsilon_sf:
            CSC_fo = 3.
        else:
            CSC_fo = 4.

# Type: Carbon steel - corrugated web
    else:
```

```
        CSC_web=1.
        cfo_corr=bfo/2+CorrPar.a3/2-tw/2

        # Upper flange, outstand part - Table 5.2 in SS-EN 1993-1-1
        ct_fo=cfo_corr/tfo
        epsilon_cf=math.sqrt(235/MtrlPar_fo.fy)
        if ct_fo  <= 9*epsilon_cf:
            CSC_fo = 1.
        elif ct_fo <= 10*epsilon_cf and ct_fo > 9*epsilon_cf:
            CSC_fo = 2.
        elif  ct_fo <= 14*epsilon_cf and ct_fo > 10*epsilon_cf:
            CSC_fo = 3.
        else:
            CSC_fo = 4.

    return [CSC_fo, CSC_web]

# ------------------------------------------------------------------------------
# 2. Cross-Section Reduction of Parts in Cross-Sectional Class 4 (CSC 4)
# Input:
#   MtrlPar_w       Material parameters of web                  [class]
#   MtrlPar_fo      Material parameters of upper flange         [class]
#   hw              (Effective) Height of web                   [float, mm]
#   tw              Thickness of web                            [float, mm]
#   bfo             Upper flange (effective) width              [float, mm]
#   tfo             Upper flange thickness                      [float, mm]
#   a               a-distance for welds                        [float, mm]
#   CSC_w           Cross-section class of web                  [object, -]
#   CSC_fo          Cross-section class of upper flange         [object, -]
#   psi_w           Stress distribution in web                  [-]
#   psi_fo          Stress distribution in flange               [-]
#
# Output:
#   hwo_eff         Effective height of web                     [float, mm]
#   bfo_eff         Effective width of upper flange             [float, mm]
#
# ------------------------------------------------------------------------------
def CSC4EffectiveWidth(MtrlPar_w, MtrlPar_fo, hw, tw, bfo, tfo, a,
                       CSC_w, CSC_fo, psi_w, psi_fo):
# Only Reduction if the part is in cross-sectional class 4.

    # Web
    if CSC_w < 4:
        hwo_eff = 0

    else:
      # Determination of epsilon-value:
        # Material: Carbon
        if MtrlPar_w.SteelType == 'Carbon':
            epsilon = math.sqrt(235/MtrlPar_w.fy)

        # Material: Stainless
        if MtrlPar_w.SteelType == 'Stainless':
            epsilon = math.sqrt((235/MtrlPar_w.fy)*(MtrlPar_w.Es/210))

      # Geometry parameters:
        b = hw
        psi = psi_w # with psi <= 0

      # Calculation of ks:
        if psi > -1.:
            ks = 7.81-6.29*psi+9.78*psi**2
        if psi == -1.:
            ks = 23.9
        if psi < -1.:
            ks = 5.98*(1-psi)**2
        lambda_p = (b/tw)/(28.4*epsilon*math.sqrt(ks))

      # Determination of rho (p):
        if lambda_p <= 0.673:
```

```
            p = 1.
        else:
            p = min((lambda_p-0.055*(3+psi))/(lambda_p**2),1.)

      # Effective height of web:
        hwo_eff = p*b/(1-psi)

    # Upper flange
    if CSC_fo < 4:
        bfo_eff = bfo
    else:
      # Determination of epsilon-value:
        # Material: Carbon
        if MtrlPar_fo.SteelType == 'Carbon':
            epsilon = math.sqrt(235/MtrlPar_fo.fy)

        # Material: Stainless
        if MtrlPar_fo.SteelType == 'Stainless':
            epsilon = math.sqrt((235/MtrlPar_fo.fy)*(MtrlPar_fo.Es/210))

      # Parameters:
        b = bfo-tw/2-math.sqrt(2)*a
        psi = psi_fo # with 1 > psi >= 0
        ks = 0.57-0.21*psi+0.07*psi**2
        lambda_p = (b/tfo)/(28.4*epsilon*math.sqrt(ks))

      # Determination of rho (p):
        if lambda_p <= 0.748:
            p = 1.
        else:
            p = min((lambda_p-0.188)/(lambda_p**2),1.)

      # Effective width:
        bfo_eff = p*b

    return [hwo_eff, bfo_eff]

# ------------------------------------------------------------------------------
# 3. Lateral Torsional Buckling (LT Buckling)
#
# Input:
#   MtrlPar_fo      Material parameters of upper flange     [class]
#   C               Distance between crossbeams             [float, mm]
#   bfo             Upper flange width                      [float, mm]
#   bfo_eff         Upper flange effective width            [float, mm]
#   tfo             Upper flange thickness                  [float, mm]
#   hw              Height of web                           [float, mm]
#   tfu             Lower flange thickness                  [float, mm]
#
# Output:
#   chi_LT          Lateral torsional buckling - reduction
#                   factor                                  [float, -]
# ------------------------------------------------------------------------------
def LTBuckling(MtrlPar_fo, C, bfo, bfo_eff, tfo, hw, tfu):

    # Critical buckling load (Euler)
    Iz = tfo*bfo**3/12
    Lcr = C
    Ncr = math.pi**2*MtrlPar_fo.Es*10**3*Iz/(Lcr**2)

    # LT-slenderness parameter (SS-EN 1993-2 6.3.4.2)
    A_eff = bfo_eff*tfo
    lambda_LT = math.sqrt(A_eff*MtrlPar_fo.fy/Ncr)

    # LT buckling reduction factor
    # (SS-EN 1993-1-1 6.3.2.2 and SS-EN 1993-1-4 5.4.3.1)
    if lambda_LT <= 0.4:
        chi_LT = 1
    else:
        # Material: Carbon
```

```
        if MtrlPar_fo.SteelType == 'Carbon':
            if (hw+tfo+tfu) <= 2*bfo:
                alpha_LT = 0.49
            else:
                alpha_LT = 0.76
            phi_LT = 0.5*(1+alpha_LT*(lambda_LT-0.2)+lambda_LT**2)
            chi_LT = min(1/(phi_LT+math.sqrt(phi_LT**2-lambda_LT**2)),1.)

        # Material: Stainless
        if MtrlPar_fo.SteelType == 'Stainless':
            alpha_LT = 0.76
            phi_LT = 0.5*(1+alpha_LT*(lambda_LT-0.4)+lambda_LT**2)
            chi_LT = min(1/(phi_LT+math.sqrt(phi_LT**2-lambda_LT**2)),1.)

    return chi_LT

# ------------------------------------------------------------------------------
# 4. Bending Moment Capacity of Steel Girders
#
# Input:
#   MtrlPar     Steel material parameters               [class]
#   SecPar      Sectional parameters                    [class]
#   Shape       Web shape, 'Flat' or 'Corrugated'       [object, -]
#   bfo         Upper flange width (effective)          [float, mm]
#   tfo         Upper flange thickness                  [float, mm]
#   bfu         Lower flange width (effective)          [float, mm]
#   tfu         Lower flange thickness                  [float, mm]
#   hw          Height of web                           [float, mm]
#   chi_LT      LT buckling reduction factor            [float, -]
#
# Output:
#   fyd         Design yield capacity                   [float, MPa]
# ------------------------------------------------------------------------------
def SteelMomentCapacity(MtrlPar, SecPar, Shape, bfo, tfo, bfu, tfu, hw,
                        chi_LT):

    # Partial Safety Factors:
    gammaM1 = 1.1
    gammaM0 = 1.

    # Shape of web: Flat
    if Shape == 'Flat':
        if chi_LT == 1:
            fyd = MtrlPar.fy/gammaM0
        else:
            fyd = chi_LT*MtrlPar.fy/gammaM1

    # Shape of web: corrugated
    if Shape == 'Corrugated':
            Mfu = bfu*tfu*MtrlPar.fy/gammaM0*(hw+(tfo+tfu)/2)
            fyd_u = abs(Mfu/SecPar.Wu)
            if chi_LT == 1:
                Mfo = bfo*tfo*MtrlPar.fy/gammaM0*(hw+(tfo+tfu)/2)
            else:
                Mfo1 = bfo*tfo*MtrlPar.fy/gammaM0*(hw+(tfo+tfu)/2)
                Mfo2 = bfo*tfo*chi_LT*MtrlPar.fy/gammaM1*(hw+(tfo+tfu)/2)
                Mfo = min(Mfo1,Mfo2)
            fyd_o = abs(Mfo/SecPar.Wo)
            fyd = min(fyd_u,fyd_o)

    return(fyd)

# ------------------------------------------------------------------------------
# 5. Shear Buckling
#
# Input:
#   MtrlPar_w       Material parameters of web              [class]
#   Shape           Web shape, 'Corrugated' or 'Flat'       [object, -]
#   hw              Height of web                           [float, mm]
#   tw              Thickness of web                        [float, mm]
```

```
#   CorrPar          Corrugation parameters                        [class]
#   a_stiff          C-C distance of transverse stiffeners         [float, mm]
#
# Output:
#   chi_w            Shear buckling reduction factor               [float, -]
# ------------------------------------------------------------------------------
def ShearBuckling(MtrlPar_w, Shape, hw, tw, CorrPar, a_stiff):

 # Shape of web: Flat
    if Shape == 'Flat':
    # Material: Carbon (SS-EN 1993-1-5 5.1)
        if MtrlPar_w.SteelType == 'Carbon':
            eta = 1.20
            epsilon = (235./MtrlPar_w.fy)**0.5

            # No longitudinal stiffeners
            if a_stiff/hw >= 1.:
                kappa_tao = 5.34+4.00*(hw/a_stiff)**2
            else:
                kappa_tao = 4.00+5.34*(hw/a_stiff)**2
            lambda_w = hw/(37.4*tw*epsilon*(kappa_tao)**0.5)

            # Check if shear buckling needs to be considered
            if hw/tw <= 31/eta*epsilon*(kappa_tao)**0.5:
                chi_w = 1.

            # Rigid endpost
            else:
                if lambda_w < 0.83/eta:
                    chi_w = eta
                if lambda_w >= 0.83/eta and lambda_w < 1.08:
                    chi_w = 0.83/lambda_w
                if lambda_w >= 1.08:
                    chi_w = 1.37/(0.7+lambda_w)

        # Material: Stainless (SS-EN 1993-1-4 5.6)
        if MtrlPar_w.SteelType == 'Stainless':
            eta = 1.20
            epsilon = (235./MtrlPar_w.fy*MtrlPar_w.Es/210.)**0.5

            # No longitudinal stiffeners
            if a_stiff/hw >= 1.:
                kappa_tao = 5.34+4.00*(hw/a_stiff)**2
            else:
                kappa_tao = 4.00+5.34*(hw/a_stiff)**2
            lambda_w = hw/(37.4*tw*epsilon*(kappa_tao)**0.5)

            # Check if shear buckling needs to be considered
            if hw/tw <= 23./eta*epsilon*(kappa_tao)**0.5:
                chi_w = 1.

            # Rigid endpost
            else:
                if lambda_w <= 0.6/eta:
                    chi_w = eta
                else:
                    chi_w = 0.11+0.64/lambda_w-0.05/(lambda_w**2)

 # Shape of web: corrugated (SS-EN 1993-1-5 Annex D)
    if Shape == 'Corrugated':
        w = CorrPar.a1+CorrPar.a4
        s = CorrPar.a1+CorrPar.a2
        Iz = tw*CorrPar.a3**2*(3*CorrPar.a1+CorrPar.a2)/12

        # Local buckling
        tao_crl = 4.83*MtrlPar_w.Es*10**3*(tw/max(CorrPar.a1, CorrPar.a2))**2
        lambda_wl = (MtrlPar_w.fy/(tao_crl*(3**0.5)))**0.5
        chi_wl = min(1.15/(0.9+lambda_wl),1.)

        # Global buckling
```

```
        Dx = MtrlPar_w.Es*10**3*tw**3/(12*(1-MtrlPar_w.nu**2))*w/s
        Dz = MtrlPar_w.Es*10**3*Iz/w
        tao_crg = 32.4/(tw*hw**2)*(Dx*Dz**3)**(1/4)
        lambda_wg = (MtrlPar_w.fy/(tao_crg*(3**0.5)))**0.5
        chi_wg = min(1.5/(0.5+lambda_wg**2),1.)

        chi_w = min(chi_wl, chi_wg)

    return chi_w

# ------------------------------------------------------------------------
# 6. Shear Resistance
#
#Input:
#   Shape           Web shape, 'Flat' or Corrugated'          [object, -]
#   MtrlPar_w       Material parameters of web                [class]
#   MtrlPar_fo      Material parameters of upper flange       [class]
#   MtrlPar_fu      Material parameters of lower flange       [class]
#   hw              Height of web                             [float, mm]
#   tw              Thickness of web                          [float, mm]
#   bfo             Upper flange (effective) width            [float, mm]
#   tfo             Upper flange thickness                    [float, mm]
#   bfu             Lower flange (effective) width            [float, mm]
#   tfu             Lower flange thickness                    [float, mm]
#   chi_w           Shear buckling reduction factor           [float, -]
#   MEd             Design moment                             [float, Nmm]
#   a_stiff         C-C distance of transverse stiffeners     [float, mm]
#   tp_s            Neutral axis of steel defined from top    [float, mm]
#
# Output:
#   VRd             Shear resistance                          [float, N]
#   VRd_w           Shear resistance of web                   [float, N]
# ------------------------------------------------------------------------
def ShearResistance(Shape, MtrlPar_w, MtrlPar_fo, MtrlPar_fu, hw, tw, bfo, tfo,
                    bfu, tfu, chi_w, MEd, a_stiff, tp_s):

    # Partial Safety Factors:
    gammaM1 = 1.1
    gammaM0 = 1.
    eta = 1.2

    # Web resistance (SS EN 1993-1-5 5.2)
    VRd_w = chi_w*MtrlPar_w.fy*hw*tw/(3**0.5*gammaM1)

    if Shape == 'Corrugated':
        VRd = VRd_w
        return (VRd, VRd_w)

    # Section Modulus
    Wy_fo = bfo*tfo*(tp_s/2)
    Wy_fu = bfu*tfu*(tfo+hw+tfu-tp_s-tfu/2)

    # Flange resistance (SS EN 1993-1-5 5.4)
    MRd_f = (Wy_fo*MtrlPar_fo.fy+Wy_fu*MtrlPar_fu.fy)/gammaM0
    if MRd_f > MEd:
        # Least axial resistance is governing
        if bfo*tfo*MtrlPar_fo.fy < bfu*tfu*MtrlPar_fu.fy:
            bf = bfo
            tf = tfo
            fyf = MtrlPar_fo.fy
        else:
            bf = bfu
            tf = tfu
            fyf = MtrlPar_fu.fy
        c = a_stiff*(0.25+(1.6*bf*tf**2*fyf)/(tw*hw**2*MtrlPar_w.fy))
        VRd_f = bf*tf**2*fyf/(c*gammaM1)*(1-(MEd/MRd_f)**2)
    else:
        VRd_f = 0

    if Shape == 'Flat':
```

```
        VRd = min(VRd_w+VRd_f, eta*MtrlPar_w.fy*hw*tw/(3**0.5*gammaM1))

        return (VRd, VRd_w)

# ------------------------------------------------------------------------
# 7. Moment and Shear Interaction
#
# Input:
#   Shape           Web shape, 'Flat' or 'Corrugated'          [object, -]
#   MtrlPar_s       Material parameters                        [class]
#   VEd             Design shear force                         [float, N]
#   VRd             Shear capacity                             [float, N]
#   VRd_w           Shear capacity of web only                 [float, N]
#   MEd             Design moment                              [float, Nmm]
#   hw              Height of web                              [float, mm]
#   tw              Thickness of web                           [float, mm]
#   bfo             Upper flange (effective) width             [float, mm]
#   tfo             Upper flange thickness                     [float, mm]
#   bfu             Lower flange (effective) width             [float, mm]
#   tfu             Lower flange thickness                     [float, mm]
#   hc              Height of concrete deck                    [float, mm]
#   bc_eff          Effective width of concrete deck           [float, mm]
#   nL              Modular ratio (permanent)                  [float, -]
#
# Output:
#   eta_int         Utilization ratio of interaction           [float, -]
# ------------------------------------------------------------------------
def Interaction(Shape, MtrlPar_s, VEd, VRd, VRd_w, MEd, hw,
                    tw, bfo, tfo, bfu, tfu, hc, bc_eff, nL):

    if Shape == 'Corrugated':
        eta_int = 0

    if Shape == 'Flat':
        if VEd <= 0.5*VRd:
            eta_int = 0
        else:
            # Areas and neutral axis:
            Afo = bfo*tfo   # obs. bfo = bfo_eff
            dfo = hc + tfo/2
            Afu = bfu*tfu
            dfu = hc + tfo + hw + tfu/2
            Aw = hw*tw
            dw = hc + tfo + hw/2
            Ac = bc_eff/nL*hc
            dc = hc/2
            tp = (Afo*dfo+Afu*dfu+Aw*dw+Ac*dc)/(Afo+Afu+Aw+Ac)

            # Plastic moment capacities
            gammaM0 = 1.
            MfRd = (Afo*abs(tp-dfo)+Afu*abs(tp-dfu)+Ac*abs(tp-dc))\
                    *MtrlPar_s.fy/gammaM0
            MplRd = (Afo*abs(tp-dfo)+Afu*abs(tp-dfu)+Aw*abs(tp-dw)+\
                     Ac*abs(tp-dc))*MtrlPar_s.fy/gammaM0

            # Interaction check:
            if MfRd >= MEd:
                eta_int = 0
            else:
                eta_int = MEd/MplRd+(1-MfRd/MplRd)*(2*VEd/VRd_w-1)**2

    return (eta_int)

# ------------------------------------------------------------------------
# 8. Shear Lag Steel
#
# Input:
#   bfo         Width of compressed flange                 [float, mm]
#   L           Span length                                [float, mm]
#   SegPos      Segment position, 'End' or 'Span'          [object, -]
```

```
#
# Output:
#   bfo_eff     Effective width of compressed flange        [float, mm]
# -----------------------------------------------------------------------------
def ShearLagSteel(bfo, L, SegPos):

    # SS-EN 1993-1-5 3.2.1
    bfo_0 = bfo/2

    # No longitudinal stiffeners are considered:
    alpha0 = 1

    kappa = alpha0*bfo_0/L

    if SegPos == 'Span':
        if kappa <= 0.02:
            beta = 1.0
        if kappa > 0.02 and kappa <= 0.7:
            # Only consider positive bending moment:
            beta = 1/(1+6.4*kappa**2)
        if kappa > 0.7:
            # Only consider positive bending moment
            beta = 1/(5.9*kappa)

    if SegPos == 'End':
        if kappa <= 0.7:
            beta1 = 1/(1+6.4*kappa**2)
        if kappa > 0.7:
            beta1 = 1/(5.9*kappa)
        beta = min((0.55+0.025/kappa)*beta1, beta1)

    bfo_eff = beta*bfo_0*2

    return bfo_eff

# -----------------------------------------------------------------------------
# 9. Shear Lag Concrete
#
# Input:
#   b0      Distance between shear connectors           [float, mm]
#   Bs      Distance between girders                    [float, mm]
#   Be1     Concrete deck cantilever 1                  [float, mm]
#   Be2     Concrete deck cantilever 2                  [float, mm]
#   L       Span length                                 [float, mm]
#   SegPos  Segment position, 'End' or 'Span'           [object, -]
#
# Output:
#   bc_eff  Effective width of concrete deck flange     [float, mm]
# -----------------------------------------------------------------------------
def ShearLagConcrete(b0, Bs, Be1, Be2, L, SegPos):
    # SS-EN 1994-2 5.4.1.2
    # Cantilever part:
    be1 = min(L/8, Be1-b0/2, Be2-b0/2)
    beta1 = min(0.55+0.025*L/be1, 1.)

    # Mid part:
    be2 = min(L/8, Bs/2-b0/2)
    beta2 = min(0.55+0.025*L/be2, 1.)

    if SegPos == 'Span':
        bc_eff = b0 + be1 + be2

    if SegPos == 'End':
        bc_eff = b0 + beta1*be1 + beta2*be2

    return bc_eff

# -----------------------------------------------------------------------------
# 10. Modular Ratio
# Input:
```

```
#    Phase       Phase to consider, 'ShortTerm', 'LongTerm'
#                or 'Shrinkage'                              [object, -]
#    MtrlPar_c   Concrete material parameters                [class]
#    MtrlPar_s   Steel material parameters                   [class]
#    RH          Relative humidity                           [float, %]
#    Ac          Cross-sectional area of concrete            [float, mm^2]
#    u           Length exposed to drying                    [float, mm]
#    t0          Age of concrete at loading                  [integer, days]
#
# Output:
#    nL          Modular Ratio                               [float, -]
#
# ------------------------------------------------------------------------------
def ModularRatio(Phase, MtrlPar_c, MtrlPar_s, RH, Ac, u, t0):

    if Phase == 'ShortTerm':
        psi = 0
    if Phase == 'LongTerm':
        psi = 1.1
    if Phase == 'Shrinkage':
        psi = 0.55

    h0 = 2*Ac/u

  # Creep coefficient, varpsi_t - SS-EN 1992-1-1 Annex B
    n0 = MtrlPar_s.Es/MtrlPar_c.Ecm
    alpha_1 = (35/MtrlPar_c.fcm)**0.7
    alpha_2 = (35/MtrlPar_c.fcm)**0.2

    if MtrlPar_c.fcm <= 35:
        var_psi_RH = 1+(1-RH/100)/(0.1*h0**(1/3))
    else:
        var_psi_RH = (1+(1-RH/100)/(0.1*h0**(1/3))*alpha_1)*alpha_2

    beta_c_t = 1
    beta_t0 = 1/(0.1+t0**0.20)
    beta_fcm = 16.8/math.sqrt(MtrlPar_c.fcm)
    var_psi_0 = var_psi_RH*beta_fcm*beta_t0
    varpsi_t = var_psi_0*beta_c_t

  # n_Lsc - Defined in SS-EN 1994-2, Section 5.4.2.2
    nL = n0*(1+psi*varpsi_t)

    return nL

# ------------------------------------------------------------------------------
# 11. Weld Resistance
# Input:
#    MtrlPar_s   Steel material parameters               [class]
#
# Output:
#    sigma_perp  Capacity perpendicular                  [float, MPa]
#    sigma_i     Capacity                                [float, MPa]
#
# ------------------------------------------------------------------------------
def WeldResistance(MtrlPar_s):

    # Correlation factor beta_w [-] and partial safety factor gamma_M2 [-]:
    if MtrlPar_s.SteelType == 'Stainless':
        beta_w = 1.0                        # SS-EN 1993-1-4 Section 6.3 (1)
        gamma_M2 = 1.25                     # SS-EN 1993-1-4 Section 5.1
    else:
        beta_w = MtrlPar_s.beta_w           # SS-EN 1993-1-8 Table 4.1
        gamma_M2 = 1.25                     # SS-EN 1993-2 Section 6.1

    # Capacity:
    sigma_perp = 0.9*MtrlPar_s.fu/gamma_M2
    sigma_i = MtrlPar_s.fu/(gamma_M2*beta_w)

    return (sigma_perp, sigma_i)
```

```
# ------------------------------------------------------------------------------
# 12. Design Shear Stud Resistance
# Input:
#    MtrlPar_s    Steel material parameters                    [class]
#    MtrlPar_c    Concrete material parameters                 [class]
#    d            Diameter of stud, between 16 and 25 mm       [float, mm]
#    hsc          Height of stud                               [float, mm]
#
# Output:
#    PRd          Design shear stud resistance                 [float, N]
#
# ------------------------------------------------------------------------------
def ShearStuds(MtrlPar_s, MtrlPar_c, d, hsc):
 # Design shear resistance of one shear stud (SS-EN 1994-2)
    # Partial factor [-] SS-EN 1994-2 6.6.3.1 (1):
    gamma_v = 1.25

    # Alpha [-]
    if hsc/d >= 3 and hsc/d <= 4:
        alpha = 0.2*(hsc/d+1)
    else:
        alpha = 1

    # Design resistance smallest of Prd_s and Prd_c [N]:
    fu = min(MtrlPar_s.fu, 500) # SS-EN 1994-2 6.6.3.1

    PRd_s = (0.8*fu*math.pi*d**2/4)/gamma_v
    PRd_c = (0.29*alpha*d**2*math.sqrt(MtrlPar_c.fck*MtrlPar_c.Ecm*1e3))\
            /gamma_v

    PRd = min(PRd_s, PRd_c)

    return PRd

# ------------------------------------------------------------------------------
# 13. Fatigue
# Input:
#    DetailCategory      Detail category of desired weld          [float, MPa]
#    ks                  Sizeffect                                [float, -]
#    Method              'DamageTolerant' or 'SafeLife'           [object, -]
#    Consequence         Consequence of failure, 'High' or 'Low'  [object, -]
#
# Output:
#    FatigueResistance   Fatigue resistance of the detail         [float, MPa]
#
# ------------------------------------------------------------------------------
def Fatigue(DetailCategory, ks, Method, Consequence):

 # Definition of the partial safety factor gamma_Mf [-]:
    if Method == 'DamageTolerant' and Consequence == 'Low':
        gamma_Mf = 1.0
    if Method == 'DamageTolerant' and Consequence == 'High':
        gamma_Mf = 1.15
    if Method == 'SafeLife' and Consequence == 'Low':
        gamma_Mf = 1.15
    if Method == 'SafeLife' and Consequence == 'High':
        gamma_Mf = 1.35

 # Capacity [MPa]:
    FatigueResistance = ks * DetailCategory /gamma_Mf

    return FatigueResistance

# ------------------------------------------------------------------------------
# 14. Bending Moment Capacity of Composite Beam
#
# Input:
#   MtrlPar_c    Material parameters of concrete          [class]
#   MtrlPar_s    Material parameters of steel             [class]
```

```
#   MtrlPar_re  Material parameters of reinforcement    [class]
#
# Output:
#   fyd         Design yield strength, steel           [float, MPa]
#   fcd         Design compressive strength, concrete  [float, MPa]
#   fctd        Design tensile strength, concrete      [float, MPa]
#   fsd         Design reinforcement capacity          [float, MPa]

# ------------------------------------------------------------------------
def CompositeMomentCapacity(MtrlPar_c, MtrlPar_s, MtrlPar_re):
    gammaM0 = 1.
    gammaC = 1.5
    gammaS = 1.15

    # Steel capacity
    fyd = MtrlPar_s.fy/gammaM0

    # Concrete capacity
    fcd = MtrlPar_c.fck/gammaC
    fctd = MtrlPar_c.fctk_005/gammaC

    # Reinforcement capacity
    fsd = MtrlPar_re.fyk/gammaS

    return(fyd, fcd, fctd, fsd)

# ------------------------------------------------------------------------
# 15. Buckling (general)
#
# Input:
#   MtrlPar         Material parameters                 [class]
#   Lcr             Buckling length                     [float, mm]
#   A               Cross-section area                  [float, mm^2]
#   h               Cross-section height                [float, mm]
#   b               Flange width (compressed)           [float, mm]
#   Iz              Minor axis inertia                  [float, mm^4]
#
# Output:
#   chi             Buckling reduction factor           [float, -]
# ------------------------------------------------------------------------
def Buckling(MtrlPar, Lcr, A, h, b, Iz):

    # Critical buckling load (Euler)
    Ncr = math.pi**2*MtrlPar.Es*10**3*Iz/(Lcr**2)

    # Slenderness parameter (SS-EN 1993-1-1 6.3.1.2)
    Lambda = math.sqrt(A*MtrlPar.fy/Ncr)

    # Buckling reduction factor
    # (SS-EN 1993-1-1 6.3.1.2)
    if Lambda <= 0.2:
        chi = 1.
    else:
        if h/b > 1.2:
            alpha = 0.34
        else:
            alpha = 0.49
        phi = 0.5*(1+alpha*(Lambda-0.2)+Lambda**2)
        chi = min(1/(phi+math.sqrt(phi**2-Lambda**2)),1.)

    return chi


# ------------------------------------------------------------------------
# 16. Minimum Reinforcement In Longitudinal Direction
#
# Input:
#   Ac_eff          Effective Concrete Area             [float, mm^2]
#   hc              Concrete Deck Thickness             [float, mm]
#   MaterialPar_c   Material Parameters Concrete        [class]
```

```
#   MaterialPar_re  Material Parameters Reinforcement       [class]
#   tp              Centre of gravity of composite section  [float, mm]
#   tp_c            Centre of gravity of concrete section   [float, mm]
#   d               Rebar diameter                          [float, mm]
#
# Output:
#   num             Number of bars in Long Direction        [float, mm^2]
# ------------------------------------------------------------------------
def MinReinforcement(Ac_eff, hc, MaterialPar_c, MaterialPar_re, tp, tp_c, d):
    # Partial Safety Factor:
    gamma_s = 1.15

    # Distance between centre of gravity of concrete and composite section [mm]
    z0 = tp - tp_c

    # k - koefficients (SS-EN 1994-2 7.4.2):
    k = 0.8
    ks = 0.9
    kc = min(1/(1+hc/(2*z0))+0.3,1.0)

    # Calculation of minimum reinforcement, [mm^2]:
    As_min = k * ks * kc*MaterialPar_c.fctm*Ac_eff/MaterialPar_re.fyk/gamma_s

    # Number of bars
    A_d = (d/2)**2*math.pi
    num = round(As_min/A_d)+1
    As_min = A_d*num

    return (num)
```

C.6 Module: StructuralAnalysisFunctions

```
# Master's Thesis
#
# Design Optimization of Composite Road Bridges using Genetic Algorithms
# -Corrugated Web Stainless Steel Girders versus Flat Web Carbon Steel Girders
#
# Cecilia Hallgren & Vilma Johansson
# June 2022
#
# ------------------------------------------------------------------------------
# MODULE OF STRUCTURAL ANALYSIS FUNCTIONS
#
# Contents
#   0. Input Modules
#   1. psi: Stress Distribution as Input for CSC4 Width Reduction
#   2. Influence Line Diagram of Simply Supported Beam (with Overhang)
#   3. Serviceability Limit State (Deflection)
#   4. Ultimate Limit State (Bending Moment and Shear)
#   5. Ultimate Limit State (Welds)
#   6. Lambda-method (Fatigue of welds)
#   7. Fatigue Limit State
#   8. Ultimate Limit State (Horizontal Loads)
#
# ------------------------------------------------------------------------------

# 0. Input Modules
import numpy as np
import math
import LoadFunctions as load

# ------------------------------------------------------------------------------
# 1. psi: Stress Distribution as Input for CSC4 Width Reduction
# Input:
#   CSC_w           Cross-section class of web                  [integer, -]
#   CSC_fo          Cross-section class of upper flange         [integer, -]
#   hw              Height of web                               [float, mm]
#   tp              Neutral axis from top of web down           [float, mm]
#
# Output:
#   psi_w           Stress distribution of web                  [float, -]
#   psi_fo          Stress distribution of flange               [float, -]
#
# ------------------------------------------------------------------------------

def CSC4psi(CSC_w, CSC_fo, hw, tp):
    # Elastic analysis and the same yield strength for all steel parts assumed!

    # Web (SS EN 1993-1-5 Table 4.1)
    if CSC_w < 4:
        psi_w = 0
    else:
        # Similar triangles gives:
        sigma1 = 1. # magnitude not relevant, only relation between stresses
        sigma2 = (hw-tp)*sigma1/tp
        psi_w = max(sigma2/-sigma1,-3.)

    # Flange
    if CSC_fo < 4:
        psi_fo = 0
    else:
        # Uniform stress along flange assumed (SS EN 1993-1-5 Table 4.2)
        psi_fo = 1.

    return [psi_w, psi_fo]

# ------------------------------------------------------------------------------
# 2. Influence Line Diagram of Simply Supported Beam (with Overhang)
```

```
# Input:
#   L           Length of beam                                      [lenght unit]
#   SubDiv      Sub division length,                                [lenght unit]
#               chosen so that N = L/SubDiv is an integer
#   x           Point on beam                                       [lenght unit]
#   L1          Overhang left side, if no overhang L1=0             [lenght unit]
#   L2          Overhang right side, if no overhang L2=0            [lenght unit]
#
# Output:
#   ILD_M       Moment influence line diagram matrix (N+1xN+1)      [lenght unit]
#   ILD_V       Shear influence line diagram matrix (N+1xN+1)       [-]
#
# with N = L/SubDiv
# each row represent one section position
# each column represent one load position
# to calculate moment, M: multiply with the value of the point load
# if several loads are added, the largest need to be applied at x
# ------------------------------------------------------------------------------
def ILD_Simply(L, SubDiv, x, L1, L2):

    N = round(L/SubDiv)
    N1 = round(L1/SubDiv)
    N2 = round(L2/SubDiv)

    if L/SubDiv == N and L1/SubDiv == N1 and L2/SubDiv == N2:
        # Moment ILD (only magnitude considered)
        ILD_M = np.empty(N+N1+N2+1)
        Y = np.linspace(-N1,N+N2,N+N1+N2+1)
        for i in range (N+N1+N2+1):
            y = Y[i]*SubDiv
            if y <= x:
                ILD_M[i] = abs((L-y)*x/L-(x-y))
            if y > x:
                ILD_M[i] = abs((L-y)*x/L)

        # Shear ILD (only total magnitude considered)
        ILD_V = np.empty(N+N1+N2+1)
        for i in range (N+N1+N2+1):
            y = Y[i]*SubDiv
            if y < x:
                ILD_V[i] = abs(-y/L)
            if y == x:
                ILD_V[i] = 1.
            if y > x:
                ILD_V[i] = abs((L-y)/L)

        return (ILD_M, ILD_V)

    else:
        raise Exception('L/SubDiv does not retrun even number of subdivisons!')

# ------------------------------------------------------------------------------
# 3. Serviceability Limit State (Deflection)
# Input:
#   MtrlPar_s       Material parameters of steel                [class]
#   G_SelfWeight    Self-weight of the structure, mean value    [float, N/mm]
#   G_pave          Self-weight of concrete cover, mean value   [float, N/mm]
#   Fcs             Shrinkage force                             [float, N]
#   Ftemp_t         Positive temperature force, mean value      [float, N]
#   Ftemp_c         Negative temperature force, mean value      [float, N]
#   Traffic         If traffic load, 'Yes' or 'No'              [object, -]
#   L               Length of bridge                            [float, mm]
#   a               Smallest distance from edge to point load   [float, mm]
#   I_long          Moment of inertia long term                 [float, mm^4]
#   I_short         Moment of inertia short term                [float, mm^4]
#   I_cs            Moment of inertia shrinkage                 [float, mm^4]
#   SubDiv          Sub division length                         [float, mm]
#   Bs              C-C distance between crossbeams             [float, mm]
#   B               Width of bridge deck                        [float, mm]
#   A               Composite area                              [float, mm^2]
```

```
#   tp_cs           Neutral axis composite section, shrinkage   [float, mm]
#   tp_c_cs         Neutral axis concrete section, shrinkage    [float, mm]
#   tp_short        Neutral axis composite section, short term  [float, mm]
#   tp_s_short      Neutral axis steel section, short term      [float, mm]
#   Wo_long         Average sectional modulus long term, top    [float, mm^3]
#   Wu_long         Average sectional modulus long term, bottom [float, mm^3]
#   Wo_cs           Average sectional modulus shrinkage, top    [float, mm^3]
#   Wu_cs           Average sectional modulus shrinkage, bottom [float, mm^3]
#   Wo_short        Average sectional modulus short term, top   [float, mm^3]
#   Wu_short        Average sectional modulus short term, bottom[float, mm^3]
#
# Output:   Delta           Maximum deflection                  [float, mm]
#
# ------------------------------------------------------------------------------
# ------------------------------------------------------------------------------
def SLS(MtrlPar_s, GSelfWeight, GPavement, Fcs, FTemp_t, FTemp_c, Traffic, L,
        a, I_long, I_short, I_cs, SubDiv, B, Bs, A, tp_cs , tp_c_cs, tp_short,
        tp_s_short ,Wo_long, Wu_long, Wo_cs, Wu_cs, Wo_short, Wu_short):

# For serviceable limit state, the frequent load combination (Eq. 6.15b) is
# used for calculation of deflection

    # Values of the upper and lower value (cc=concrete cover)
    # SS-EN 1991-1-1 section 5.2.3 (4)
    sup=1.0
    sup_cc=1.1

    # Load combination - Servicable Limit State (SLS)
    gamma_G = 1.0
    gamma_Q = 1.0
    gamma_acc = 0.6
    gamma_sh = 1.0
    psi1_traDis = 0.75
    psi1_traPoi = 0.4
    psi2_traffic = 0
    psi1_acc=0.75
    psi2_acc=0.
    psi1_temp=0.6
    psi2_temp=0.5

    # CALCULATION OF MOMENT (global)
    x = L/2

    # Self-weight
    MEk_SelfWeight = GSelfWeight/2*(L*x-x**2)
    MEk_Pave = GPavement/2*(L*x-x**2)

    # Shrinkage load
    MEk_cs = Fcs*(tp_cs-tp_c_cs)

    # Acceleration/braking load
    if Traffic == 'yes':
        Qacc = load.AccBraLoad(L)
        MEK_Acc = Qacc*tp_short
    if Traffic == 'no':
        MEK_Acc = 0

    # Temperature load
    MEk_Temp_pos = FTemp_t*abs(tp_s_short-tp_short)

    # Traffic load
    if Traffic == 'yes':
        [qk, Qk, PC, w, C] = load.TrafficLoadLM1()
        n = B/2/w # number of lanes that could possibly fit half of bridge
        if n < 1:
            Ptraffic = 0
            Qtraffic = 0
            PtrafficT = 0
            QtrafficT = 0
        if n == 1:
```

```
    Ptraffic = Qk[0]
    Qtraffic = qk[0]*w
    # Transverse moment
    Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)
    Md = qk[0]*w*(w/2)
if n > 1 and n < 2:
    Ptraffic = Qk[0]
    Qtraffic = qk[0]*w+qk[1]*(B/2-500-w)
    # Transverse moment
    Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)
    Md = qk[0]*w*(B/2-500-w/2)+qk[1]*((B/2-500-w)/2)**2
if n == 2:
    Ptraffic = Qk[0]+Qk[1]
    Qtraffic = qk[0]*w+qk[1]*w
    # Transverse moment
    Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)+\
         Qk[1]*(B/2-500-C-1e3)+Qk[1]*(B/2-500-C-1e3-C)
    Md = qk[0]*w*(B/2-500-w/2)+qk[1]*w*(w/2)
if n > 2 and n < 3:
    Ptraffic = Qk[0]+Qk[1]
    Qtraffic = qk[0]*w+qk[1]*w+qk[2]*(B/2-500-2*w)
    # Transverse moment
    Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)+\
         Qk[1]*(B/2-500-C-1e3)+Qk[1]*(B/2-500-C-1e3-C)
    Md = qk[0]*w*(B/2-500-w/2)+qk[1]*w*(B/2-500-w)+\
         qk[2]*((B/2-500-2*w)/2)**2
if n == 3:
    Ptraffic = Qk[0]+Qk[1]+Qk[2]
    Qtraffic = qk[0]*w+qk[1]*w+qk[2]*w
    # Transverse moment
    Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)+\
         Qk[1]*(B/2-500-C-1e3)+Qk[1]*(B/2-500-C-1e3-C)+\
         Qk[2]*(B/2-500-C-1e3-C-1e3)+Qk[2]*(B/2-500-C-1e3-C-1e3-C)
    Md = qk[0]*w*(B/2-500-w/2)+qk[1]*w*(B/2-500-w)+qk[2]*w*(w/2)
if n > 3:
    Ptraffic = Qk[0]+Qk[1]+Qk[2]
    Qtraffic = qk[0]*w+qk[1]*w+qk[2]*w+qk[3]*(B/2-500-3*w)
    # Transverse moment
    Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)+\
         Qk[1]*(B/2-500-C-1e3)+Qk[1]*(B/2-500-C-1e3-C)+\
         Qk[2]*(B/2-500-C-1e3-C-1e3)+Qk[2]*(B/2-500-C-1e3-C-1e3-C)
    Md = qk[0]*w*(B/2-500-w/2)+qk[1]*w*(B/2-500-w)+\
         qk[3]*((B/2-500-3*w)/2)**2

# Distributed load
MEk_DisTra = Qtraffic/2*(L*x-x**2)

# Point load
[ILD_M, ILD_V] = ILD_Simply(L, SubDiv, x, 0, 0)
i = int(x/SubDiv)
isubC = int((x-C)/SubDiv)
iaddC = int((x+C)/SubDiv)
isubC2 = int((x-C/2)/SubDiv)
iaddC2 = int((x+C/2)/SubDiv)
MEk_PoiTra_left = Ptraffic*(ILD_M[i]+ILD_M[isubC])
MEk_PoiTra_right = Ptraffic*(ILD_M[i]+ILD_M[iaddC])
MEk_PoiTra_eq = Ptraffic*(ILD_M[isubC2]+ILD_M[iaddC2])
MEk_PoiTra = max(MEk_PoiTra_left, MEk_PoiTra_right, MEk_PoiTra_eq)

# Distributed load transverse moment contribution
QtrafficT = Md/(Bs)
MEk_DisTraT = QtrafficT/2*(L*x-x**2)

# Point load moment transverse contribution
PtrafficT = Mp/(Bs)
MEk_PoiTraT_left = PtrafficT*(ILD_M[i]+ILD_M[isubC])
MEk_PoiTraT_right = PtrafficT*(ILD_M[i]+ILD_M[iaddC])
MEk_PoiTraT_eq = PtrafficT*(ILD_M[isubC2]+ILD_M[iaddC2])
MEk_PoiTraT = max(MEk_PoiTraT_left, MEk_PoiTraT_right, MEk_PoiTraT_eq)
```

```
    if Traffic == 'no':
        MEk_DisTra = 0
        MEk_PoiTra = 0
        MEk_PoiTraT = 0
        MEk_DisTraT = 0

    # For stainless steel, the secant modulus is used for deflection
    # calculations in SLS.
    if MtrlPar_s.SteelType == 'Stainless':
        sigmaG_o = MEk_SelfWeight/Wo_long
        sigmaG_u = MEk_SelfWeight/Wu_long

        sigmaGp_o = MEk_Pave/Wo_long
        sigmaGp_u = MEk_Pave/Wu_long

        sigmaEk_cs_o = MEk_cs/Wo_cs
        sigmaEk_cs_u = MEk_cs/Wu_cs
        sigmaEk_Fcs = Fcs/A

        sigmaEk_Acc_pos_o = MEK_Acc/Wo_short       # compression
        sigmaEk_Acc_pos_u = MEK_Acc/Wu_short       # tension

        sigmaEk_Temp_pos_o = MEk_Temp_pos/Wo_short # compression
        sigmaEk_Temp_pos_u = MEk_Temp_pos/Wu_short # tension

        sigmaEk_DisTra_o = MEk_DisTra/Wo_short
        sigmaEk_DisTra_u = MEk_DisTra/Wu_short

        sigmaEk_PoiTra_o = MEk_PoiTra/Wo_short
        sigmaEk_PoiTra_u = MEk_PoiTra/Wu_short

        sigmaEk_PoiTraT_o = MEk_PoiTraT/Wo_short
        sigmaEk_PoiTraT_u = MEk_PoiTraT/Wu_short

        sigmaEk_DisTraT_o = MEk_DisTraT/Wo_short
        sigmaEk_DisTraT_u = MEk_DisTraT/Wu_short

        # Combination Eq. 6.15b
        sigmaEd_o = sup * gamma_G * sigmaG_o + sup_cc * gamma_G * sigmaGp_o+\
                    gamma_sh * sup * sigmaEk_cs_o +\
                    gamma_sh * sup * sigmaEk_Fcs+\
                    gamma_Q * psi2_temp * sigmaEk_Temp_pos_o+\
                    gamma_Q * gamma_acc * psi2_acc * sigmaEk_Acc_pos_o+\
                    gamma_Q * psi1_traDis * sigmaEk_DisTra_o+\
                    gamma_Q * psi1_traPoi * sigmaEk_PoiTra_o+\
                    gamma_Q * psi1_traDis * sigmaEk_DisTraT_o+\
                    gamma_Q * psi1_traPoi * sigmaEk_PoiTraT_o

        sigmaEd_u = sup * gamma_G * sigmaG_u + sup_cc * gamma_G * sigmaGp_u+\
                    gamma_sh * sup * (sigmaEk_cs_u + sigmaEk_Fcs)+\
                    gamma_Q * psi2_temp * sigmaEk_Temp_pos_u+\
                    gamma_Q * gamma_acc * psi2_acc * sigmaEk_Acc_pos_u+\
                    gamma_Q * psi1_traDis * sigmaEk_DisTra_u +\
                    gamma_Q * psi1_traPoi * sigmaEk_PoiTra_u+\
                    gamma_Q * psi1_traDis * sigmaEk_DisTraT_u+\
                    gamma_Q * psi1_traPoi * sigmaEk_PoiTraT_u

        # Secant Modulus of Elasticity
        n = 8.
        E = MtrlPar_s.Es*1e3 # MPa
        Esc = E / (1+0.002*(E/sigmaEd_o)*(sigmaEd_o/MtrlPar_s.fy)**n)
        Est = E / (1+0.002*(E/sigmaEd_u)*(sigmaEd_u/MtrlPar_s.fy)**n)

        Es = (Esc + Est)/2*1e-3 # GPa

    if MtrlPar_s.SteelType == 'Carbon':
        Es = MtrlPar_s.Es # GPa

    # CALCULATION OF DEFLECTION:
    # Self-weight [N/mm]:
```

```
    Gdselfweight = sup * gamma_G * GSelfWeight

    # Self-weight [N/mm]:
    Gdpave = sup_cc * gamma_G * GPavement

    # Traffic [N and N/mm]:
    qd_main = gamma_Q * psi1_traPoi * (Ptraffic+PtrafficT)
    Qd_main = gamma_Q * psi1_traDis * (Qtraffic+QtrafficT)

    # Deflection traffic
    deltaTra = 5*L**4*(Qd_main)/(384*Es*1e3*I_short)\
               +(qd_main*a*L**2/(48*Es*1e3*I_short))*(3-4*a**2/L**2)\

    # Delta permanent loads
    deltaPerm = 5*L**4*(Gdselfweight+Gdpave)/(384*Es*1e3*I_long)

    return (deltaTra, deltaPerm)

# ------------------------------------------------------------------------
# 4. Ultimate Limit State (Bending Moment and Shear)
# Input:
#    SafetyClass     Specified safety class, 1, 2 or 3   [integer, -]
#    GSelfWeight     Self-weight of beam                 [float, N/mm]
#    GPavement       Self-weight of pavement             [float, N/mm]
#    Fcs             Shrinkage force                     [float, N]
#    FTemp_t         Positive temperature force          [float, N]
#    FTemp_c         Negative temperature force          [float, N]
#    Traffic         Include traffic load, 'yes' or 'no' [object, -]
#    L               Length of bridge                    [float, mm]
#    SubDiv          Sub division length                 [float, mm]
#    x_start         Start coordinate of segment         [float, mm]
#    x_end           End coordinate of segment           [float, mm]
#    B               Width of bridge deck                [float, mm]
#    Bs              C-C distance between crossbeams     [float, mm]
#    H               Total height of bridge              [float, mm]
#    SecPar_long     Long term section parameters        [class]
#    SecPar_short    Short term section parameters       [class]
#    SecPar_cs       Shrinkage section parameters        [class]
#
# Note that traffic load input are called on in the function via the module
# LoadFunctions.
#
# Output:
#    sigmaEd_ot      Design stress at top fibre, tensile [float, MPa]
#    sigmaEd_oc      Design stress at top fibre, comp.   [float, MPa]
#    sigmaEd_u       Design stress at bottom fibre       [float, MPa]
#    VEd             Design shear load                   [float, N]
#    VEd_studs       Design shear at interface           [float, N/mm]
#    MEd             Design moment                       [float, Nmm]
# ------------------------------------------------------------------------
def ULS(SafetyClass, GSelfWeight, GPavement, Fcs, FTemp_t, FTemp_c, Traffic,
        L, SubDiv, x_start, x_end, B, Bs, H, Ac, SecPar_long, SecPar_short,
        SecPar_cs):

# Ultimate limit state is divided into two phases, construction and service.
# To calculate the design loads, 2 different equations are used: 6.10a and
# 6.10b. For strength checks, the worst of Eq. 6.10a and 6.10b should be used
# (SS-EN 1990 Annex A2).

  # Determination of the factor gamma_d depending on safety class
  # TSFS 2018:57 Chapter 2:8
    if SafetyClass == 1:
       gamma_d= 0.83
    if SafetyClass == 2:
       gamma_d=0.91
    if SafetyClass == 3:
       gamma_d=1.0
    else:
       gamma_d=1.1
```

```
  # Values of the upper and lower value (cc=concrete cover)
  # SS-EN 1991-1-1 section 5.2.3 (4)
    sup=1.0
    sup_cc=1.1

# CALCULATION OF MOMENT (global)
    x = x_end # worse case closest to mid

    # Self-weight
    MEk_SelfWeight = GSelfWeight/2*(L*x-x**2)
    sigmaG_c = MEk_SelfWeight/SecPar_long.Wc/SecPar_long.nL
    sigmaG_o = MEk_SelfWeight/SecPar_long.Wo
    sigmaG_u = MEk_SelfWeight/SecPar_long.Wu

    MEk_Pave = GPavement/2*(L*x-x**2)
    sigmaGp_c = MEk_Pave/SecPar_long.Wc/SecPar_long.nL
    sigmaGp_o = MEk_Pave/SecPar_long.Wo
    sigmaGp_u = MEk_Pave/SecPar_long.Wu

    # Shrinkage load
    MEk_cs = Fcs*(SecPar_cs.tp-SecPar_cs.tp_c)

    # Positive moment due to compression force acting on the composite section
    sigmaEk_cs_c = MEk_cs/SecPar_cs.Wc/SecPar_cs.nL
    sigmaEk_cs_o = MEk_cs/SecPar_cs.Wo
    sigmaEk_cs_u = MEk_cs/SecPar_cs.Wu

    # Tension force acting on concrete section due to the steel restraint
    sigmaEk_Fcs_c = Fcs/Ac - Fcs/SecPar_cs.A/SecPar_cs.nL
    sigmaEk_Fcs_o = -Fcs/SecPar_cs.A
    sigmaEk_Fcs_u = -Fcs/SecPar_cs.A

    # Acceleration/braking load
    if Traffic == 'yes':
        Qacc = load.AccBraLoad(L)
        MEK_Acc = Qacc*SecPar_short.tp
    if Traffic == 'no':
        Qacc = 0
        MEK_Acc = 0
    sigmaEk_Acc_pos_c = MEK_Acc/SecPar_short.Wc/SecPar_short.nL  # compression
    sigmaEk_Acc_neg_c = -MEK_Acc/SecPar_short.Wc/SecPar_short.nL # tension
    sigmaEk_Acc_pos_o = MEK_Acc/SecPar_short.Wo                  # compression
    sigmaEk_Acc_pos_u = MEK_Acc/SecPar_short.Wu                  # tension

    # Temperature load
    MEk_Temp_pos = FTemp_t*abs(SecPar_short.tp_s-SecPar_short.tp)
    MEk_Temp_neg = FTemp_c*abs(SecPar_short.tp_s-SecPar_short.tp)
    sigmaEk_Temp_pos_c = MEk_Temp_pos/SecPar_short.Wc/SecPar_short.nL#compres
    sigmaEk_Temp_neg_c = MEk_Temp_neg/SecPar_short.Wc/SecPar_short.nL#tension
    sigmaEk_Temp_pos_o = MEk_Temp_pos/SecPar_short.Wo                #compres
    sigmaEk_Temp_pos_u = MEk_Temp_pos/SecPar_short.Wu                #tension

    # Traffic load
    if Traffic == 'yes':
        [qk, Qk, PC, w, C] = load.TrafficLoadLM1()
        n = B/2/w # number of lanes that could possibly fit half of bridge
        if n < 1:
            Ptraffic = 0
            Qtraffic = 0
            PtrafficT = 0
            QtrafficT = 0
        if n == 1:
            Ptraffic = Qk[0]
            Qtraffic = qk[0]*w
            # Transverse moment
            Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)
            Md = qk[0]*w*(w/2)
        if n > 1 and n < 2:
            Ptraffic = Qk[0]
            Qtraffic = qk[0]*w+qk[1]*(B/2-500-w)
```

```
            # Transverse moment
            Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)
            Md = qk[0]*w*(B/2-500-w/2)+qk[1]*((B/2-500-w)/2)**2
        if n == 2:
            Ptraffic = Qk[0]+Qk[1]
            Qtraffic = qk[0]*w+qk[1]*w
            # Transverse moment
            Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)+\
                 Qk[1]*(B/2-500-C-1e3)+Qk[1]*(B/2-500-C-1e3-C)
            Md = qk[0]*w*(B/2-500-w/2)+qk[1]*w*(w/2)
        if n > 2 and n < 3:
            Ptraffic = Qk[0]+Qk[1]
            Qtraffic = qk[0]*w+qk[1]*w+qk[2]*(B/2-500-2*w)
            # Transverse moment
            Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)+\
                 Qk[1]*(B/2-500-C-1e3)+Qk[1]*(B/2-500-C-1e3-C)
            Md = qk[0]*w*(B/2-500-w/2)+qk[1]*w*(B/2-500-w)+\
                 qk[2]*((B/2-500-2*w)/2)**2
        if n == 3:
            Ptraffic = Qk[0]+Qk[1]+Qk[2]
            Qtraffic = qk[0]*w+qk[1]*w+qk[2]*w
            # Transverse moment
            Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)+\
                 Qk[1]*(B/2-500-C-1e3)+Qk[1]*(B/2-500-C-1e3-C)+\
                 Qk[2]*(B/2-500-C-1e3-C-1e3)+Qk[2]*(B/2-500-C-1e3-C-1e3-C)
            Md = qk[0]*w*(B/2-500-w/2)+qk[1]*w*(B/2-500-w)+qk[2]*w*(w/2)
        if n > 3:
            Ptraffic = Qk[0]+Qk[1]+Qk[2]
            Qtraffic = qk[0]*w+qk[1]*w+qk[2]*w+qk[3]*(B/2-500-3*w)
            # Transverse moment
            Mp = Qk[0]*(B/2-500)+Qk[0]*(B/2-500-C)+\
                 Qk[1]*(B/2-500-C-1e3)+Qk[1]*(B/2-500-C-1e3-C)+\
                 Qk[2]*(B/2-500-C-1e3-C-1e3)+Qk[2]*(B/2-500-C-1e3-C-1e3-C)
            Md = qk[0]*w*(B/2-500-w/2)+qk[1]*w*(B/2-500-w)+\
                 qk[3]*((B/2-500-3*w)/2)**2

        # Distributed load
        MEk_DisTra = Qtraffic/2*(L*x-x**2)

        # Point load
        [ILD_M, ILD_V] = ILD_Simply(L, SubDiv, x, 0, 0)
        i = int(x/SubDiv)
        isubC = int((x-C)/SubDiv)
        iaddC = int((x+C)/SubDiv)
        isubC2 = int((x-C/2)/SubDiv)
        iaddC2 = int((x+C/2)/SubDiv)
        MEk_PoiTra_left = Ptraffic*(ILD_M[i]+ILD_M[isubC])
        MEk_PoiTra_right = Ptraffic*(ILD_M[i]+ILD_M[iaddC])
        MEk_PoiTra_eq = Ptraffic*(ILD_M[isubC2]+ILD_M[iaddC2])
        MEk_PoiTra = max(MEk_PoiTra_left, MEk_PoiTra_right, MEk_PoiTra_eq)

        # Distributed load transverse moment contribution
        QtrafficT = Md/(Bs)
        MEk_DisTraT = QtrafficT/2*(L*x-x**2)

        # Point load moment transverse contribution
        PtrafficT = Mp/(Bs)
        MEk_PoiTraT_left = PtrafficT*(ILD_M[i]+ILD_M[isubC])
        MEk_PoiTraT_right = PtrafficT*(ILD_M[i]+ILD_M[iaddC])
        MEk_PoiTraT_eq = PtrafficT*(ILD_M[isubC2]+ILD_M[iaddC2])
        MEk_PoiTraT = max(MEk_PoiTraT_left, MEk_PoiTraT_right, MEk_PoiTraT_eq)

    if Traffic == 'no':
        MEk_DisTra = 0
        MEk_PoiTra = 0
        MEk_PoiTraT = 0
        MEk_DisTraT = 0

    sigmaEk_DisTra_c = MEk_DisTra/SecPar_short.Wc/SecPar_short.nL
    sigmaEk_DisTra_o = MEk_DisTra/SecPar_short.Wo
```

```
    sigmaEk_DisTra_u = MEk_DisTra/SecPar_short.Wu

    sigmaEk_PoiTra_c = MEk_PoiTra/SecPar_short.Wc/SecPar_short.nL
    sigmaEk_PoiTra_o = MEk_PoiTra/SecPar_short.Wo
    sigmaEk_PoiTra_u = MEk_PoiTra/SecPar_short.Wu

    sigmaEk_PoiTraT_c = MEk_PoiTraT/SecPar_short.Wc/SecPar_short.nL
    sigmaEk_PoiTraT_o = MEk_PoiTraT/SecPar_short.Wo
    sigmaEk_PoiTraT_u = MEk_PoiTraT/SecPar_short.Wu

    sigmaEk_DisTraT_c = MEk_DisTraT/SecPar_short.Wc/SecPar_short.nL
    sigmaEk_DisTraT_o = MEk_DisTraT/SecPar_short.Wo
    sigmaEk_DisTraT_u = MEk_DisTraT/SecPar_short.Wu


# CALCULATION OF SHEAR (global)
    x = x_start # worse case closest to end

    VEk_SelfWeight=GSelfWeight*(L/2-x)
    VEk_Pave=GPavement*(L/2-x)

    # Traffic load
    if Traffic == 'yes':
        # Distributed load
        VEk_DisTra=Qtraffic*(L/2-x)

        # Point load
        [ILD_M, ILD_V] = ILD_Simply(L, SubDiv, x, 0, 0)
        VEk_PoiTra_left = Ptraffic*(ILD_V[i]+ILD_V[isubC])
        VEk_PoiTra_right = Ptraffic*(ILD_V[i]+ILD_V[iaddC])
        VEk_PoiTra_eq = Ptraffic*(ILD_V[isubC2]+ILD_V[iaddC2])
        VEk_PoiTra = max(VEk_PoiTra_left, VEk_PoiTra_right, VEk_PoiTra_eq)

        # Distributed load transverse moment contribution
        VEk_DisTraT=QtrafficT*(L/2-x)

        # Point load transverse moment contribution
        VEk_PoiTraT_left = PtrafficT*(ILD_V[i]+ILD_V[isubC])
        VEk_PoiTraT_right = PtrafficT*(ILD_V[i]+ILD_V[iaddC])
        VEk_PoiTraT_eq = PtrafficT*(ILD_V[isubC2]+ILD_V[iaddC2])
        VEk_PoiTraT = max(VEk_PoiTraT_left, VEk_PoiTraT_right, VEk_PoiTraT_eq)

    if Traffic == 'no':
        VEk_DisTra = 0
        VEk_PoiTra = 0
        VEk_PoiTraT = 0
        VEk_DisTraT = 0

# Load combination - Ultimate Limit State (ULS)
    gamma_G = 1.35
    gamma_Gb = 0.89
    gamma_Q = 1.5
    gamma_acc = 0.6
    gamma_sh = 1.0
    psi0_traDis = 0.75
    psi0_traPoi = 0.4
    psi0_acc=0.75
    psi0_temp=0.6

# Combination Eq. 6.10a
    sigmaEd_cta = gamma_d * sup * gamma_G * sigmaG_c +\
                  gamma_d * sup_cc * gamma_G * sigmaGp_c+\
                  gamma_sh * sup * sigmaEk_cs_c +\
                  gamma_sh * sup * sigmaEk_Fcs_c+\
                  gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_neg_c+\
                  gamma_d * gamma_Q * gamma_acc * psi0_acc*sigmaEk_Acc_neg_c+\
                  gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_c+\
                  gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_c+\
                  gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_c+\
                  gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_c
```

```
    sigmaEd_cca = gamma_d * sup * gamma_G * sigmaG_c +\
                  gamma_d * sup_cc * gamma_G * sigmaGp_c+\
                  gamma_sh * sup * sigmaEk_cs_c +\
                  gamma_sh * sup * sigmaEk_Fcs_c+\
                  gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_pos_c+\
                  gamma_d * gamma_Q * gamma_acc * psi0_acc*sigmaEk_Acc_pos_c+\
                  gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_c+\
                  gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_c+\
                  gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_c+\
                  gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_c

    sigmaEd_oa = gamma_d * sup * gamma_G * sigmaG_o +\
                 gamma_d * sup_cc * gamma_G * sigmaGp_o+\
                 gamma_sh * sup * sigmaEk_cs_o +\
                 gamma_sh * sup *sigmaEk_Fcs_o+\
                 gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_pos_o+\
                 gamma_d * gamma_Q * gamma_acc * psi0_acc * sigmaEk_Acc_pos_o+\
                 gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_o +\
                 gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_o+\
                 gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_o+\
                 gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_o

    sigmaEd_ua = gamma_d * sup * gamma_G * sigmaG_u +\
                 gamma_d * sup_cc * gamma_G * sigmaGp_u+\
                 gamma_sh * sup * sigmaEk_cs_u + \
                 gamma_sh * sup *sigmaEk_Fcs_u+\
                 gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_pos_u+\
                 gamma_d * gamma_Q * gamma_acc * psi0_acc * sigmaEk_Acc_pos_u+\
                 gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_u +\
                 gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_u+\
                 gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_u+\
                 gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_u

    VEda =  gamma_d * sup * gamma_G * VEk_SelfWeight +\
            gamma_d * sup_cc * gamma_G * VEk_Pave+\
            gamma_d * gamma_Q * psi0_traDis * VEk_DisTra +\
            gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTra+\
            gamma_d * gamma_Q * psi0_traDis * VEk_DisTraT+\
            gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTraT

    VEda_studs =  (gamma_d * sup * gamma_G * VEk_SelfWeight +\
                   gamma_d * sup_cc * gamma_G * VEk_Pave)\
                   *SecPar_long.Sc/SecPar_long.I +\
                  (gamma_d * gamma_Q * psi0_traDis * VEk_DisTra +\
                   gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTra+\
                   gamma_d * gamma_Q * psi0_traDis * VEk_DisTraT+\
                   gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTraT)\
                   *SecPar_short.Sc/SecPar_short.I +\
                   gamma_d * gamma_Q * gamma_acc * psi0_acc * Qacc/L +\
                  (gamma_d * gamma_Q * psi0_temp*(abs(FTemp_t)/SecPar_short.A
    +\
                   abs(MEk_Temp_pos)/SecPar_short.I*\
                   abs(SecPar_short.tp-SecPar_short.tp_s))*
                  (SecPar_short.A-Ac/SecPar_short.nL))/L

    MEda = gamma_d * sup * gamma_G * MEk_SelfWeight +\
           gamma_d * sup_cc * gamma_G * MEk_Pave+\
           gamma_sh * sup * MEk_cs+\
           gamma_d * gamma_Q *  gamma_acc * psi0_acc * MEK_Acc+\
           gamma_d * gamma_Q * psi0_temp * MEk_Temp_pos+\
           gamma_d * gamma_Q * psi0_traDis * MEk_DisTra+\
           gamma_d * gamma_Q * psi0_traPoi * MEk_PoiTra+\
           gamma_d * gamma_Q * psi0_traDis * MEk_DisTraT+\
           gamma_d * gamma_Q * psi0_traPoi * MEk_PoiTraT

# Combination Eq. 6.10b
    # Comb. 1 - Traffic load main load, temp and acceleration other load:
    sigmaEd_ctb1 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_c + \
                   gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_c+\
```

```
                gamma_sh * sup * sigmaEk_cs_c +\
                gamma_sh * sup * sigmaEk_Fcs_c+\
                gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_neg_c+\
                gamma_d * gamma_Q * gamma_acc*psi0_acc*sigmaEk_Acc_neg_c+\
                gamma_d * gamma_Q * sigmaEk_DisTra_c+\
                gamma_d * gamma_Q * sigmaEk_PoiTra_c+\
                gamma_d * gamma_Q * sigmaEk_DisTraT_c+\
                gamma_d * gamma_Q * sigmaEk_PoiTraT_c

sigmaEd_ccb1 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_c + \
                gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_c+\
                gamma_sh * sup * sigmaEk_cs_c +\
                gamma_sh * sup * sigmaEk_Fcs_c+\
                gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_pos_c+\
                gamma_d * gamma_Q * gamma_acc*psi0_acc*sigmaEk_Acc_pos_c+\
                gamma_d * gamma_Q * sigmaEk_DisTra_c+\
                gamma_d * gamma_Q * sigmaEk_PoiTra_c+\
                gamma_d * gamma_Q * sigmaEk_DisTraT_c+\
                gamma_d * gamma_Q * sigmaEk_PoiTraT_c

sigmaEd_ob1 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_o + \
               gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_o+\
               gamma_sh * sup * (sigmaEk_cs_o + sigmaEk_Fcs_o)+\
               gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_pos_o+\
               gamma_d * gamma_Q * gamma_acc*psi0_acc*sigmaEk_Acc_pos_o+\
               gamma_d * gamma_Q * sigmaEk_DisTra_o +\
               gamma_d * gamma_Q * sigmaEk_PoiTra_o+\
               gamma_d * gamma_Q * sigmaEk_DisTraT_o+\
               gamma_d * gamma_Q * sigmaEk_PoiTraT_o

sigmaEd_ub1 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_u + \
               gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_u+\
               gamma_sh * sup * (sigmaEk_cs_u + sigmaEk_Fcs_u)+\
               gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_pos_u+\
               gamma_d * gamma_Q * gamma_acc*psi0_acc*sigmaEk_Acc_pos_u+\
               gamma_d * gamma_Q * sigmaEk_DisTra_u +\
               gamma_d * gamma_Q * sigmaEk_PoiTra_u+\
               gamma_d * gamma_Q * sigmaEk_DisTraT_u+\
               gamma_d * gamma_Q * sigmaEk_PoiTraT_u

VEdb1 =  gamma_d * sup * gamma_G * gamma_Gb * VEk_SelfWeight +\
         gamma_d * sup_cc * gamma_G * gamma_Gb * VEk_Pave+\
         gamma_d * gamma_Q * VEk_DisTra +\
         gamma_d * gamma_Q * VEk_PoiTra+\
         gamma_d * gamma_Q * VEk_DisTraT+\
         gamma_d * gamma_Q * VEk_PoiTraT

VEdb1_studs = (gamma_d * sup * gamma_G * gamma_Gb * VEk_SelfWeight +\
               gamma_d * sup_cc * gamma_G * gamma_Gb * VEk_Pave)\
               *SecPar_long.Sc/SecPar_long.I +\
              (gamma_d * gamma_Q * VEk_DisTra +\
               gamma_d * gamma_Q * VEk_PoiTra+\
               gamma_d * gamma_Q * VEk_DisTraT+\
               gamma_d*gamma_Q*VEk_PoiTraT)*SecPar_short.Sc/SecPar_short.I
+\
               gamma_d * gamma_Q * gamma_acc * psi0_acc * Qacc/L +\
              (gamma_d * gamma_Q * psi0_temp*(abs(FTemp_t)/SecPar_short.A+\
               abs(MEk_Temp_pos)/SecPar_short.I*\
               abs(SecPar_short.tp-SecPar_short.tp_s))*
              (SecPar_short.A-Ac/SecPar_short.nL))/L

MEdb1 = gamma_d * sup * gamma_G * gamma_Gb * MEk_SelfWeight + \
        gamma_d * sup_cc * gamma_G * gamma_Gb * MEk_Pave+\
        gamma_sh * sup * MEk_cs+\
        gamma_d * gamma_Q * gamma_acc * psi0_acc * MEK_Acc+\
        gamma_d * gamma_Q * psi0_temp * MEk_Temp_pos+\
        gamma_d * gamma_Q * MEk_DisTra+\
        gamma_d * gamma_Q * MEk_PoiTra+\
        gamma_d * gamma_Q * MEk_DisTraT+\
        gamma_d * gamma_Q * MEk_PoiTraT
```

```
# Comb. 2 - Acceleratiom load main load, temp and traffic other load:
sigmaEd_ctb2 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_c + \
               gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_c+\
               gamma_sh * sup * sigmaEk_cs_c +\
               gamma_sh * sup * sigmaEk_Fcs_c+\
               gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_neg_c+\
               gamma_d * gamma_Q * gamma_acc * sigmaEk_Acc_neg_c+\
               gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_c+\
               gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_c+\
               gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_c+\
               gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_c

sigmaEd_ccb2 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_c + \
               gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_c+\
               gamma_sh * sup * sigmaEk_cs_c +\
               gamma_sh * sup * sigmaEk_Fcs_c+\
               gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_pos_c+\
               gamma_d * gamma_Q * gamma_acc * sigmaEk_Acc_pos_c+\
               gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_c+\
               gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_c+\
               gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_c+\
               gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_c

sigmaEd_ob2 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_o +\
              gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_o+\
              gamma_sh * sup * (sigmaEk_cs_o + sigmaEk_Fcs_o)+\
              gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_pos_o+\
              gamma_d * gamma_Q * gamma_acc * sigmaEk_Acc_pos_o+\
              gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_o +\
              gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_o+\
              gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_o+\
              gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_o

sigmaEd_ub2 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_u +\
              gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_u+\
              gamma_sh * sup * (sigmaEk_cs_u + sigmaEk_Fcs_u)+\
              gamma_d * gamma_Q * psi0_temp * sigmaEk_Temp_pos_u+\
              gamma_d * gamma_Q * gamma_acc * sigmaEk_Acc_pos_u+\
              gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_u +\
              gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_u+\
              gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_u+\
              gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_u

VEdb2 =  gamma_d * sup * gamma_G * gamma_Gb * VEk_SelfWeight + \
         gamma_d * sup_cc * gamma_G * gamma_Gb * VEk_Pave+\
         gamma_d * gamma_Q * psi0_traDis * VEk_DisTra +\
         gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTra+\
         gamma_d * gamma_Q * psi0_traDis * VEk_DisTraT+\
         gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTraT

VEdb2_studs =  (gamma_d * sup * gamma_G * gamma_Gb * VEk_SelfWeight+\
                gamma_d * sup_cc * gamma_G * gamma_Gb * VEk_Pave)\
                *SecPar_long.Sc/SecPar_long.I +\
               (gamma_d * gamma_Q * psi0_traDis * VEk_DisTra +\
                gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTra+\
                gamma_d * gamma_Q * psi0_traDis * VEk_DisTraT+\
                gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTraT)\
                *SecPar_short.Sc/SecPar_short.I +\
                gamma_d * gamma_Q * gamma_acc * Qacc/L +\
               (gamma_d * gamma_Q * psi0_temp*(abs(FTemp_t)/SecPar_short.A
+\
                abs(MEk_Temp_pos)/SecPar_short.I*\
                abs(SecPar_short.tp-SecPar_short.tp_s))*
               (SecPar_short.A-Ac/SecPar_short.nL))/L

MEdb2 = gamma_d * sup * gamma_G * gamma_Gb * MEk_SelfWeight +\
        gamma_d * sup_cc * gamma_G * gamma_Gb * MEk_Pave+\
        gamma_sh * sup * MEk_cs+\
        gamma_d * gamma_Q * gamma_acc * MEK_Acc+\
```

```
        gamma_d * gamma_Q * psi0_temp * MEk_Temp_pos+\
        gamma_d * gamma_Q * psi0_traDis * MEk_DisTra+\
        gamma_d * gamma_Q * psi0_traPoi * MEk_PoiTra+\
        gamma_d * gamma_Q * psi0_traDis * MEk_DisTraT+\
        gamma_d * gamma_Q * psi0_traPoi * MEk_PoiTraT

# Comb. 3 - Temp load main load, acceleration and traffic other load:
sigmaEd_ctb3 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_c +\
               gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_c+\
               gamma_sh * sup * sigmaEk_cs_o +\
               gamma_sh * sup * sigmaEk_Fcs_c+\
               gamma_d * gamma_Q * sigmaEk_Temp_neg_c+\
               gamma_d * gamma_Q * gamma_acc*psi0_acc*sigmaEk_Acc_neg_c+\
               gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_c+\
               gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_c+\
               gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_c+\
               gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_c

sigmaEd_ccb3 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_c +\
               gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_c+\
               gamma_sh * sup * sigmaEk_cs_c +\
               gamma_sh * sup * sigmaEk_Fcs_c+\
               gamma_d * gamma_Q * sigmaEk_Temp_pos_c+\
               gamma_d * gamma_Q * gamma_acc*psi0_acc*sigmaEk_Acc_pos_c+\
               gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_c+\
               gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_c+\
               gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_c+\
               gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_c

sigmaEd_ob3 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_o + \
              gamma_d* sup_cc * gamma_G * gamma_Gb * sigmaGp_o+\
              gamma_sh * sup * (sigmaEk_cs_o + sigmaEk_Fcs_o)+\
              gamma_d * gamma_Q * sigmaEk_Temp_pos_o+\
              gamma_d * gamma_Q * gamma_acc*psi0_acc*sigmaEk_Acc_pos_o+\
              gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_o +\
              gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_o+\
              gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_o+\
              gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_o

sigmaEd_ub3 = gamma_d * sup * gamma_G * gamma_Gb * sigmaG_u + \
              gamma_d * sup_cc * gamma_G * gamma_Gb * sigmaGp_u+\
              gamma_sh * sup * (sigmaEk_cs_u + sigmaEk_Fcs_u)+\
              gamma_d * gamma_Q * sigmaEk_Temp_pos_u+\
              gamma_d * gamma_Q * gamma_acc*psi0_acc*sigmaEk_Acc_pos_u+\
              gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTra_u +\
              gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTra_u+\
              gamma_d * gamma_Q * psi0_traDis * sigmaEk_DisTraT_u+\
              gamma_d * gamma_Q * psi0_traPoi * sigmaEk_PoiTraT_u

VEdb3_studs =  (gamma_d * sup * gamma_G * gamma_Gb * VEk_SelfWeight + \
                gamma_d * sup_cc * gamma_G * gamma_Gb * VEk_Pave)\
                *SecPar_long.Sc/SecPar_long.I +\
               (gamma_d * gamma_Q * psi0_traDis * VEk_DisTra +\
                gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTra+\
                gamma_d * gamma_Q * psi0_traDis * VEk_DisTraT+\
                gamma_d * gamma_Q * psi0_traPoi * VEk_PoiTraT)\
                *SecPar_short.Sc/SecPar_short.I +\
                gamma_d * gamma_Q * gamma_acc * psi0_acc * Qacc/L +\
               (gamma_d * gamma_Q * (abs(FTemp_t)/SecPar_short.A +\
                abs(MEk_Temp_pos)/SecPar_short.I*\
                abs(SecPar_short.tp-SecPar_short.tp_s))*
               (SecPar_short.A-Ac/SecPar_short.nL))/L


MEdb3 = gamma_d * sup * gamma_G * gamma_Gb * MEk_SelfWeight + \
        gamma_d * sup_cc * gamma_G * gamma_Gb * MEk_Pave+\
        gamma_sh * sup * MEk_cs+\
        gamma_d * gamma_Q * gamma_acc * psi0_acc * MEK_Acc+\
        gamma_d * gamma_Q * MEk_Temp_pos+\
        gamma_d * gamma_Q * psi0_traDis * MEk_DisTra+\
```

```
            gamma_d * gamma_Q * psi0_traPoi * MEk_PoiTra+\
            gamma_d * gamma_Q * psi0_traDis * MEk_DisTraT+\
            gamma_d * gamma_Q * psi0_traPoi * MEk_PoiTraT

    # Shear studs shrinkage
    VEd_studs_cs = (gamma_sh * sup * (Fcs/SecPar_cs.A +\
                    MEk_cs/SecPar_cs.I*abs(SecPar_cs.tp-SecPar_cs.tp_s))*
                   (SecPar_cs.A-Ac) +\
                    gamma_d * gamma_Q * (abs(FTemp_c)/SecPar_short.A +\
                    abs(MEk_Temp_neg)/SecPar_short.I*\
                    abs(SecPar_short.tp-SecPar_short.tp_s))*
                   (SecPar_short.A-Ac/SecPar_short.nL))/L

    # Worst combination:
    sigmaEd_ct = max(sigmaEd_cta, sigmaEd_ctb1, sigmaEd_ctb2, sigmaEd_ctb3)
    sigmaEd_cc = max(sigmaEd_cca, sigmaEd_ccb1, sigmaEd_ccb2, sigmaEd_ccb3)
    sigmaEd_o = max(sigmaEd_oa, sigmaEd_ob1, sigmaEd_ob2, sigmaEd_ob3)
    sigmaEd_u = max(sigmaEd_ua, sigmaEd_ub1, sigmaEd_ub2, sigmaEd_ub3)
    VEd = max(VEda, VEdb1, VEdb2)
    VEd_studs = max(VEda_studs, VEdb1_studs, VEdb2_studs, VEdb3_studs,
                    VEd_studs_cs)
    MEd = max(MEda, MEdb1, MEdb2, MEdb3)

    # Design loads explained:
    # sigmaEd_ct = worse case tensile combination for concrete
    # sigmaEd_cc = worse case compressive combination. for concrete
    # sigmaEd_u = worse case compressive combination for steel
    # sigmaEd_u = worse case tensile combination for steel
    # VEd = worse case highest shear force
    # VEd_studs = worse case combination for shear force at interface
    # MEd = worse case maximum moment (used to calculate NEd for crossbeams)

    return (sigmaEd_ct, sigmaEd_cc, sigmaEd_o, sigmaEd_u, VEd, VEd_studs, MEd)

# ------------------------------------------------------------------------
# 5. Ultimate Limit State (Welds)
# Input:
#    G_cb            Self-weight of crash barriers               [float, N/mm]
#    G_concrete      Self-weight of concrete                     [float, N/mm]
#    GPavement       Self-weight of pavement                     [float, N/mm]
#    hc              Height of concrete deck                     [float, mm]
#    hp              Height of pavement                          [float, mm]
#    a               Weld thickness                              [float, mm]
#    tw              Web thickness                               [float, mm]
#    SecPar_short    Sectional parameters for short term load    [class]
#    VEd             Maximum shear force in the considered section[float, N]
#    SegPos          Position of section                         [object, -]
#
# Output:
#    TauPar1         Shear stresses parallel the weld            [float, MPa]
#    sigmai1         Normal stresses                             [float, MPa]
#    SigmaPer2       Normal stresses perpendicular the weld      [float, MPa]
#    sigmai2         Normal stresses                             [float, MPa]
# ------------------------------------------------------------------------

def ULS_welds(G_cb, G_concrete, GPavement, hc, hp, a, tw,
        SecPar_short, VEd, SegPos):

  # Values of the upper and lower value (cc=concrete cover)
  # SS-EN 1991-1-1 section 5.2.3 (4)
    sup=1.0
    sup_cc=1.1

# Load combination - Ultimate Limit State (ULS)
    gamma_G=1.35
    gamma_Q=1.5

 # Considered loads: Self-weight from concrete, pavement and crash barriers:
    # Import Traffic Load Model 2 from function:
    (Qk2, C, b_wheels) = load.TrafficLoadLM2()
```

```
    # The distribution of traffic load through concrete and pavement are 1:1:
    x = b_wheels + 2* (hc + hp)     # [mm]

    # Load from each wheel:
    q_wheelA = 0.5 * Qk2 / x        #[N/mm]
    q_wheelB = 0.5 * Qk2 / (x + C)  #[N/mm]

    # Total load from traffic and self-weight, [N/mm]:
    qtot = sup * gamma_G * G_concrete +\
            sup * gamma_G * G_cb + sup_cc * gamma_G * GPavement\
            + gamma_Q * (q_wheelA + q_wheelB)

    if SegPos == 'End':
        # 0.5 m from support - bottom weld (fillet weld)
        TauPar1 = (SecPar_short.Su * VEd)/(SecPar_short.I * 2 * a)
        SigmaPer1 = 0
        TauPer1 = 0

        sigmai1 = math.sqrt(SigmaPer1**2 + 3*(TauPer1**2+TauPar1**2))

        # For every section - Top weld (butt weld)
        TauPar2 = (SecPar_short.So * VEd)/(SecPar_short.I * tw)
        SigmaPer2 = qtot * math.sqrt(2) / (4 * a)
        TauPer2 = SigmaPer2

        sigmai2 = math.sqrt(SigmaPer2**2 + 3*(TauPer2**2+TauPar2**2))

    if SegPos == 'Span':
        sigmai1 = 0
        TauPar1 = 0
        # For every section - Top weld (butt weld)
        TauPar2 = (SecPar_short.So * VEd)/(SecPar_short.I * tw)
        SigmaPer2 = qtot * math.sqrt(2) / (4 * a)
        TauPer2 = SigmaPer2

        sigmai2 = math.sqrt(SigmaPer2**2 + 3*(TauPer2**2+TauPar2**2))

    return (TauPar1, sigmai1, SigmaPer2, sigmai2)
# ------------------------------------------------------------------------------
# 6. Lambda-method (Fatigue of welds)
# Input:
#    L              Span length                              [float, mm]
#    Nobs           Total amount of trucks per year          [float, -]
#    t_Ld           Service Life of the Bridge               [integer, years]
#
# Output:
#    lambd_span_s   Lambda value for shear stress in span    [float, -]
#    lambd_support  Lambda value for support section         [float, -]
#    lambda_span_m  Lambda value for bending stress in span  [float, -]
# ------------------------------------------------------------------------------
def LambdaMethod(L, Nobs, t_Ld):
 # Input data:
    Q0 = 480 * 10**3                #[N] SS-EN 1993-1-9
    N0 = 0.5 * 10**6                #[-] SS-EN 1993-1-9
    Qmi = 445 * 10**3               #[N] Krav Brobyggande

 # Lambda-method:

    # Lambda_2:
    lambda_2 = (Qmi/Q0)*(Nobs/N0)**(1/5)

    # Lambda_3:
    lambda_3 = (t_Ld/100)**(1/5)

    # Lambda_4:
    lambda_4 = 1.0

    # Lambda_1 and Lambda_max - Span section
    # Lambda_1:
```

```
    lambda_1_s = 2.55 - 0.7*(0.4*L/1000-10)/70
    lambda_1_m = 2.55 - 0.7*(L/1000-10)/70

    # Lambda_max:
     # For shear:
    if L >= 10000 and L <= 25000:
        lambda_maxs = 2.5 - 0.5 * (0.4*L/1000-10)/15
    if L > 25000 and L <= 80000:
        lambda_maxs = 2.0

     # For bending:
    if L >= 10000 and L <= 25000:
        lambda_maxm = 2.5 - 0.5 * (L/1000-10)/15
    if L > 25000 and L <= 80000:
        lambda_maxm = 2.0

    lambd_span_s = min(lambda_1_s * lambda_2 * lambda_3 *lambda_4, lambda_maxs)
    lambda_span_m = min(lambda_1_m * lambda_2 * lambda_3 *lambda_4, lambda_maxm
    )

    # Lambda_1 and Lambda_max - Support section:

    # Lambda_1:
    if L >= 10000 and L <= 30000:
        lambda_1_su = 2.0 - 0.3 * (L/1000-10)/20
    elif L > 30000 and L <= 80000:
        lambda_1_su = 1.7 - 0.5 * (L/1000-30)/50

    # Lambda_max:
    if L >= 10000 and L <= 30000:
        lambda_max_su = 1.8
    elif L > 30000 and L <= 80000:
        lambda_max_su = 1.8 - 0.9 * (L/1000-30)/50

    lambd_support = min(lambda_1_su * lambda_2 * lambda_3 *lambda_4,
    lambda_max_su)

    return (lambd_span_s, lambda_span_m, lambd_support, lambda_2, lambda_3,
    lambda_4, lambda_1_s, lambda_1_m)
# ------------------------------------------------------------------------------
# 7. Fatigue (Details)
# Input:    lambd_span_s    Lambda value in span, shear         [float, -]
#           lambd_span_m    Lambda value in span, moment        [float, -]
#           lambd_support   Lambda value at support             [float, -]
#           SecPar_short    Sectional parameters                [class]
#           a               Weld throat thickness               [float, mm]
#           VEd             Shear                               [float, N]
#           MEd             Bending                             [float, Nmm]
#           tw              Web thickness                       [float, mm]
#           hw              Web height                          [float, mm]
#           tfo             Thickness of upper flange           [float, mm]
#           tfu             Thickness of lower flange           [float, mm]
#           hc              Height of concrete                  [float, mm]
#           SegPos          Position of section                 [float, -]
#
# Output:   A               Design stress Mode A                [float, MPa]
#           B               Design stress Mode B                [float, MPa]
#           C               Design stress Mode C                [float, MPa]
#           D               Design stress Mode D                [float, MPa]
#           E               Design stress Mode E                [float, MPa]
#           F               Design stress Mode F                [float, MPa]
# ------------------------------------------------------------------------------
def Details_FAT(lambd_span_s, lambda_span_m, lambd_support, SecPar_short, a,
        VEd, MEd, tw, hw, tfo, tfu, hc, SegPos):

    # Partial safety factor [-]
    gamma_Ff = 1.0

    if SegPos == 'Span':
        # MODE A - Not relevant for span section
```

```
        A = 0

        # MODE B - Shear stress in web at support and bending in span:
        SigmaB = MEd / SecPar_short.I * (hc + hw + tfo - SecPar_short.tp)
        B2 = lambda_span_m * gamma_Ff * SigmaB
        B1 = 0

        # MODE C - Principle stress in web (Interaction - several sections)
        TauC = VEd * SecPar_short.Su / (SecPar_short.I * tw)
        SigmaC = MEd / SecPar_short.I * (hc + hw + tfo - SecPar_short.tp)

        SigmaiC = SigmaC/2 + 0.5 * math.sqrt(SigmaC**2 + 4*TauC**2)
        C = lambda_span_m * gamma_Ff * SigmaiC

        # MODE D - Maximum bending stress in flange (Span)
        SigmaD = MEd / SecPar_short.I * (hc + hw + tfo + tfu/2 - SecPar_short.
    tp)

        D = lambda_span_m * gamma_Ff * SigmaD

        # MODE E - Bending stress in flange
        SigmaE = MEd / SecPar_short.I * (hc + hw + tfo + tfu - SecPar_short.tp)

        E = lambda_span_m * gamma_Ff * SigmaE

        # MODE F - Bending stress in flange
        SigmaF = MEd / SecPar_short.I * (hc + hw + tfo + tfu - SecPar_short.tp)

        F = lambda_span_m * gamma_Ff * SigmaF

    if SegPos == 'End':

        # MODE A - Shear Stress in weld. At support:
        TauA = VEd * SecPar_short.Su / (SecPar_short.I * 2 * a)
        A =  lambd_support * gamma_Ff * TauA

        # MODE B - Shear stress in web at support and bending in span:
        TauB = VEd * SecPar_short.Su / (SecPar_short.I * tw)
        B1 = lambd_support * gamma_Ff * TauB
        B2 = 0

        # MODE C - Principle stress in web (Interaction - several sections)
        TauC = VEd * SecPar_short.Su / (SecPar_short.I * tw)
        SigmaC = MEd / SecPar_short.I * (hc + hw + tfo - SecPar_short.tp)

        SigmaiC = SigmaC/2 + 0.5 * math.sqrt(SigmaC**2 + 4*TauC**2)
        C = lambd_support * gamma_Ff * SigmaiC

        # MODE D - Not relevant for support section
        D = 0

        # MODE E - Bending stress in flange
        E = 0

        # MODE F - Bending stress in flange
        SigmaF = MEd / SecPar_short.I * (hc + hw + tfo + tfu - SecPar_short.tp)

        F = lambd_support * gamma_Ff * SigmaF
    return (A, B1, B2, C, D, E, F)


# ------------------------------------------------------------------------------
# 7. Fatigue Limit State
# Input:
#   L           Length of bridge        [mm]
#   SubDiv      Subdivison of bridge    [mm]
#   x_start     Shear check position    [mm]
#   x_end       Moment check position   [mm]
#
# Output:
```

```
#   VEd         Design shear fatigue    [N]
#   MEd         Design moment fatigue   [Nmm]
# ------------------------------------------------------------------------
def FAT(L, SubDiv, x_start, x_end):

# CALCULATION OF MOMENT
    x = x_end # worse case closest to mid

    # Traffic load - Fatigue Load Model 3
    (q, Ctran, Clong, Cvehic, w) = load.FatigueTrafficLoadLM3()

    # Point load
    [ILD_M, ILD_V] = ILD_Simply(L, SubDiv, x, 0, 0)
    i = int(x/SubDiv)
    isubC = int((x-Clong)/SubDiv)
    iaddC = int((x+Clong)/SubDiv)
    isubC1 = int((x-Clong-Cvehic)/SubDiv)
    iaddC1 = int((x+Clong+Cvehic)/SubDiv)
    isubC2 = int((x-2*Clong-Cvehic)/SubDiv)
    iaddC2 = int((x+2*Clong+Cvehic)/SubDiv)
    isubC3 = int((x-Cvehic/2)/SubDiv)
    iaddC3 = int((x+Cvehic/2)/SubDiv)
    isubC4 = int((x-Cvehic/2-Clong)/SubDiv)
    iaddC4 = int((x+Cvehic/2+Clong)/SubDiv)

    MEk_PoiTra_left = q*(ILD_M[i]+ILD_M[iaddC]+ILD_M[iaddC1]+ILD_M[iaddC2])
    MEk_PoiTra_right = q*(ILD_M[i]+ILD_M[isubC]+ILD_M[isubC1]+ILD_M[isubC2])
    MEk_PoiTra_eq = q*(ILD_M[isubC3]+ILD_M[iaddC3]+ILD_M[isubC4]+ILD_M[iaddC4])
    MEk_PoiTra = max(MEk_PoiTra_left, MEk_PoiTra_right, MEk_PoiTra_eq)

# CALCULATION OF SHEAR
    x = x_start # worse case closest to end

    # Traffic load - Fatigue Load Model 3
    VEk_PoiTra_left = q*(ILD_V[i]+ILD_V[iaddC]+ILD_V[iaddC1]+ILD_V[iaddC2])
    VEk_PoiTra_right = q*(ILD_V[i]+ILD_V[isubC]+ILD_V[isubC1]+ILD_V[isubC2])
    VEk_PoiTra_eq = q*(ILD_V[isubC3]+ILD_V[iaddC3]+ILD_V[isubC4]+ILD_V[iaddC4])
    VEk_PoiTra = max(VEk_PoiTra_left, VEk_PoiTra_right, VEk_PoiTra_eq)


# Load combination - Fatigue Limit State (FAT)
    gamma_Q=1.0
    psi0_traPoi=0.75

# Combination
    VEd = gamma_Q * psi0_traPoi * VEk_PoiTra

    MEd = gamma_Q * psi0_traPoi * MEk_PoiTra

    return (VEd, MEd)

# ------------------------------------------------------------------------
# 8. Ultimate Limit State (Horizontal Loads)
# Input:
#   Fwind       Wind load                                   [N/mm]
#   MEd         Maximum moment                              [Nmm]
#   L           Length of bridge                            [mm]
#   hc          Height of concrete deck                     [mm]
#   hw          Height of web                               [mm]
#   tfo         Thickness of upper flange                   [mm]
#   tfu         Thickness of lower flange                   [mm]
#   C           C-C distance between crossbeams             [mm]

# Output:
#   NEd         Normal force in crossbeams                  [N]
#   MEd         Moment in crossbeams                        [Nmm]
# ------------------------------------------------------------------------
def ULS_horizontal(Fwind, MEd_max, L, hc, hw, tfo, fu, C):

    # Heights [mm]
```

```
    Hs = hw+tfo+fu          # steel section

    # Wind load [N/mm]
    gamma_Q1 = 1.5
    NEd_wind = gamma_Q1*Fwind*C  # wind load per crossbeam

    # Bracing load
    num = L/C
    MEd_max = MEd_max/num   # moment per crossbeam
    NEd_moment = MEd_max/Hs
    m = 2                   # two main girders
    NEd_bracing = math.sqrt(0.5*(1+1/m))*NEd_moment/100

    # I-section or truss
    if Hs < 2e3:
        # Crossbeam placed at mid of steel section:
        # Normal force
        NEd = NEd_wind+NEd_bracing
        # Moment
        MEd_wind = hc/2*NEd_wind
        MEd_bracing = Hs/2*NEd_bracing
        MEd = MEd_wind + MEd_bracing

    else:
        # Truss beam placed evenly spaced from the mid of steel section:
        # Normal force
        MEd_wind = hc/2*NEd_wind
        MEd_bracing = Hs/2*NEd_bracing
        c = Hs/2 # truss deep as half of girder height
        NEd_moment = (MEd_wind+MEd_bracing)/c
        NEd = NEd_wind+NEd_bracing + NEd_moment
        # Moment (zero in truss)
        MEd = 0

    return(NEd, MEd)
```

D

Life Cycle Cost Optimization Results

In this appendix the design dimensions and resulting utilization rates are presented for the life cycle cost studies presented in Chapter 5.2. This appendix also includes the convergence graphs of each alternative studied in each sub-study.

D.1 S355 Flat Web with hw < 2 m

The design dimensions chosen by the optimization program for flat web girders in S355 steel with N_{obs} = 0.05e6, ADT = 5e3 and a height domain of 1-2 meters are presented in Table D.1. Additionally, the program choose a bracing system with I-beams every 4.25 meters of section HEA120. The resulting utilization rations are presented in Table D.2.

Table D.1: Design dimensions [mm]: S355 Flat Web with hw < 2 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	1890	1890	1890	1890	1890
tw	20	20	20	20	20
bfo	500	500	500	500	500
tfo	16	25	35	45	55
bfu	1300	1300	1300	1300	1300
tfu	16	30	45	55	55
C-C Studs	97	128	165	235	408
Start x-coord.	0	5550	9650	14750	22850
End x-coord.	5550	9650	14750	22850	25500

Table D.2: Utilization ratios [%]: S355 Flat Web with hw < 2 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	36,8	28,8	22,9	15,5	3,8
	M	95,4	87,0	98,4	99,6	88,3
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Service Phase	**V**	92,5	76,8	63,7	46,2	25,3
	M_s	96,9	99,6	97,0	98,4	98,9
	M_{cc}	33,1	43,2	51,6	58,1	57,3
	M_{ct}	0,0	0,0	0,0	0,0	0,0
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Weld	**i1**	53,6	-	-	-	-
	2	7,8	7,8	7,8	7,8	7,8
	i2	12,6	12,7	12,7	12,7	12,7
FAT	**A**	10,0	-	-	-	-
	B1	4,0	-	-	-	-
	B2	-	40,7	38,9	37,9	38,3
	C	35,6	52,2	50,0	48,4	49,3
	D	-	51,4	49,3	48,2	48,6
	E	-	41,4	40,0	39,2	39,6
	F	25,2	38,4	40,1	41,0	41,4
SLS	**Defection***	-	-	-	-	42,4

**Maximum mid span deflection using the average bending stiffness of all segments.*

The convergence plots of the three runs are shown in Figure D.1. To be able to visually see the change between the runs, only iterations with objectives below 8 million SEK are plotted. Run 1 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

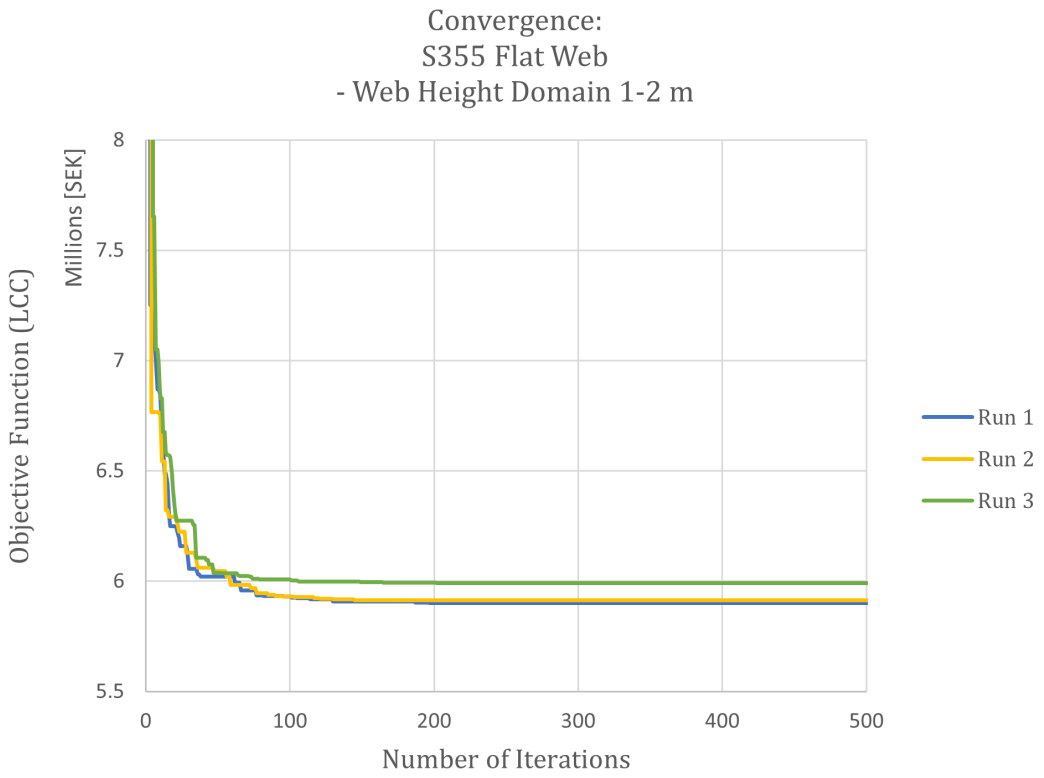


Figure D.1: Convergence plots: S355 Flat Web hw < 2 m.

D.2 S355 Flat Web with hw < 3 m

The design dimensions chosen by the optimization program for flat web girders in S355 steel with $N_{obs} = 0.05e6$, ADT = 5e3 and a height domain between 1-3 meters are presented in Table D.3. Additionally, the program choose a K-truss bracing system with trusses every 8.5 meters, with element section HEA120. The resulting utilization rations are presented in Table D.4.

Table D.3: Design dimensions [mm]: S355 Flat Web with hw < 3 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	2470	2470	2470	2470	2470
tw	25	25	25	25	25
bfo	650	650	650	650	650
tfo	16	16	20	35	40
bfu	1050	1050	1050	1050	1050
tfu	16	25	35	40	45
C-C Studs	122	160	200	258	367
Start x-coord.	0	7550	11150	15250	18350
End x-coord.	7550	11150	15250	18350	25500

Table D.4: Utilization ratios [%]: S355 Flat Web with hw < 3 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	23,9	16,8	13,5	9,6	6,7
	M	71,7	90,1	96,5	94,1	99,0
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Service Phase	**V**	59,0	46,1	38,8	31,2	23,1
	M_s	91,6	98,9	99,1	98,9	98,8
	M_{cc}	31,2	39,0	44,4	43,8	45,7
	M_{ct}	0,0	0,0	0,0	0,0	0,0
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Weld	**i1**	30,5	-	-	-	-
	2	7,8	7,8	7,8	7,8	7,8
	i2	12,6	12,6	12,6	12,7	12,7
FAT	**A**	5,6	-	-	-	-
	B1	1,8	-	-	-	-
	B2	-	39,9	39,5	38,8	38,3
	C	32,5	50,1	49,7	48,8	48,2
	D	-	50,1	49,8	48,9	48,3
	E	-	40,3	40,2	39,5	39,1
	F	23,3	36,0	38,4	38,8	39,2
SLS	**Defection***	-	-	-	-	30,9

**Maximum mid span deflection using the average bending stiffness of all segments.*

The convergence plots of the three runs are shown in Figure D.2. To be able to visually see the change between the runs, only iterations with objectives below 8 million SEK are plotted. Run 1 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

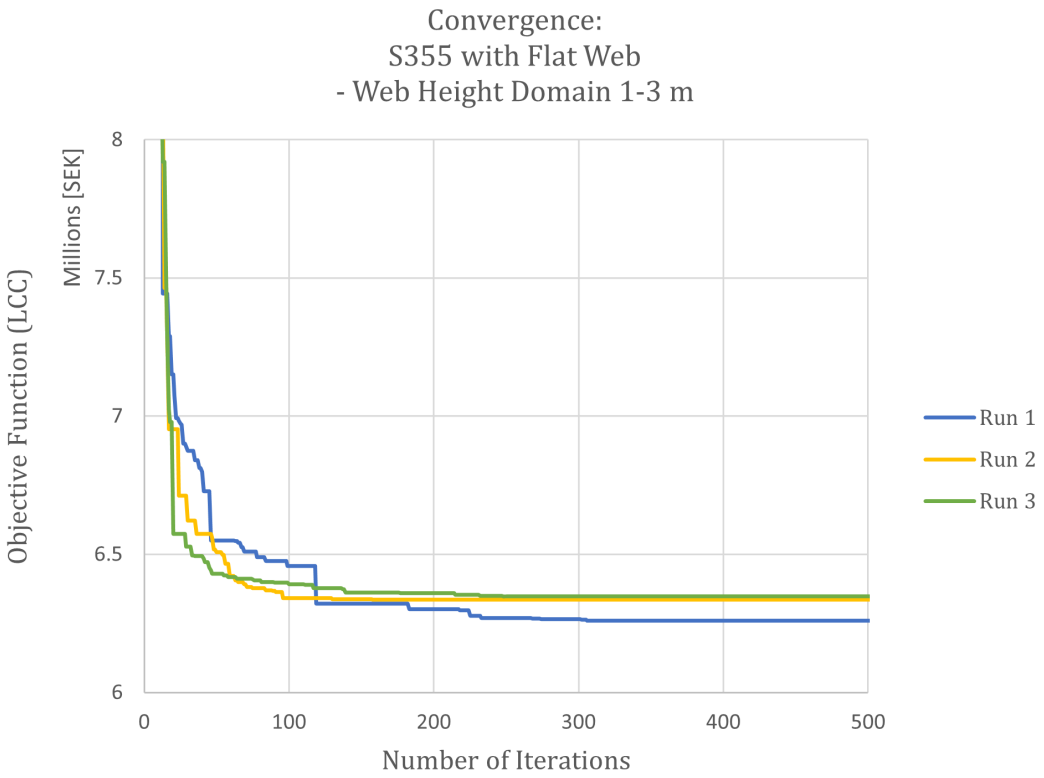


Figure D.2: Convergence plots: S355 Flat Web hw < 3 m.

D.3 Duplex Corrugated Web with hw < 2 m

The design dimensions chosen by the optimization program for corrugated web girders in Duplex steel with N_{obs} = 0.05e6, ADT = 5e3 and a height domain between 1-2 meters are presented in Table D.5. Additionally, the program choose a bracing system with I-beams every 3.4 meters of section HEA120. The resulting utilization rations are presented in Table D.6.

Table D.5: Design dimensions [mm]: Duplex Corrugated Web with hw < 2 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	1890	1890	1890	1890	1890
tw	10	10	10	10	10
bfo	450	450	450	450	450
tfo	25	35	45	50	60
bfu	1450	1450	1450	1450	1450
tfu	25	35	40	45	50
C-C Studs	113	141	171	209	296
Start x-coord.	0	6550	10650	14250	18850
End x-coord.	6550	10650	14250	18850	25500

Table D.6: Utilization ratios [%]: Duplex Corrugated Web with hw < 2 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	39.3	29.2	22.9	17.3	10.2
	M	93.5	98.1	92.6	96.2	85.7
	Interaction	-	-	-	-	-
ULS Service Phase	**V**	99.1	79.7	66.8	54.3	37.4
	$\mathbf{M}_s$	97.8	95.7	98.7	99.7	94.9
	$\mathbf{M}_{cc}$	47.6	56.1	61.8	67.1	67.8
	$\mathbf{M}_{ct}$	0.0	0.0	0.0	0.0	0.0
	Interaction	-	-	-	-	-
ULS Weld	**i1**	70.8	-	-	-	-
	2	5.9	5.9	5.9	5.9	5.9
	i2	10.6	10.7	10.7	10.7	10.7
FAT	**A**	15.7	-	-	-	-
	B1	12.6	-	-	-	-
	B2	-	43.8	46.0	46.1	43.7
	C	42.6	61.9	63.6	62.7	61.3
	D	-	55.3	58.2	58.3	55.5
	E	-	44.7	47.1	47.3	45.0
	F	26.7	42.7	46.2	47.5	46.2
SLS	**Deflection***	-	-	-	-	49.5

**Maximum mid span deflection using the average bending stiffness of all segments.*

Additionally, the corrugation parameters of the optimized solution is presented in Table D.7.

Table D.7: Corrugation parameters. Duplex Corrugated Web with hw $<$ 2 m.

$\mathbf{a}_1$	180	mm
$\mathbf{a}_2$	100	mm
$\mathbf{a}_3$	50	mm
$\mathbf{a}_4$	87	mm
α	30	degrees

The convergence plots of the three runs are shown in Figure D.3. To be able to visually see the change between the runs, only iterations with objectives below 10.5 million SEK are plotted. Run 3 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

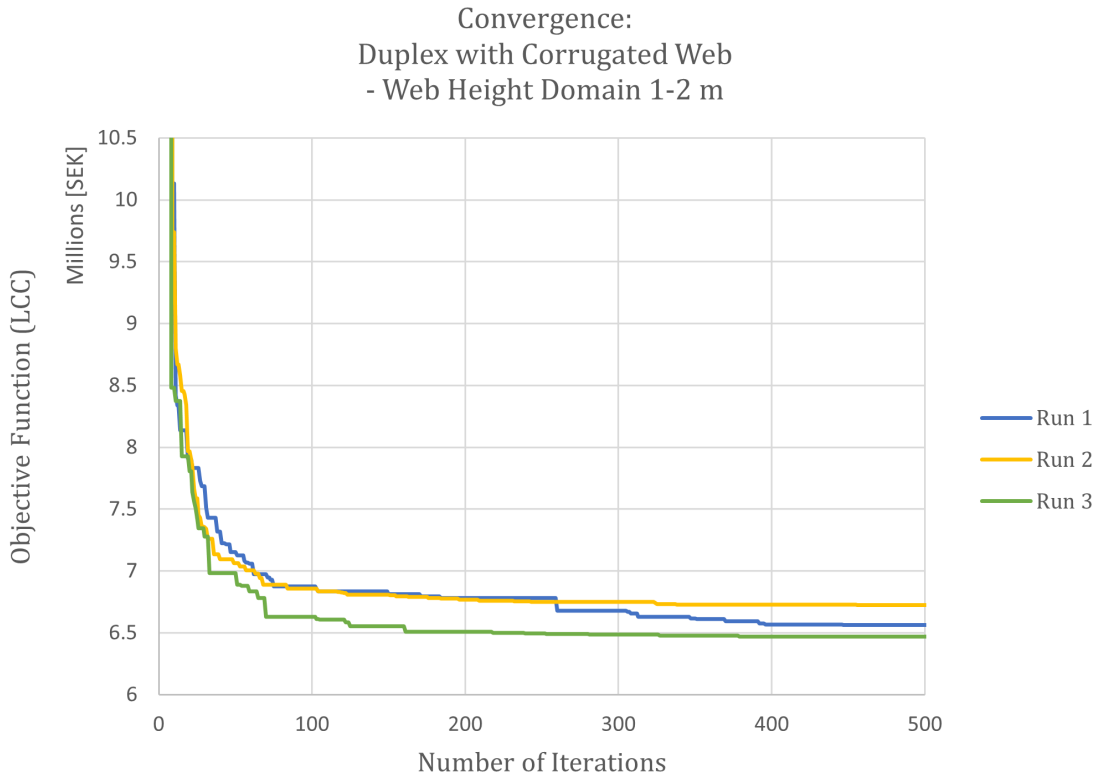


Figure D.3: Convergence plots: Duplex Corrugated Web with hw $<$ 2 m.

D.4 Duplex Corrugated Web with hw < 3 m

The design dimensions chosen by the optimization program for corrugated web girders in Duplex steel with N_{obs} = 0.05e6, ADT = 5e3 and a height domain between 1-3 meters are presented in Table D.8. Additionally, the program choose a K-truss bracing system with trusses every 5.1 meters, with element section HEA100. The resulting utilization rations are presented in Table D.9.

Table D.8: Design dimensions [mm]: Duplex Corrugated Web with hw < 3 m.

Segment	**1**	**2**	**3**	**4***	**5***
hw	2850	2850	2850	2850	2850
tw	8	8	8	8	8
bfo	450	450	450	450	450
tfo	25	30	40	45	45
bfu	1200	1200	1200	1200	1200
tfu	18	28	35	40	40
C-C studs	160	196	243	329	448
xstart(seg)	0	5050	10150	15750	20350
xend(seg)	5050	10150	15750	20350	25500

**Same dimensions for segment 4 and 5 means no welding between these segments.*

Table D.9: Utilization ratios [%]: Duplex Corrugated Web with hw < 3 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	38.5	30.9	23.2	14.7	7.8
	M	90.7	95.6	95.7	95.4	99.5
	Interaction	-	-	-	-	-
ULS Service Phase	**V**	99.9	83.5	66.6	49.3	33.6
	M_s	99.1	98.0	99.7	96.4	99.9
	M_{cc}	31.5	38.5	44.3	46.7	48.4
	M_{ct}	0.0	0.0	0.0	0.0	0.0
	Interaction	-	-	-	-	-
ULS Weld	**i1**	48.9	-	-	-	-
	2	5.9	5.9	5.9	5.9	5.9
	i2	10.6	10.7	10.7	10.7	10.6
FAT	**A**	11.1	-	-	-	-
	B1	11.1	-	-	-	-
	B2	-	44.4	46.6	44.5	46.2
	C	39.0	60.9	62.6	59.4	62.7
	D	-	55.7	58.7	56.1	58.2
	E	-	44.8	47.2	45.2	46.9
	F	24.5	40.9	45.1	44.4	46.0
SLS	**Deflection***	-	-	-	-	32.8

**Maximum mid span deflection using the average bending stiffness of all segments.*

Additionally, the corrugation parameters of the optimized solution is presented in Table D.10.

Table D.10: Corrugation parameters. Duplex Corrugated Web with hw $<$ 3 m.

$\mathbf{a}_1$	320	mm
$\mathbf{a}_2$	140	mm
$\mathbf{a}_3$	70	mm
$\mathbf{a}_4$	121	mm
α	30	degrees

The convergence plots of the three runs are shown in Figure D.4. To be able to visually see the change between the runs, only iterations with objectives below 9.5 million SEK are plotted. Run 1 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

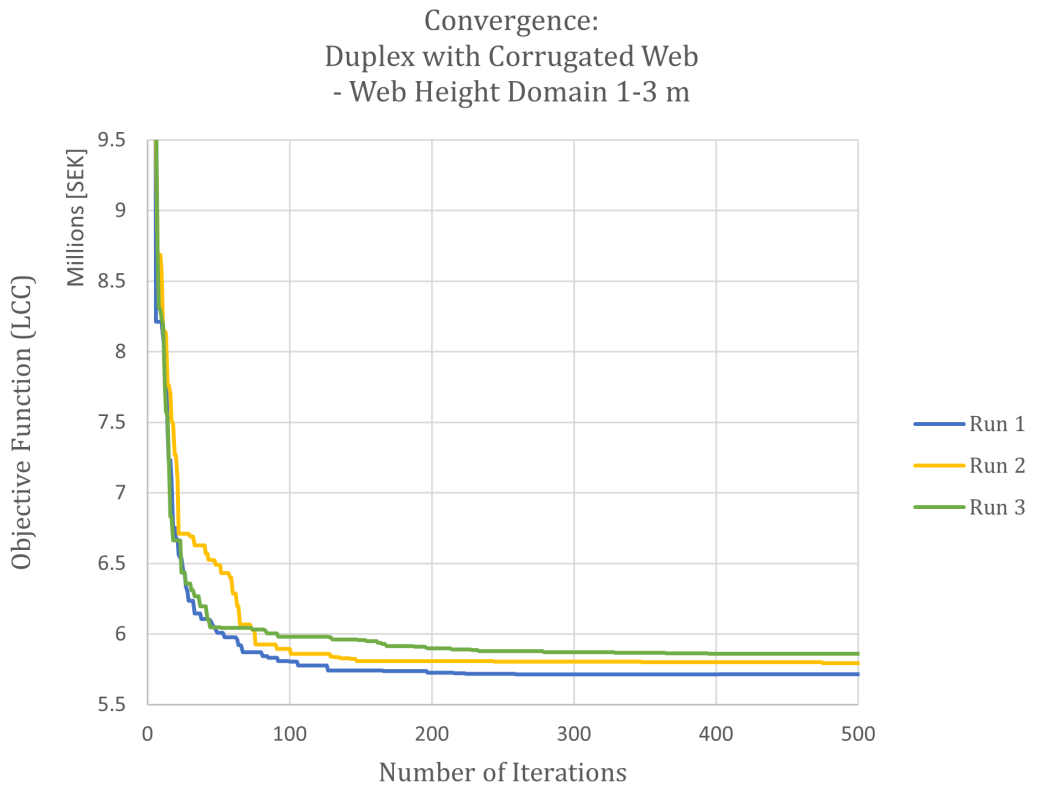


Figure D.4: Convergence plots: Duplex Corrugated Web with hw $<$ 3 m.

D.5 S355 Flat Web with Increased Nobs

The design dimensions chosen by the optimization program for flat web girders in S355 steel with N_{obs} = 0.5e6, ADT = 50e6 and a height domain of 1-2 meters are presented in Table D.11. Additionally, the program choose a bracing system with I-beams every 4.25 meters of section HEA120. The resulting utilization rations are presented in Table D.12.

Table D.11: Design dimensions [mm]: S355 Flat with Increased Nobs.

Segment	**1**	**2**	**3**	**4***	**5***
hw	1890	1890	1890	1890	1890
tw	20	20	20	20	20
bfo	500	500	500	500	500
tfo	16	25	35	50	50
bfu	1200	1200	1200	1200	1200
tfu	18	35	45	60	60
C-C Studs	98	130	164	214	332
Start x-coord.	0	5550	10150	13750	19850
End x-coord.	5550	10150	13750	19850	25500

**Same dimensions for segments 4 and 5 means no welding between these segments.*

Table D.12: Utilization ratios [%]: S355 Flat Web with Height 1-2 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	36,8	28,8	22,1	17,0	8,2
	M	94,7	89,5	95,0	89,2	93,9
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Service Phase	**V**	92,5	76,5	63,3	50,7	32,2
	$\mathbf{M}_s$	94,8	98,0	99,1	94,0	98,7
	$\mathbf{M}_{cc}$	32,8	44,1	50,2	55,0	57,8
	$\mathbf{M}_{ct}$	0,0	0,0	0,0	0,0	0,0
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Weld	**i1**	54,4	-	-	-	-
	2	7,8	7,8	7,8	7,8	7,8
	i2	12,6	12,7	12,7	12,7	12,7
FAT	**A**	16,0	-	-	-	-
	B1	6,4	-	-	-	-
	B2	-	61,5	61,4	56,1	58,6
	C	54,8	78,9	78,8	72,0	75,6
	D	-	77,7	77,8	71,4	74,6
	E	-	62,8	63,1	58,2	60,8
	F	38,7	59,9	63,3	61,9	64,7
SLS	**Defection***	-	-	-	-	42,3

**Maximum mid span deflection using the average bending stiffness of all segments.*

The convergence plots of the three runs are shown in Figure D.5. To be able to visually see the change between the runs, only iterations with objectives below 9 million SEK are plotted. Run 1 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

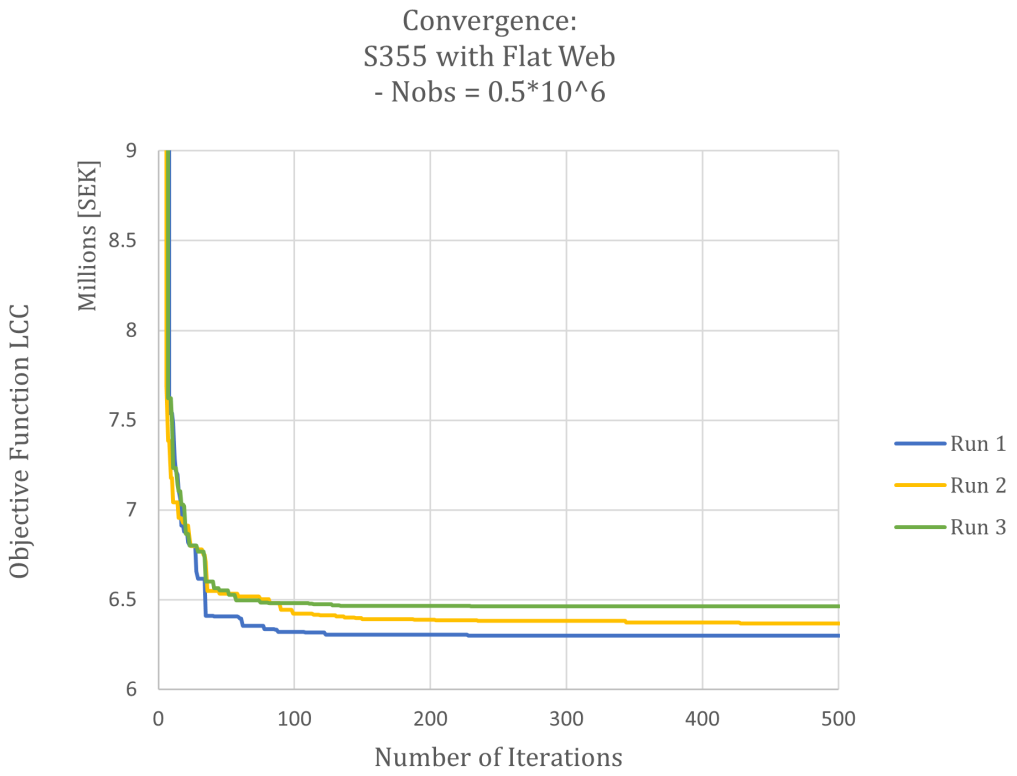


Figure D.5: Convergence plots: S355 Flat Web with $N_{obs} = 0.5 \cdot 10^6$.

D.6 Duplex Corrugated Web with Increased Nobs

The design dimensions chosen by the optimization program for corrugated web girders in Duplex steel with N_{obs} = 0.5e6, ADT = 50e6 and a height domain between 1-2 meters are presented in Table D.13. Additionally, the program choose a bracing system with I-beams every 3.4 meters of section HEA120. The resulting utilization rations are presented in Table D.14.

Table D.13: Design dimensions [mm]: Duplex Corrugated Web with high Nobs.

Segment	**1**	**2**	**3**	**4**	**5**
hw	1890	1890	1890	1890	1890
tw	10	10	10	10	10
bfo	450	450	450	450	450
tfo	25	35	45	50	60
bfu	1400	1400	1400	1400	1400
tfu	20	35	45	50	50
C-C studs	111	134	175	236	332
Start x-coord.	0	4550	10650	16250	20850
End x-coord.	4550	10650	16250	20850	25500

Table D.14: Utilization ratios [%]: Duplex Corrugated Web with high Nobs.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	37.6	30.9	21.9	13.6	6.9
	M	68.0	98.0	99.7	99.6	85.7
	Interaction	-	-	-	-	-
ULS Service Phase	**V**	96.4	81.1	62.4	45.7	31.0
	$\mathbf{M}_s$	97.4	99.0	97.2	96.1	98.2
	$\mathbf{M}_{cc}$	41.3	56.4	64.5	68.1	68.2
	$\mathbf{M}_{ct}$	0.0	0.0	0.0	0.0	0.0
	Interaction	-	-	-	-	-
ULS Weld	**i1**	71.6	-	-	-	-
	2	5.9	5.9	5.9	5.9	5.9
	i2	10.6	10.7	10.7	10.7	10.7
FAT	**A**	25.5	-	-	-	-
	B1	20.4	-	-	-	-
	B2	-	69.9	69.8	67.8	69.8
	C	64.5	98.0	96.1	92.3	97.1
	D	-	88.2	88.4	86.1	88.5
	E	-	71.3	71.7	69.9	71.8
	F	39.3	68.1	72.0	71.6	73.7
SLS	**Deflection***	-	-	-	-	50.2

**Maximum mid span deflection using the average bending stiffness of all segments.*

Additionally, the corrugation parameters of the optimized solution is presented in Table D.15.

Table D.15: Corrugation parameters. Duplex Corrugated Web with high Nobs.

$\mathbf{a}_1$	150	mm
$\mathbf{a}_2$	100	mm
$\mathbf{a}_3$	50	mm
$\mathbf{a}_4$	87	mm
α	30	degrees

The convergence plots of the three runs are shown in Figure D.6. To be able to visually see the change between the runs, only iterations with objectives below 10.5 million SEK are plotted. Run 1 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

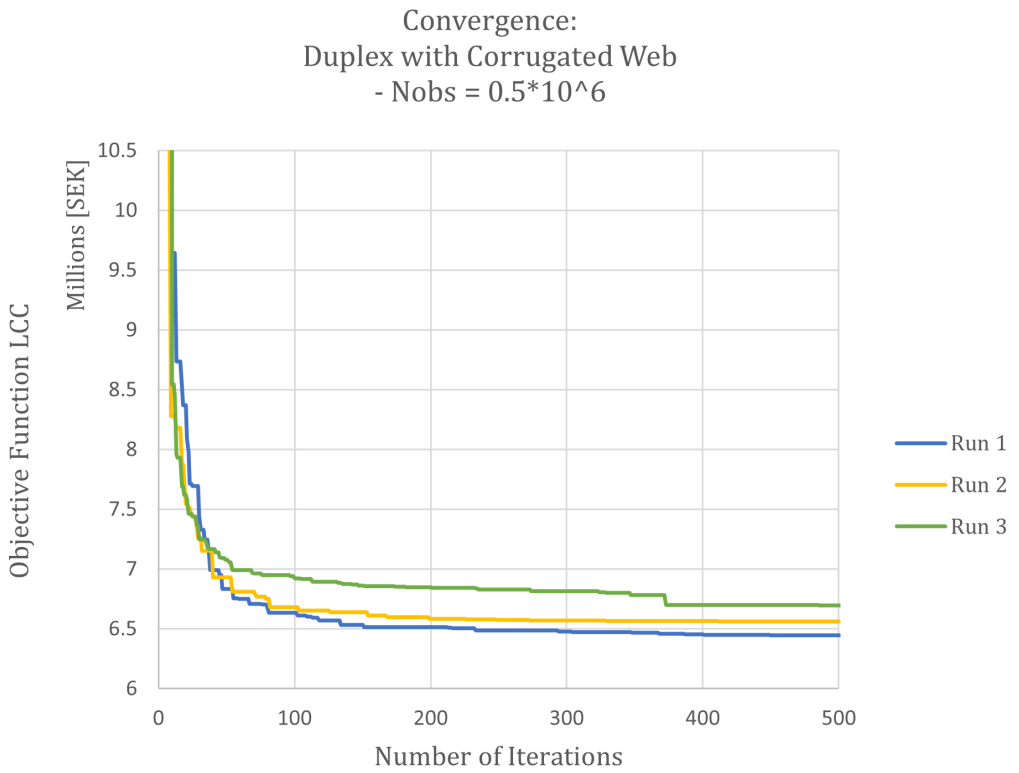


Figure D.6: Convergence plots: Duplex Corrugated Web with $N_{obs} = 0.5 \cdot 10^6$.

D.7 S355 Flat Web with Decreased Price

The design dimensions chosen by the optimization program for flat web girders in S355 steel with $N_{obs} = 0.5e6$, ADT = 50e6 and a height domain between 1-2 meters are presented in Table D.16. Additionally, the program choose a bracing system with I-beams every 5.1 meters of section HEA120. The resulting utilization rations are presented in Table D.17.

Table D.16: Design dimensions [mm]: S355 Flat Web with Decreased Material Price.

Segment	**1**	**2**	**3**	**4***	**5***
hw	1900	1900	1900	1900	1900
tw	20	20	20	20	20
bfo	600	600	600	600	600
tfo	16	25	28	40	40
bfu	1200	1200	1200	1200	1200
tfu	18	28	45	60	60
C-C Studs	98	124	153	205	294
Start x-coord.	0	5550	8150	13750	17850
End x-coord.	5550	8150	13750	17850	25500

**Same dimensions for segment 4 and 5 means no welding between these segments.*

Table D.17: Utilization ratios [%]: S355 Flat Web with decreased material price.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	37,1	29,0	25,3	17,1	11,1
	M	93,4	92,9	96,2	86,8	95,3
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Service Phase	**V**	93,5	78,7	68,6	52,6	37,2
	$\mathbf{M}_s$	94,1	96,5	98,9	90,0	98,6
	$\mathbf{M}_{cc}$	32,5	39,5	50,2	52,7	57,9
	$\mathbf{M}_{ct}$	0,0	0,0	0,0	0,0	0,0
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Weld	**i1**	54,0	-	-	-	-
	2	7,8	7,8	7,8	7,8	7,8
	i2	12,6	12,7	12,7	12,7	12,7
FAT	**A**	10,1	-	-	-	-
	B1	4,0	-	-	-	-
	B2	-	39,6	39,8	35,1	38,0
	C	34,6	50,8	51,0	45,3	49,1
	D	-	49,9	50,4	44,7	48,4
	E	-	40,2	40,9	36,5	39,5
	F	24,4	36,8	41,0	38,8	42,0
SLS	**Defection***	-	-	-	-	43,1

**Maximum mid span deflection using the average bending stiffness of all segments.*

The convergence plots of the three runs are shown in Figure D.7. To be able to visually see the change between the runs, only iterations with objectives below 6.5 million SEK are plotted. Run 2 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

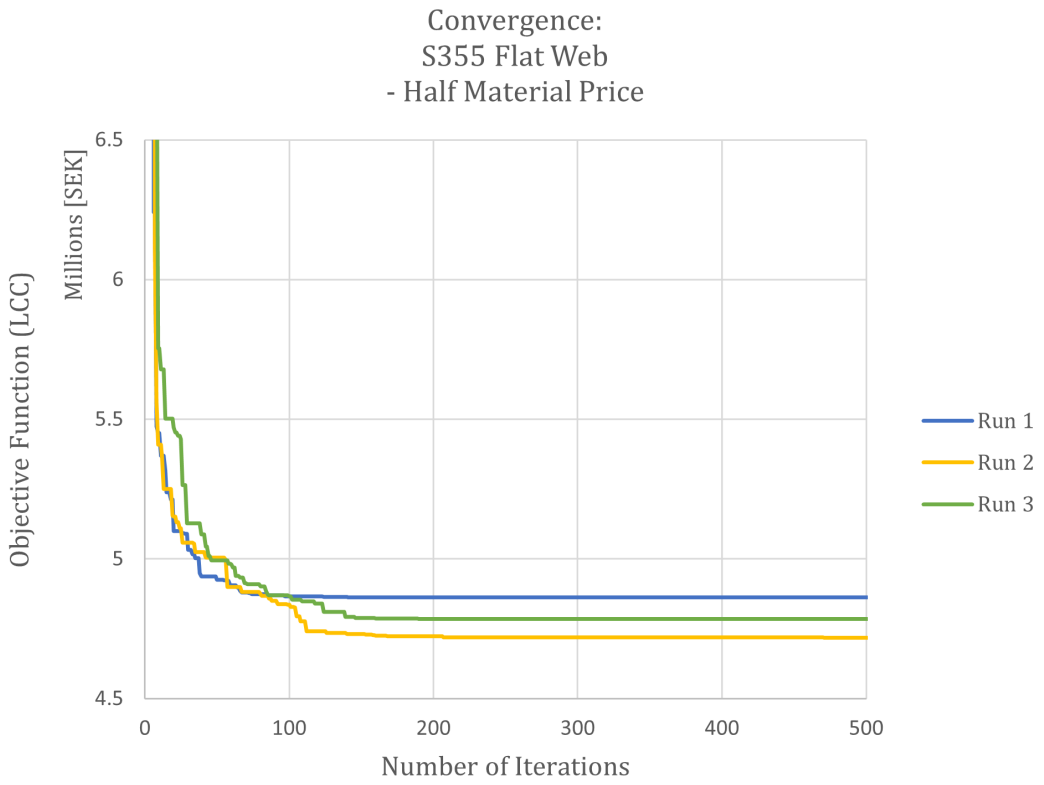


Figure D.7: Convergence plots: S355 Flat Web with decreases material price.

D.8 Duplex Corrugated Web with Decreased Price

The design dimensions chosen by the optimization program for corrugated web girders in Duplex steel with N_{obs} = 0.5e6, ADT = 50e6 and a height domain between 1-2 meters are presented in Table D.18. Additionally, the program choose a bracing system with I-beams every 4.3 meters of section HEA120. The resulting utilization rations are presented in Table D.19.

Table D.18: Design dimensions [mm]: Duplex Corrugated Web with decreased material price.

Segment	**1**	**2**	**3**	**4**	**5**
hw	1890	1890	1890	1890	1890
tw	10	10	10	10	10
bfo	450	450	450	450	450
tfo	25	35	45	50	60
bfu	1400	1400	1400	1400	1400
tfu	25	35	45	50	50
C-C studs	112	139	175	236	332
Start x-coord.	0	6050	10650	16250	20850
End x-coord.	6050	10650	16250	20850	25500

Table D.19: Utilization ratios [%]: Duplex Corrugated Web with decreased material price.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	39.3	30.0	22.9	14.3	7.2
	M	87.5	98.2	99.9	99.8	85.8
	Interaction	-	-	-	-	-
ULS Service Phase	**V**	99.6	81.0	65.2	47.8	32.4
	$\mathbf{M}_s$	96.0	99.2	97.4	96.3	98.4
	$\mathbf{M}_{cc}$	46.0	56.6	64.7	68.3	68.3
	$\mathbf{M}_{ct}$	0.0	0.0	0.0	0.0	0.0
	Interaction	-	-	-	-	-
ULS Weld	**i1**	71.1	0.0	0.0	0.0	0.0
	2	5.9	5.9	5.9	5.9	5.9
	i2	10.6	10.7	10.7	10.7	10.7
FAT	**A**	15.9	0.0	0.0	0.0	0.0
	B1	12.7	0.0	0.0	0.0	0.0
	B2	0.0	45.4	45.3	44.0	45.3
	C	41.7	63.6	62.4	59.9	63.0
	D	0.0	57.3	57.4	55.9	57.5
	E	0.0	46.3	46.5	45.3	46.6
	F	25.8	44.2	46.7	46.5	47.8
SLS	**Deflection***	-	-	-	-	48.7

**Maximum mid span deflection using the average bending stiffness of all segments.*

Additionally, the corrugation parameters of the optimized solution is presented in Table D.20.

Table D.20: Corrugation parameters. Duplex Corrugated Web with high Nobs.

$\mathbf{a}_1$	180	mm
$\mathbf{a}_2$	100	mm
$\mathbf{a}_3$	50	mm
$\mathbf{a}_4$	87	mm
α	30	degrees

The convergence plots of the three runs are shown in Figure D.8. To be able to visually see the change between the runs, only iterations with objectives below 6 million SEK are plotted. Run 2 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

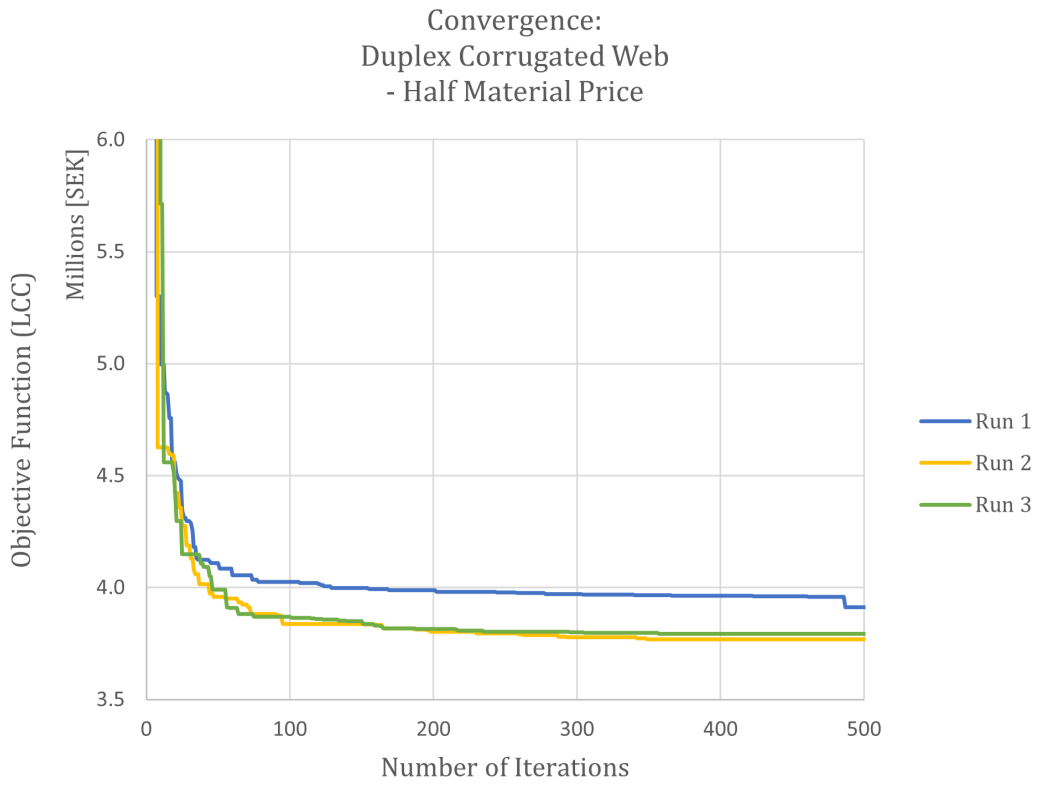


Figure D.8: Convergence plots: Duplex Corrugated Web with decreased material price.

E

Mass Optimization Results

In this appendix the design dimensions and resulting utilization rates are presented for the weight studies presented in Chapter 5.3. This appendix also includes the convergence graphs of each alternative studied.

E.1 S355 Flat Web with hw < 2 m

The design dimensions chosen by the optimization program for flat web girders in S355 steel with $N_{obs} = 0.05e6$, ADT=5e3 and a height domain of 1-2 meters are presented in Table E.1. Additionally, the program choose a K-truss bracing system with trusses every 4.25 meters, with element section HEA100. The resulting utilization rations are presented in Table E.2.

Table E.1: Design dimensions [mm]: S355 Flat Web with hw < 2 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	1970	1970	1970	1970	1970
tw	20	20	20	20	20
bfo	500	500	500	500	500
tfo	16	25	28	40	45
bfu	1350	1350	1350	1350	1350
tfu	16	28	40	45	50
C-C studs	103	136	167	214	300
Start x-coord.	0	6050	9650	14750	17850
End x-coord.	6050	9650	14750	17850	25500

Table E.2: Utilization ratios [%]: S355 Flat Web with hw < 2 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	35.9	27.4	22.3	15.1	10.8
	M	94.9	82.0	99.5	94.4	95.5
	Interaction	-	-	-	-	-
ULS Service Phase	**V**	90.3	74.1	62.5	48.8	36.1
	$\mathbf{M}_s$	96.9	97.2	99.0	99.1	99.6
	$\mathbf{M}_{cc}$	33.4	41.2	50.8	53.2	56.5
	$\mathbf{M}_{ct}$	0.0	0.0	0.0	0.0	0.0
	Interaction	-	-	-	-	-
ULS Weld	**i1**	51.4	-	-	-	-
	2	7.8	7.8	7.8	7.8	7.8
	i2	12.6	12.7	12.7	12.7	12.7
FAT	**A**	9.6	-	-	-	-
	B1	3.8	-	-	-	-
	B2	-	39.8	40.0	39.2	38.8
	C	35.6	51.0	51.1	50.1	49.8
	D	-	50.2	50.5	49.7	49.2
	E	-	40.5	40.9	40.3	39.9
	F	25.2	37.0	40.1	40.4	41.0
SLS	**Defection***	-	-	-	-	41.4

**Maximum mid span deflection using the average bending stiffness of all segments.*

The convergence plots of the three runs are shown in Figure E.1. To be able to visually see the change between the runs, only iterations with objectives below 125 ton are plotted. Run 2 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

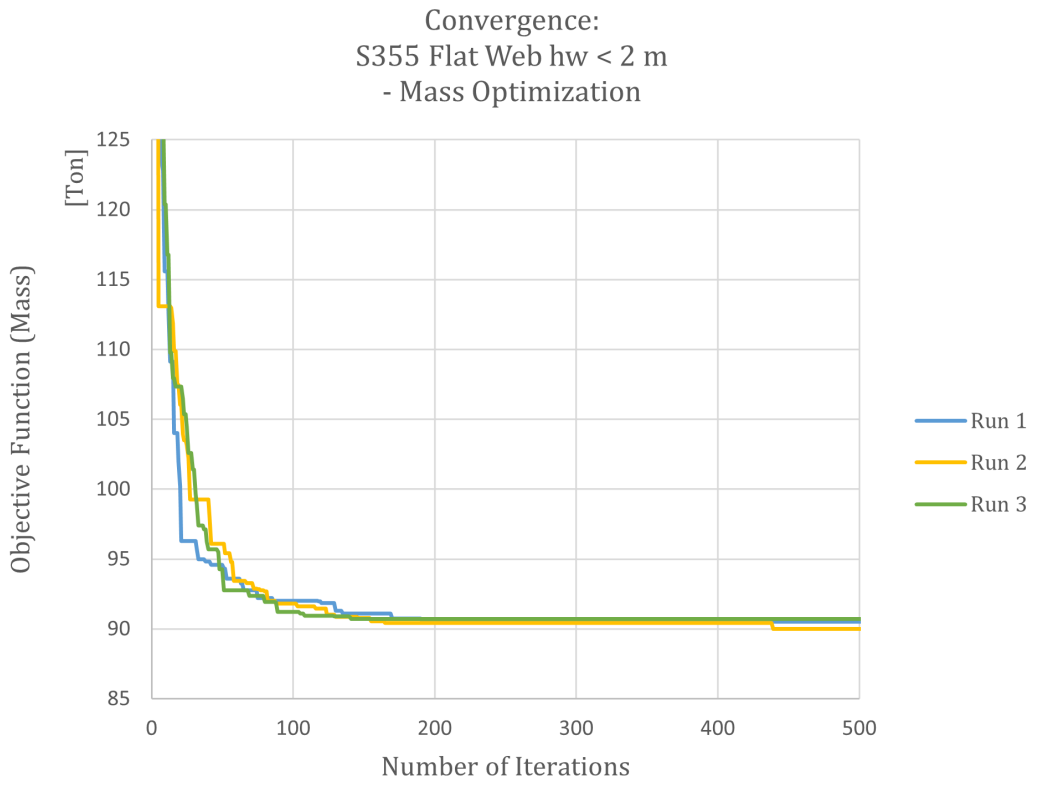


Figure E.1: Convergence plots: S355 Flat Web hw $<$ 2 m.

E.2 S355 Flat Web with hw < 3 m

The design dimensions chosen by the optimization program for flat web girders in S355 steel with $N_{obs} = 0.05e6$, ADT = 5e3 and a height domain between 1-3 meters are presented in Table E.3. Additionally, the program choose a K-truss bracing system with trusses every 3.4 meters, with element section HEA100. The resulting utilization rations are presented in Table E.4.

Table E.3: Design dimensions [mm]: S355 Flat Web with hw < 3 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	2790	2790	2790	2790	2790
tw	18	18	18	18	18
bfo	400	400	400	400	400
tfo	18	20	28	35	40
bfu	950	950	950	950	950
tfu	16	25	35	40	45
C-C studs	140	185	234	295	397
Start x-coord.	0	7550	11150	15250	18350
End x-coord.	7550	11150	15250	18350	25500

Table E.4: Utilization ratios [%]: S355 Flat Web with hw < 3 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	34.6	24.3	19.5	13.9	9.7
	M	79.6	99.7	99.1	93.8	98.0
	Interaction	0.0	0.0	0.0	0.0	0.0
ULS Service Phase	**V**	86.9	67.9	57.2	46.2	34.1
	$\mathbf{M}_s$	93.7	99.9	99.0	99.1	98.7
	$\mathbf{M}_{cc}$	26.9	33.5	37.9	39.9	41.9
	$\mathbf{M}_{ct}$	0.0	0.0	0.0	0.0	0.0
	Interaction	0.0	0.0	0.0	0.0	0.0
ULS Weld	**i1**	28.4	-	-	-	-
	2	7.8	7.8	7.8	7.8	7.8
	i2	12.7	12.7	12.7	12.7	12.7
FAT	**A**	5.3	-	-	-	-
	B1	2.3	-	-	-	-
	B2	-	41.0	40.2	39.5	38.8
	C	33.9	51.7	50.8	49.8	49.2
	D	-	51.5	50.6	49.7	49.0
	E	-	41.4	40.8	40.1	39.5
	F	24.2	36.9	38.9	39.4	39.7
SLS	**Defection***	-	-	-	-	27.6

**Maximum mid span deflection using the average bending stiffness of all segments.*

The convergence plots of the three runs are shown in Figure E.2. To be able to visually see the change between the runs, only iterations with objectives below 115 ton are plotted. Run 3 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

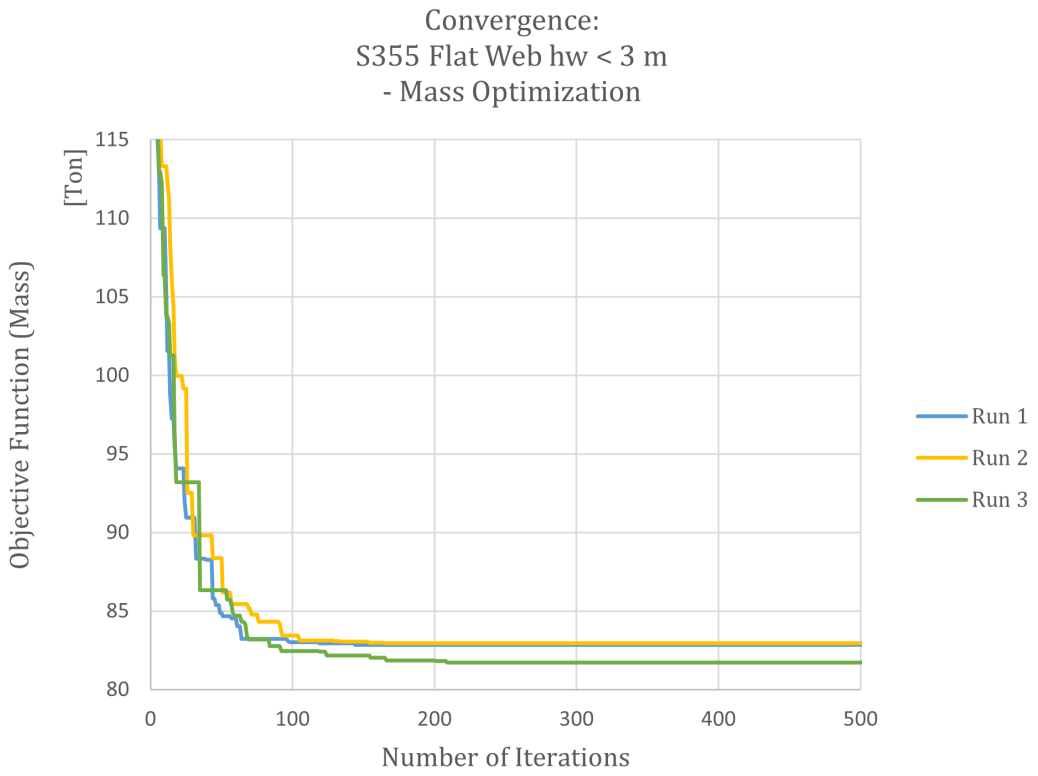


Figure E.2: Convergence plots: S355 Flat Web hw $<$ 3 m.

E.3 Duplex Corrugated Web with hw < 2 m

The design dimensions chosen by the optimization program for corrugated web girders in Duplex steel with $N_{obs} = 0.05e6$, ADT=5e3 and a height domain between 1-2 meters are presented in Table E.5. Additionally, the program choose a bracing system with I-beams every 3.4 meters of section HEA120. The resulting utilization rations are presented in Table E.6.

Table E.5: Design dimensions [mm]: Duplex Corrugated Web with hw < 2 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	1900	1900	1900	1900	1900
tw	10	10	10	10	10
bfo	500	500	500	500	500
tfo	28	30	40	45	50
bfu	1400	1400	1400	1400	1400
tfu	25	35	45	50	50
C-C studs	115	141	172	232	332
Start x-coord.	0	6550	10150	15750	20850
End x-coord.	6550	10150	15750	20850	25500

Table E.6: Utilization ratios [%]: Duplex Corrugated Web with hw < 2 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	39.1	29.1	23.5	15.0	7.1
	M	74.8	99.1	99.1	99.4	92.5
	Interaction	-	-	-	-	-
ULS Service Phase	**V**	98.7	79.7	66.5	48.8	32.2
	$\mathbf{M}_s$	99.8	96.2	95.8	96.1	98.5
	$\mathbf{M}_{cc}$	46.7	55.2	63.7	68.1	69.0
	$\mathbf{M}_{ct}$	0.0	0.0	0.0	0.0	0.0
	Interaction	-	-	-	-	-
ULS Weld	**i1**	70.4	-	-	-	-
	2	5.9	5.9	5.9	5.9	5.9
	i2	10.6	10.7	10.7	10.7	10.7
FAT	**A**	15.6	-	-	-	-
	B1	12.5	-	-	-	-
	B2	-	43.7	44.5	43.9	45.3
	C	43.5	61.9	61.6	59.8	63.0
	D	-	55.2	56.4	55.7	57.4
	E	-	44.6	45.7	45.2	46.6
	F	27.4	42.6	45.9	46.4	47.8
SLS	**Defection***	-	-	-	-	48.5

**Maximum mid span deflection using the average bending stiffness of all segments.*

Additionally, the corrugation parameters of the optimized solution is presented in Table E.7.

Table E.7: Corrugation parameters. Duplex Corrugated Web with hw $<$ 2 m.

$\mathbf{a}_1$	180	mm
$\mathbf{a}_2$	100	mm
$\mathbf{a}_3$	50	mm
$\mathbf{a}_4$	87	mm
α	30	degrees

The convergence plots of the three runs are shown in Figure E.3. To be able to visually see the change between the runs, only iterations with objectives below 120 ton are plotted. Run 1 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

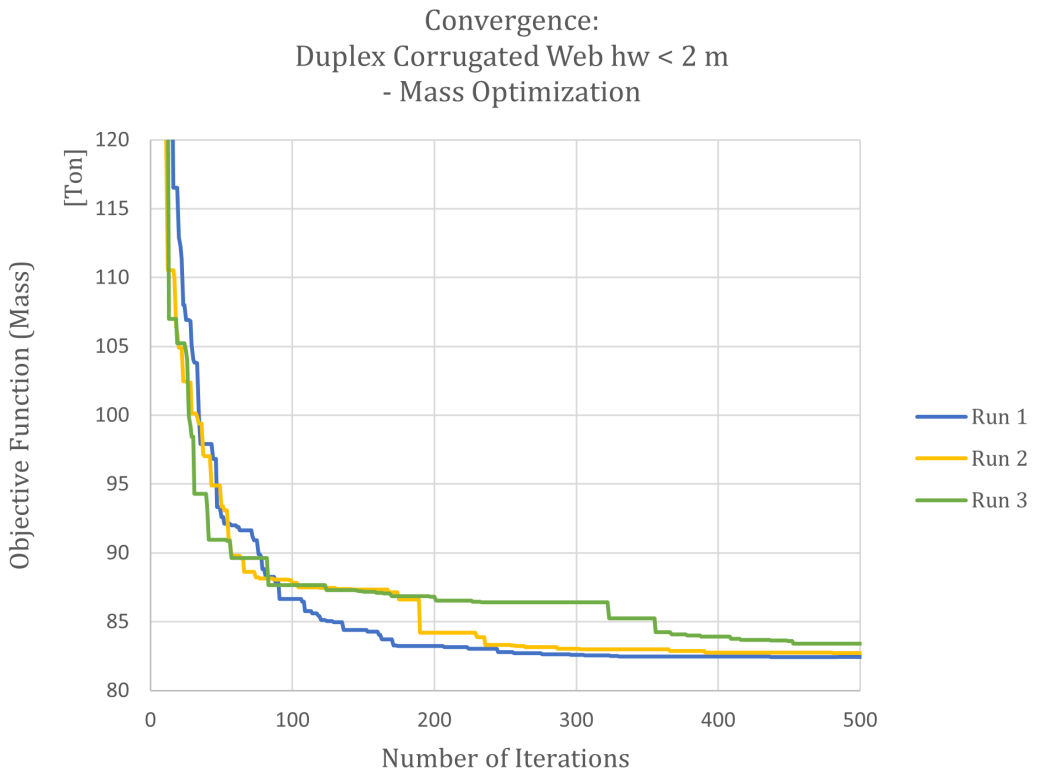


Figure E.3: Convergence plots: Duplex Corrugated Web with hw $<$ 2 m.

E.4 Duplex Corrugated Web with hw < 3 m

The design dimensions chosen by the optimization program for corrugated web girders in Duplex steel with N_{obs} = 0.05e6, ADT = 5e3 and a height domain between 1-3 meters are presented in Table E.8. Additionally, the program choose a K-truss bracing system with trusses every 3.0 meters, with element section HEA100. The resulting utilization rations are presented in Table E.9.

Table E.8: Design dimensions [mm]: Duplex Corrugated Web with hw < 3 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	2870	2870	2870	2870	2870
tw	8	8	8	8	8
bfo	400	400	400	400	400
tfo	25	25	35	35	40
bfu	1200	1200	1200	1200	1200
tfu	16	28	35	40	40
C-C studs	163	193	243	306	409
Start x-coord.	0	4550	10150	15750	18350
End x-coord.	4550	10150	15750	18350	25500

Table E.9: Utilization ratios [%]: Duplex Corrugated Web with hw < 3 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	38.0	31.2	22.9	14.5	10.6
	M	49.1	96.0	91.6	98.7	93.7
	Interaction	-	-	-	-	-
ULS Service Phase	**V**	99.4	84.0	66.0	50.4	38.2
	$\mathbf{M}_s$	98.4	97.8	99.3	93.0	99.3
	$\mathbf{M}_{cc}$	28.3	39.0	44.9	46.6	49.0
	$\mathbf{M}_{ct}$	0.0	0.0	0.0	0.0	0.0
	Interaction	-	-	-	-	-
ULS Weld	**i1**	48.6	0.0	0.0	0.0	0.0
	2	5.9	5.9	5.9	5.9	5.9
	i2	10.6	10.6	10.7	10.6	10.6
FAT	**A**	11.1	-	-	-	-
	B1	11.1	-	-	-	-
	B2	0.0	44.1	46.4	43.0	46.0
	C	39.6	60.6	62.3	57.9	62.3
	D	-	55.4	58.3	54.2	57.9
	E	-	44.6	47.0	43.7	46.7
	F	24.9	40.7	44.9	42.8	45.8
SLS	**Defection***	-	-	-	-	33.0

**Maximum mid span deflection using the average bending stiffness of all segments.*

Additionally, the corrugation parameters of the optimized solution is presented in Table E.10.

Table E.10: Corrugation parameters. Duplex Corrugated Web with hw $<$ 3 m.

$\mathbf{a}_1$	320	mm
$\mathbf{a}_2$	140	mm
$\mathbf{a}_3$	70	mm
$\mathbf{a}_4$	121	mm
α	30	degrees

The convergence plots of the three runs are shown in Figure E.4. To be able to visually see the change between the runs, only iterations with objectives below 125 kg are plotted. Run 1 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

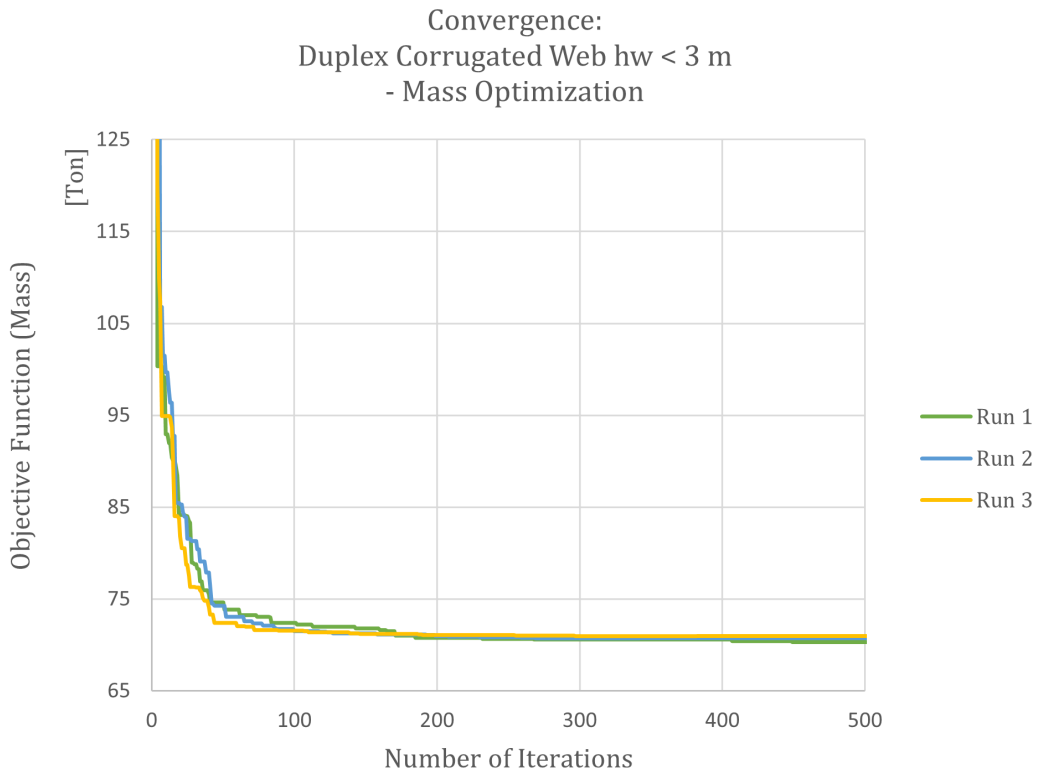


Figure E.4: Convergence plots: Duplex Corrugated Web with hw $<$ 3 m.

F

Life Cycle Assessment Optimization Results

In this appendix the design dimensions and resulting utilization rates are presented for the life cycle assessment studies presented in Chapter 5.4. This appendix also includes the convergence graphs of each alternative studied.

F.1 S355 Flat Web with hw < 2 m

The design dimensions chosen by the optimization program for flat web girders in S355 steel with $N_{obs} = 0.05e6$, ADT=5e3 and a height domain of 1-2 meters are presented in Table F.1. Additionally, the program choose a K-truss bracing system with trusses every 4.25 meters, with element section HEA100. The resulting utilization rations are presented in Table F.2.

Table F.1: Design dimensions [mm]: S355 Flat Web with hw < 2 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	1990	1990	1990	1990	1990
tw	20	20	20	20	20
bfo	550	550	550	550	550
tfo	16	25	28	35	40
bfu	1350	1350	1350	1350	1350
tfu	16	28	40	45	50
C-C Studs	103	137	173	221	311
Start x-coord.	0	6050	10150	15250	18350
End x-coord.	6050	10150	15250	18350	25500

Table F.2: Utilization ratios [%]: S355 Flat Web with hw < 2 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	35,8	27,3	21,5	14,4	10,0
	M	93,3	80,1	95,2	96,5	95,6
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Service Phase	**V**	89,9	73,5	60,7	47,1	34,8
	$\mathbf{M}_s$	95,8	99,8	100,0	99,6	98,9
	$\mathbf{M}_{cc}$	33,0	42,1	50,9	53,5	56,1
	$\mathbf{M}_{ct}$	0,0	0,0	0,0	0,0	0,0
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Weld	**i1**	50,8	-	-	-	-
	2	7,8	7,8	7,8	7,8	7,8
	i2	12,6	12,7	12,7	12,7	12,7
FAT	**A**	9,5	-	-	-	-
	B1	3,8	-	-	-	-
	B2	-	40,9	40,3	39,3	38,5
	C	35,2	52,2	51,4	50,2	49,5
	D	-	51,5	50,9	49,8	48,8
	E	-	41,5	41,2	40,4	39,6
	F	24,9	37,9	40,4	40,5	40,6
SLS	**Defection***	-	-	-	-	40,7

**Maximum mid span deflection using the average bending stiffness of all segments.*

The convergence plots of the three runs are shown in Figure F.1. To be able to visually see the change between the runs, only iterations with objectives below 300 CO_2-eq are plotted. Run 2 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

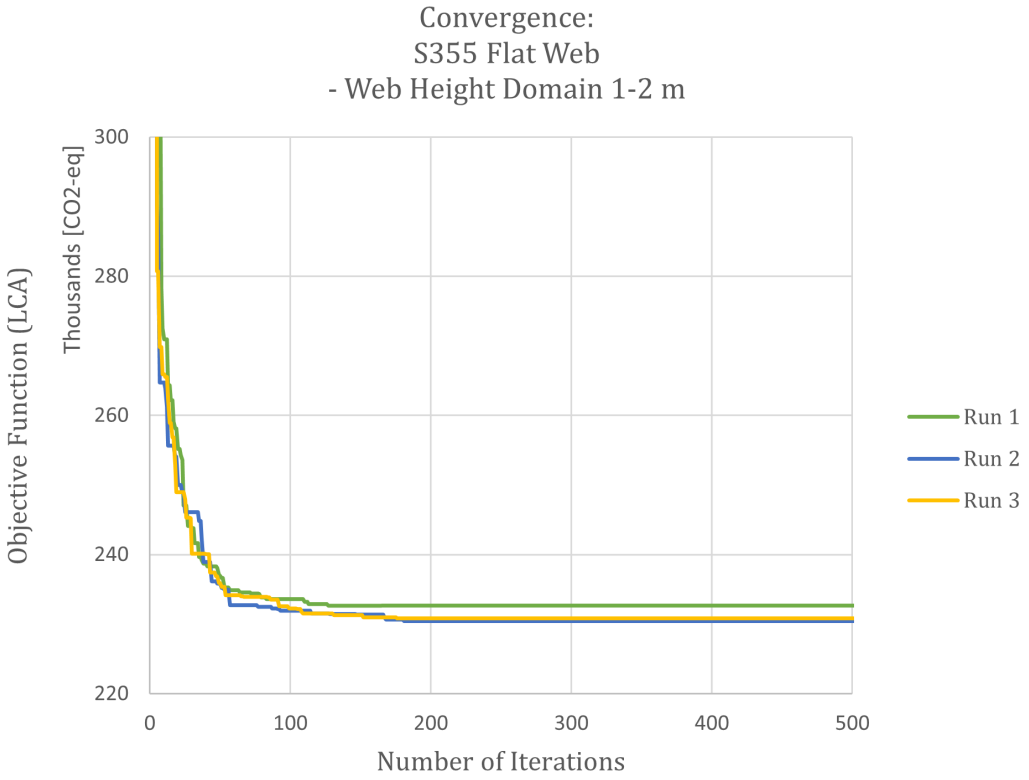


Figure F.1: Convergence plots: S355 Flat Web hw $<$ 2 m.

F.2 S355 Flat Web with hw < 3 m

The design dimensions chosen by the optimization program for flat web girders in S355 steel with $N_{obs} = 0.05e6$, ADT = 5e3 and a height domain between 1-3 meters are presented in Table F.3. Additionally, the program choose a K-truss bracing system with trusses every 4.25 meters, with element section HEA100. The resulting utilization rations are presented in Table F.4.

Table F.3: Design dimensions [mm]: S355 Flat Web with hw < 3 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	2880	2880	2880	2880	2880
tw	18	18	18	18	18
bfo	500	500	500	500	500
tfo	16	25	25	25	28
bfu	1150	1150	1150	1150	1150
tfu	16	18	28	35	35
C-C Studs	144	189	237	313	424
Start x-coord.	0	7050	10650	15750	19350
End x-coord.	7050	10650	15750	19350	25500

Table F.4: Utilization ratios [%]: S355 Flat Web with hw < 3 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	35,8	25,9	20,9	13,7	8,6
	M	98,4	72,4	91,5	100,0	98,6
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Service Phase	**V**	90,2	71,7	59,9	45,9	33,0
	M_s	77,7	99,9	99,2	94,2	99,6
	M_{cc}	24,0	31,2	37,2	39,4	41,4
	M_{ct}	0,0	0,0	0,0	0,0	0,0
	Interaction	0,0	0,0	0,0	0,0	0,0
ULS Weld	**i1**	30,2	-	-	-	-
	2	7,8	7,8	7,8	7,8	7,8
	i2	12,6	12,7	12,7	12,7	12,7
FAT	**A**	5,6	-	-	-	-
	B1	2,5	-	-	-	-
	B2	-	41,0	40,3	37,4	39,3
	C	28,2	51,7	50,8	47,2	49,8
	D	-	51,5	50,6	47,1	49,5
	E	-	41,3	40,7	37,9	39,9
	F	20,0	36,9	37,2	36,2	38,1
SLS	**Defection***	-	-	-	-	26,0

**Maximum mid span deflection using the average bending stiffness of all segments.*

The convergence plots of the three runs are shown in Figure F.2. To be able to visually see the change between the runs, only iterations with objectives below 300 CO_2-eq are plotted. Run 3 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

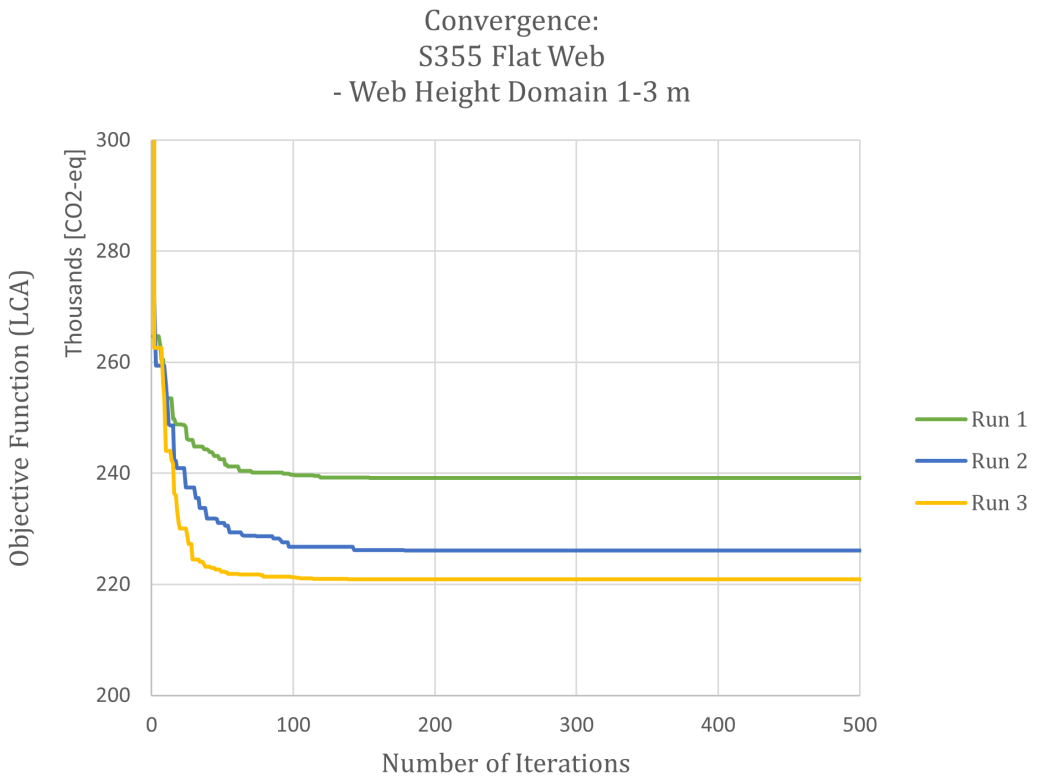


Figure F.2: Convergence plots: S355 Flat Web hw $<$ 3 m.

F.3 Duplex Corrugated Web with hw < 2 m

The design dimensions chosen by the optimization program for corrugated web girders in Duplex steel with $N_{obs} = 0.05e6$, ADT=5e3 and a height domain between 1-2 meters are presented in Table F.5. Additionally, the program choose a bracing system with I-beams every 3.4 meters of section HEA120. The resulting utilization rations are presented in Table F.6.

Table F.5: Design dimensions [mm]: Duplex Corrugated Web with hw < 2 m.

Segment	**1**	**2**	**3**	**4**	**5**
hw	1890	1890	1890	1890	1890
tw	10	10	10	10	10
bfo	450	450	450	450	450
tfo	25	28	45	50	60
bfu	1400	1400	1400	1400	1400
tfu	20	30	45	50	50
C-C Studs	111	129	162	236	332
Start x-coord.	0	4550	8150	16250	20850
End x-coord.	4550	8150	16250	20850	25500

Table F.6: Utilization ratios [%]: Duplex Corrugated Web with hw < 2 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	38.6	31.7	26.2	14.0	7.0
	M	67.7	99.6	99.4	99.3	85.4
	Interaction	-	-	-	-	-
ULS Service Phase	**V**	99.2	85.3	70.3	47.0	31.9
	$\mathbf{M}_s$	96.8	97.0	96.9	95.8	97.9
	$\mathbf{M}_{cc}$	40.9	51.0	64.2	67.8	67.9
	$\mathbf{M}_{ct}$	0.0	0.0	0.0	0.0	0.0
	Interaction	-	-	-	-	-
ULS Weld	**i1**	71.6	-	-	-	-
	2	5.9	5.9	5.9	5.9	5.9
	i2	10.6	10.6	10.8	10.7	10.7
FAT	**A**	16.2	-	-	-	-
	B1	13.0	-	-	-	-
	B2	-	43.4	45.3	44.0	45.3
	C	41.0	62.0	62.4	59.9	63.0
	D	-	54.7	57.4	55.9	57.5
	E	-	44.1	46.5	45.3	46.6
	F	25.0	40.8	46.7	46.5	47.8
SLS	**Defection***	-	-	-	-	51.7

**Maximum mid span deflection using the average bending stiffness of all segments.*

Additionally, the corrugation parameters of the optimized solution is presented in Table F.7.

Table F.7: Corrugation parameters. Duplex Corrugated Web with hw $<$ 2 m.

$\mathbf{a}_1$	170	mm
$\mathbf{a}_2$	100	mm
$\mathbf{a}_3$	50	mm
$\mathbf{a}_4$	87	mm
α	30	degrees

The convergence plots of the three runs are shown in Figure F.3. To be able to visually see the change between the runs, only iterations with objectives below 400 CO_2-eq are plotted. Run 3 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

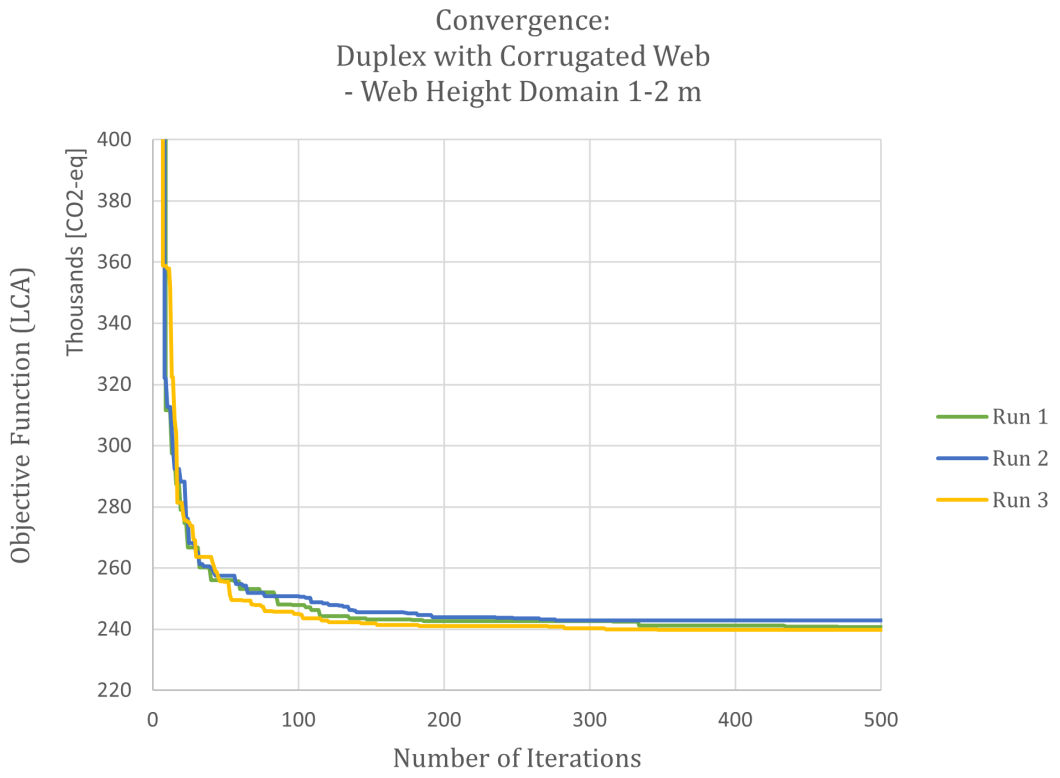


Figure F.3: Convergence plots: Duplex Corrugated Web with hw $<$ 2 m.

F.4 Duplex Corrugated Web with hw < 3 m

The design dimensions chosen by the optimization program for corrugated web girders in Duplex steel with N_{obs} = 0.05e6, ADT = 5e3 and a height domain between 1-3 meters are presented in Table F.8. Additionally, the program choose a K-truss bracing system with trusses every 4.25 meters, with element section HEA100. The resulting utilization rations are presented in Table F.9.

Table F.8: Design dimensions [mm]: Duplex Corrugated Web with hw < 3 m.

Segment	**1**	**2**	**3**	**4***	**5***
hw	2880	2880	2880	2880	2880
tw	8	8	8	8	8
bfo	500	500	500	500	500
tfo	20	28	30	35	35
bfu	1200	1200	1200	1200	1200
tfu	16	25	35	40	40
C-C Studs	159	184	233	329	448
Start x-coord.	0	4050	8650	15750	20350
End x-coord.	4050	8650	15750	20350	25500

**Same dimensions for segment 4 and 5 means no welding between these segments.*

Table F.9: Utilization ratios [%]: Duplex Corrugated Web with hw < 3 m.

	Segment	**1**	**2**	**3**	**4**	**5**
ULS Construction Phase	**V**	37.9	31.9	25.1	14.5	7.7
	M	93.4	95.8	97.9	94.1	98.1
	Interaction	-	-	-	-	-
ULS Service Phase	**V**	99.5	86.1	69.5	48.7	33.2
	$\mathbf{M}_s$	99.6	99.2	99.2	95.7	99.2
	$\mathbf{M}_{cc}$	29.7	37.1	44.7	46.9	48.6
	$\mathbf{M}_{ct}$	0.0	0.0	0.0	0.0	0.0
	Interaction	-	-	-	-	-
ULS Weld	**i1**	48.7	-	-	-	-
	2	5.9	5.9	5.9	5.9	5.9
	i2	10.6	10.6	10.7	10.7	10.6
FAT	**A**	11.2	-	-	-	-
	B1	11.2	-	-	-	-
	B2	-	43.6	46.3	44.2	45.9
	C	36.5	60.3	62.2	59.0	62.2
	D	-	54.8	58.3	55.7	57.8
	E	-	44.0	46.9	44.9	46.6
	F	22.4	39.3	44.8	44.1	45.7
SLS	**Deflection**	-	-	-	-	33.5

**Maximum mid span deflection using the average bending stiffness of all segments.*

Additionally, the corrugation parameters of the optimized solution is presented in Table F.10.

Table F.10: Corrugation parameters. Duplex Corrugated Web with hw $<$ 3 m.

$\mathbf{a}_1$	320	mm
$\mathbf{a}_2$	140	mm
$\mathbf{a}_3$	70	mm
$\mathbf{a}_4$	121	mm
α	30	degrees

The convergence plots of the three runs are shown in Figure F.4. To be able to visually see the change between the runs, only iterations with objectives below 300 CO_2-eq are plotted. Run 2 generated the lowest cost and was therefore presented in the results above and in Chapter 5.

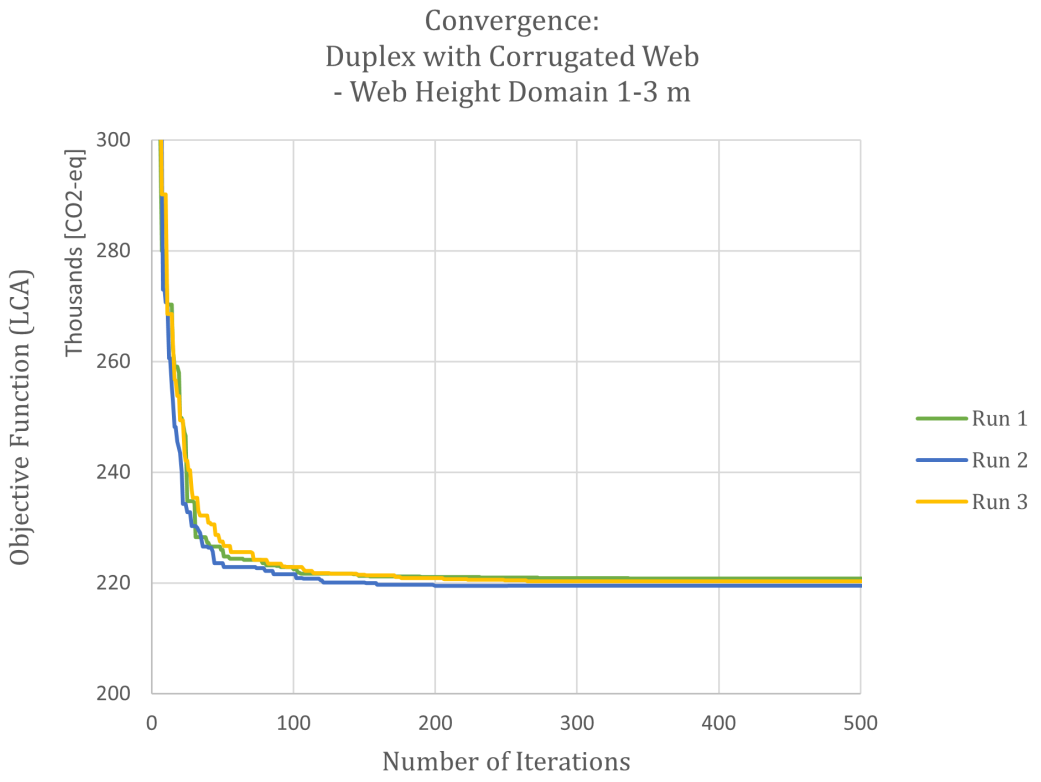


Figure F.4: Convergence plots: Duplex Corrugated Web with hw $<$ 3 m.

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY