NTNU – Trondheim
Norwegian University of
Science and Technology

Steel weight optimisation with respect to stiffener spacing and plate thickness of mid ship structure for cargo vessels

Lina Marie Storås
Stokkeland

Marine Technology
Submission date: June 2013
Supervisor: Jørgen Amdahl, IMT
Co-supervisor: Evelin Tankovic, Rolls Royce Marine
Sören Ehlers, IMT

Norwegian University of Science and Technology
Department of Marine Technology

NTNU
Norges teknisk-naturvitenskapelige universitet
Institutt for marin teknikk

MASTER THESIS 2013
for
Stud. techn. Lina Stokkeland

Steel weight optimisation with respect to stiffener spacing and plate thickness of mid ship structure for cargo vessels

Optimalsiering av stålvekt med hensyn på stiveravstand til midtskips tverrsnitt for lasteskip

For the time being, there are no standardised and recognised methods in RRM STM for optimising the steel weight with regard to primary and secondary spacing of the structure in the early design phase. The spacing is decided based on previous ships and experience, and only a few combinations are considered before decisions are made. The purpose of this work is to develop a tool based on common software to investigate and compare steel weight of cargo area/part of cargo area for merchant ship types, for different structural lay-out.

This project has been proposed by the structural department of Rolls Royce Marine - Ship Technology Merchant Vessels in Ålesund, Norway. RRM's expectation for the outcome of the project is to get a tool which will be used for its specified range of application for input to early design arrangements and steel weight estimation.

It is proposed to carry out the work in the following steps:

1. Formulate the optimization problem and develop a detailed flow chart of the analysis procedure. Develop cost functions that can facilitate optimization with respect to fabrication costs in addition to weight. Formulate the constraints of optimization, e.g. DnV rule requirements to plate thickness, stiffener section modulus and hull girder modulus. Define the load conditions that need to be considered. It is also necessary to develop an empirical relationship between section modulus and stiffener area for relevant stiffeners, using a representative effective flange.
2. The relevant optimization constraints shall be defined (e.g. ship classification rule requirements to structural scantlings) and the selection of free variables including any practical limitations of the variation range shall be discussed. Limitation of the task with regard to ship type(s) shall be defined.
3. Develop a MATLAB program which, by use of iteration processes, can optimize the structural weight/production costs for a chosen ship type, main perpendiculars, number of decks, deck loads. The output of the program shall be optimum stiffener spacing and plate thickness of the structure in an early design phase as well as steel weight and production costs. For the optimum spacing required dimensions for the longitudinal strength members shall be given. The governing constraints, i.e. active rule requirements, or similar for the various components shall be given.
4. Conclusions and suggestions for further work

Prerequisite:

RRM will provide set of as built ships with steel weight and arrangement for comparison / input.

Literature studies of specific topics relevant to the thesis work may be included.

The work scope may prove to be larger than initially anticipated. Subject to approval from the supervisors, topics may be deleted from the list above or reduced in extent.

In the thesis the candidate shall present her personal contribution to the resolution of problems within the scope of the thesis work.

Theories and conclusions should be based on mathematical derivations and/or logic reasoning identifying the various steps in the deduction.

The candidate should utilise the existing possibilities for obtaining relevant literature.

Thesis format
The thesis should be organised in a rational manner to give a clear exposition of results, assessments, and conclusions. The text should be brief and to the point, with a clear language. Telegraphic language should be avoided.

The thesis shall contain the following elements: A text defining the scope, preface, list of contents, summary, main body of thesis, conclusions with recommendations for further work, list of symbols and acronyms, references and (optional) appendices. All figures, tables and equations shall be numerated.

The supervisors may require that the candidate, in an early stage of the work, present a written plan for the completion of the work. The plan should include a budget for the use of computer and laboratory resources, which will be charged to the department. Overruns shall be reported to the supervisors.

The original contribution of the candidate and material taken from other sources shall be clearly defined. Work from other sources shall be properly referenced using an acknowledged referencing system.

The report shall be submitted in two copies:
- Signed by the candidate
- The text defining the scope included
- In bound volume(s)
- Drawings and/or computer prints that cannot be bound should be organised in a separate folder.

Ownership
NTNU has according to the present rules the ownership of the thesis. Any use of the thesis has to be approved by NTNU (or external partner when this applies). The department has the right to use the thesis as if a NTNU employee carried out the work, if nothing else has been agreed in advance.

Thesis supervisor
Prof. Jørgen Amdahl and Prof. Sören Ehlers

Contact person at RRM:
Evelin Tankovic

Deadline: June 10, 2013

Trondheim, June 10, 2013

Jørgen Amdahl

Abstract

When building a ship, the weight and cost are important factors. By reducing weight there will be more cargo capacity which ensures the income for the ship owner. On the other hand, creating light weight structure often increase the cost of building, and there may be limits for what the buyer will pay for a ship to be built. Ship design is often based on experience and adapting previous design to the project at hand. This procedure may end with both the weight and the cost of the ship becoming higher than needed.

This thesis aims to create a early design tool the designers can use to find the plate thickness and distance between stiffeners which will give the lowest weight, cost or find a solution which reduces on both the former objectives. The program was done in MATLAB using a Particle Swarm Optimisation algorithm. The main problem was making the code recognise the structure of the cross section and use this information on the restrictions set by the DNV, Det Norske Veritas, rules for ship over 100 meters. Also handling and simplifying the DNV rules was a challenge. The different goals for the optimisation was handled by representing both a weight and cost functions in a weighted objective function.

With the limitations set by code and rule simplification, the program was able to reduce both the weight and cost during a trial with tree object functions, one with pure cost optimisation, one with pure weight optimisation and one trial with a combination of the two. The labor cost was dominating in all the optimisations and the cost parameters should, in further work, be changed to more correct data. In all the optimisation there was a reduction in the cross section area and therefore a reduction in the weight.

All in all, the program provides a spread of possible solutions the designer can work from in the early design phase. It provides a good start point for the individual project, and may help avoid making the design overly conservative when more design conditions are to be added. To improve the program, constraints regarding the build-ability can be imposed to prevent a large spread in the thickness of the plates in the cross section. Also the constraint of buckling capacity should be included to allow the designer to experiment with higher distances between the stiffeners and avoid critical stress.

Sammendrag

I skipsbygging er vekt og pris viktige faktorer. Ved å redusere vekten på skipet vill det være mulig å frakte mer last, og via dette øke inntektene til skipet. Problemet med å bygge lette skip er at den lette strukturen vil trenge mer arbeid å sette sammen. Selv om kjøperen ønsker et lett skip, vil det finnes grenser for hvor mye man er villig å betale for dette resultatet. Når et skip designes blir ofte dimensjoner valgt på grunnlag av tidligere lignende prosjekter og designerens egne erfaringer. Denne metoden kan presse opp både kostnader og vekt.

Denne master oppgaven sikter på å lage et tidlig design verktøy for å finne plate tykkelser og avstanden mellom stivere som vil gi lavest vekt, bygge kostnader eller finne dimensjonen som vil forbedre begge de øvrige målene. Programmet ble laget i MATLAB og det ble brukt en partikkel sverm algoritme i leteprosessen. Hoved problemet i koden var å sende videre den viktige informasjonen om tverrsnitts geometrien til regelverk kravene hentet fra DNV sine regler for skip over 100 meter. Det å behandle og bruke DNV reglene viste seg å være mer vanskelig enn antatt. For å slå sammen målene om vekt og kost optimalisering ble en vekt funksjon brukt.

Med grunnlag i begrensningene gjort, var programmet i stand til å finne tall som både reduserte vekt og kostnader. Resultatene viste at kostnadene for arbeid var veldig høye for alle optimaliseringene, noe som indikerer at parametrene i denne formelen kan forbedres. I alle optimaliseringsforsøkene var det reduksjon i tverrsnittsarealet og derfor en reduksjon i vekt.

Programmet levere varierte løsningsforslag som kan dras nytte av som en design plattform i design fasen. For å forbedre programmet bør blant annet bruddkapasitet legges til i tillegg til flere restriksjoner som kan underbygge gode designmessige valg i optimaliseringen.

Acknowledgements

I would like to thank my supervisor Jørgen Amdahl for all his suggestions and good advise. Also due a large thanks is my co-supervisor Sören Ehlers for being a great resource in handling the optimisation theory, encouraging me and most of all making me feel smart. In addition I have through the whole of this project been able to rely on precise and good answers from my contact in Rolls Royce, Evelin Tankovic, whom has always been helpful and attentive.

Lina Stokkeland

Trondheim, 10th June 2013

Contents

List of Figures

List of Tables

Abbreviation

a	Distance between points of zero bending moment
A_i	Area of an element
a_{ij}	Constraint limit
a_s	General strength demand
A_s	Area of stiffener
a_v	Vertical acceleration parameter
B	Breadth of the ship
b_a	General actual loading
b_e	Effective flange breadth
b_f	Flange breadth
b_j	Constraint limit
c	Cognitive constant
c_i	Width coefficient
C	Cost of section based on the variables
C_B	Block coefficient
C_c	Cost of consumables
c_i	Weight factor for design variable
$C_i init$	Cost of initial example
C_l	Labour cost per meter
C_{plate}	Cost per kg of steel plates
C_{steel}	Cost of plates and stiffeners in raw material
$C_{stiffener}$	Cost per kg of stiffeners
C_w	Wave coefficient
C_{wu}	Wave coefficient restricted
C^0_{8x}	Cost of consumables per meter
d_i	Distance local center of mass to lowest point
DNV	Det Norske Veritas
E	Young modulus
E_0	Preferred thickness of plate
E_{0x}	Preferred thickness of stiffener
e_x	Distance from center of stiffener to bottom
f(x)	Object function dependent on variables x
f_1	Material factor of steel
g(x)	Constraint function dependent on variables x
H_{hp}	Height of stiffener
HP	Bulb profile
i	Numerator for panels
I_e	Moment of inertia with effective flange

Continued on next page

Table 1 – *Continued from previous page*

I_s	Moment of inertia of stiffener
j	Numerator for constraints
k	Buckling coefficient
K	Coefficient connecting work load in hours to material processed
k_t	Empirical coefficient
L	Length of the ship
L_1	Length of the ship or max 300 m
l_s	Length of stiffener
m	Meters
MATLAB	Matrix laboratory, numerical computing environment
mm	Millimetres
M_s	Still water moment
M_w	Moments induced by waves
p	Local pressure on panel
P_i	Position of best particle swarm
P_g	Position of the swarms best particle
P_4^0	Workload for welding longitudinal stiffener
P_{10}^0	Workload required to prepare 1 m^2 plate
r	Random number between 0 and 1
s	Distance between stiffeners, a design variable
Scantings	Thickness and stiffener distance
SUMT	Sequential Unconstrained Minimisation Technique
t	Plate thickness, a design variable
t_a	Added thickness from smeared thickness
t_c	Corrosion addition to the thickness
t_l	Thickness parameter dependent on panel location
t_{min}	Minimum rule based plate thickness
t_{d1}	Minimum thickness demand dependent on s
t_{d2}	Minimum thickness demand, empirical formula
t_p	Plate thickness without corrosion addition
t_w	Thickness of web
v_i	Velocity of particle at current position
v_{i+1}	Velocity of particle at the next position
w	Inertia coefficient
W	Weight of section based on the variables
W_F	Weight of the frame divided on l_s
w_h	Height of web
W_i	Weight of example
W_p	Weight of the plates

Continued on next page

Table 1 – *Continued from previous page*

W_s	Weight of the stiffeners
x	Design variable
x_i	Location if the particle
x_{i+i}	Location of the particle in the next step
Z	Optimal value
Z_{built}	Section modulus of the cross section based on the design variables
Z_{cs}	Section modulus demand on the whole cross section
Z_e	Section modulus calculated with effective flange
Z_{loc}	Section modulus demand for stiffener
z_n	Distance to neutral axis
α	Weighing parameter for cost and weight optimisation
η	Usage factor
σ	Bending stress
σ_a	Actual compressive stress
σ_c	Critical compressive buckling stress
ρ_s	Steel density
δC_{8x}	Cost for deviation from the preferred thickness of stiffener
$\delta P4$	Cost for deviation from the preferred thickness of stiffener
$\delta P10$	Cost for deviation from the preferred thickness of plate
ν	Poisson's ratio

Chapter 1

Introduction

When building ships, there are requirements to the structural integrity that must be fulfilled to be able to get a safe ship that can handle the loads afflicted on it. In addition the owner may have special criteria the ship, for example that it should have a specific size and be able to load a certain amount of cargo. On top of this, the yard must make a profit on delivering the ship. To fulfil all these limitations one can apply optimisation to find the best related structure. Optimisation theory was initially developed to distribute resources to a task and ensure overall profit, this can easily be related to the ship building industry. In the 60's there was optimisation programs available for ships which aimed to provide designers with the best cross section scantlings, plate thickness and distance between the stiffener, to obtain the best lightweight hull possible. The driving force at this time was the oil and gas market's need for tankers. In 1970, during the oil criss, the need for tankers disappeared, and with it a lot of the programs for optimisation where forgotten or discarded, and they are today not possible to find. Today optimisation programs are again produced, with increasing complexity, aiming to include the complex strength calculation in the early design phase, as it is here the design is at it's freest.

During a summer job, I was set to find the cross section scantlings of a ship. The task was time consuming and tedious, and lead me to believe that there should be a way to solve this task in a better way. Later, when I approached Rolls Royce with the idea of a optimisation program, they where very interested. For Royce Merchant AS, the possibility to get the the scantlings which would give them the lowest weight was the most interesting. As the company designs cargo ships, their design target is to make the ship as light as possible, to allow the highest amount of cargo. At the moment their early design process is dominated by routine and the experience of their designers.

1.1 Goal

This thesis aims to make a optimisation program in order to optimise the cross section scantlings to get the lowest weight, but also the lowest cost because it is interesting to see the connection to the design, and because it is a important factor in making ships. In addition it should be possible to make a combination and optimise for both weight and cost at the same time. The program is made for optimising a section between two frames in the cargo area of a dry cargo ship. The algorithm used is a Particle Swarm Optimisation and the program will be written in MATLAB. As cargo ships most often have a traditional cargo area, the strength requirements are set by the DNV, Det Norske Veritas, rules for ship over 100 meter. Since the program is made as an optimisation tool for early design for Rolls Royce, the program should be user friendly and easy to understand.

1.2 Background

Optimisation methods have developed in many directions over the years, making it possible to solve more and more complex problems. The optimisation problems contains a object function, a that function represents the element which desires improvement. In addition there is a constraint function, that contains the restrictions which keeps the solution within a feasible area. In the following text, some ways to solve ship structure related problems, developed through the years, are presented. [Hiller and Lieberman, 2010]

In the 60's there was a high demand for oil transport, making it profitable for the designers to create ships with better cargo capacity. [Carlsen and Kavlie, 1975] describes the program INDETS, a weight optimisation program that used modules to optimise longitudinal scantlings, transverse frames and transverse bulkheads. The requirements set for the optimisation was based on the DNV rules, but also girder system analyses were done to find the allowable stress. Because of the problem with solving nonlinear constraints, the best way to solve the problem was seen to transforms the original problem into a series of unconstrained optimisation problems and use penalty factors through a Sequential Unconstrained Minimisation Technique, SUMT, to find the optimal solution. The main focus of the optimisation was to decrease the weight to allow more cargo. The program, along with many others, disappeared in the 70's as a result of low interest for weight optimisation when the tanker marked crashed.

Even though weight optimisation have always seemed to be a popular optimisation goal, there was after a time also developed optimisations with the goal of finding

the most cost efficient structure. In the 80's optimisation were once again in the wind, and in [Hughes et al., 1980] a method was developed to provide the constraints from finite element into a redesign program which could handle a large amount of nonlinear constraints. By using a direct approach to calculate the structural demands on the structure, the program did not need to use the classification societies rules in the design process. The goal of stepping away from the classification societies rules was to avoid the conservative margins built into the rules. In optimising the ship, one and one plate section with stiffeners was considered at the time, and later corrected for global demands. To account for the nonlinear constraints, a Taylor series was used to make the problem able to fit into the Dual Revised Simplex Method.

The method of starting from scratch and using direct strength analysis to find the strength demand is called rational based design and is especially good for new ships as their structure may differ from traditional design. The classification societies rules are good for use on ships with simple and traditional layouts, as they are the types the rules in the first place where made to include. In [Karr et al., 2012] the two methods of designing a ship, rule based or rational based, are tested up against each other. The paper show that the rule based and the rational based optimisation calculates similar bending stresses, but end up with different scantlings. The rational based optimisation was naturally able to consider complex failure models. The failure analysis is something which must be done in addition when using pure class rules as constraints for complex ships.

For a long time the main focus of the structural optimisation was on weight, but [Rigo, 2001] developed a new method of finding the cost of the ship, accounting for labour, consumables and material for each component in a section. To tackle the nonlinearity in the optimisation, convex linearisation was used to approximate both the object function and the constrain function. In each iteration the functions were parted up into smaller problems which were easy to solve. The solution was on the conservative side, but for each iteration the deviance got smaller until it converges on the design optimum. Also this program is based on rational design.

Taking the optimisation a step further, and bringing more knowledge and experience into the early design stage to improve the design is something that was focused in [Ehlers et al., 2010]. This paper presents a multi objective optimisation method, with focus on decision support. The importance of including more than weight or cost in the early design tool is emphasised. The multi-objective optimisation looks at weight, cost and fatigue life and sees their influence on the ship structure. The optimisation is based on rational design, although using loading parameters from the DNV rules. In the paper, the decision support algorithm is made to help identify the best solution for both the owner and the yard and to help the user

choose the right constraints. By including features that will affect the ship in the future in the early design phase, one saves money because it is done in the phase where the solutions easily can be implemented.

In [Ehlers, 2012] the Particle Swarm Optimisation algorithm is introduced to find the structure with the best crash-worthiness, which means the highest energy per mass ratio, to get a light structure which can take high crash loads. The PSO handles a great amount of constraints, and only adapts them by normalisation of the values before they are tested. The optimisation starts by generating feasible solutions, particles, of the design variables chosen and use these combinations as a start base, called a population. Through each iteration, the best value of the population and the individual particles are stored and used for developing the solution further.

In [Nerem, 1990] a computer program for optimisation of a midship section of a fishing vessel was made using FORTRAN. The program was built such that after finding the limits set by the restrictions, the DNV rules, the weight and section modulus for every possible combination was calculated. This made the optimal solution the combination with the lowest weight and high enough cross section section modulus.

Considering the optimisation of ship structure done through the years, one can see that there are many possible ways of handling nonliear problems. There is also a large variety in the user influence on the different programs. In the latest research the emphasis is on that the user can influence the optimisation in more ways by implementing concerns usually handled after the initial design phase. The choice of restrictions should be based on the ship in question. If it is a new and innovative design, a rational design approach is useful as the normal class rules may not apply for the special ship. On the other hand, a normal ship design might not loose anything by using the long tried and recognized class rules to dimensioning the ship. The class society will either way always have to check the structure.

1.3 Thesis layout

In the following text, the constraints and object function of the optimisation program will be introduced. The the optimisation algorithm chosen, Particle Swarm Optimisation, will be presented before the structure of the program is explained. After this, the scantlings and optimum solutions produced by the optimisation on a case study is evaluated. Ultimately the conclusions and further work is presented.

Chapter 2

The optimisation of midship cross section scantlings

In an optimisation one seeks a combination of parameters, often subjected to a set of constraints, which aims to find the best value of the objective function, the function representing a value the user wants to make better through optimisation.

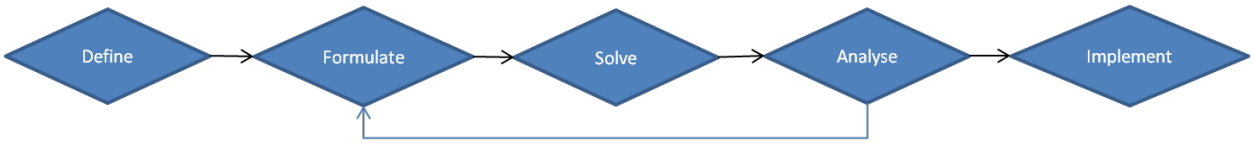


Figure 2.1: Flow chart of optimization

When one seeks to construct a optimisation problem, finding the variables and constraints which affect the object function, one should follow the flow chart for the problem solving process in Figure 2.1. Notice the feedback from the analysis of the solution, where one might go back to the description of the problem and adjust for errors so that the effect of the choices are presented in a correct way. Relating this to the ship industry, the definition of the optimization problem can be set by the buyer, depending on if he wants a low cost ship, or a ship with low weight. The formulation of the problem is left to the designers, whom know the relations between the structural demands and the costs involved in different choices. When solving the problem, one should find a ship which one can compare the improvement of weight and cost, this in order to keep track of the improvement.

During the analyses of the optimisation problem, the designers should involve the ship yard to implement constraints which is important to make the building process

easier. To add the optimisation process in ship design, one should keep in mind that the solution provided by the optimisation needs to be processed by people with experience in ship building, this is because the optimisation might provide a solution with low weight or cost, but not following the preferred building etiquette.

Going into the mathematical formulation of the optimization problem, one will have an object function $f(x)$, here exemplified by the linear equation given by Equation (2.1), this equation combined with the constraint function in Equation (2.3) represents a optimisation problem which may be optimize. The optimisation can either a maximisation or minimisation problem, meaning one can search for the highest or lowest value allowed. The function in Equation (2.1),depends on design variables x_i, which is weighted by the weight factors c_i, these factors decide the impact of the variables on the result. The problem constraints $g(x)$, restricts the solution to a feasible area, dependent on the limit value b_j and the weight parameter a_{ij}. In Figure (2.2), one can see a simple graphical presentation of a linear programming problem, with constrains of the type $g(x) \leq b$. The feasible area for this problem is shown at the coloured area under or to the left of the plotted functions. Any combination of x_1 and x_2 within the feasible area will give a feasible solution of the object function where all of the restraints are satisfied. The design variables will in the optimisation be changed to find the optimal solution value, here defined as Z. Setting this in relation to ship building, the constraints will be related to the rules regarding the ships structural integrity, and the object function will be the lowest cost or weight possible to obtain. [Hiller and Lieberman, 2010]

$$\max Z = f(x) = \sum_i c_i \cdot x_i \tag{2.1}$$

$$(or \max -Z = -f(x) = \sum_i -c_i \cdot x_i) \tag{2.2}$$

$$g_j(x) = \sum_i a_{ij} \cdot x_i \leq b_j \tag{2.3}$$

$$x_i \geq 0$$

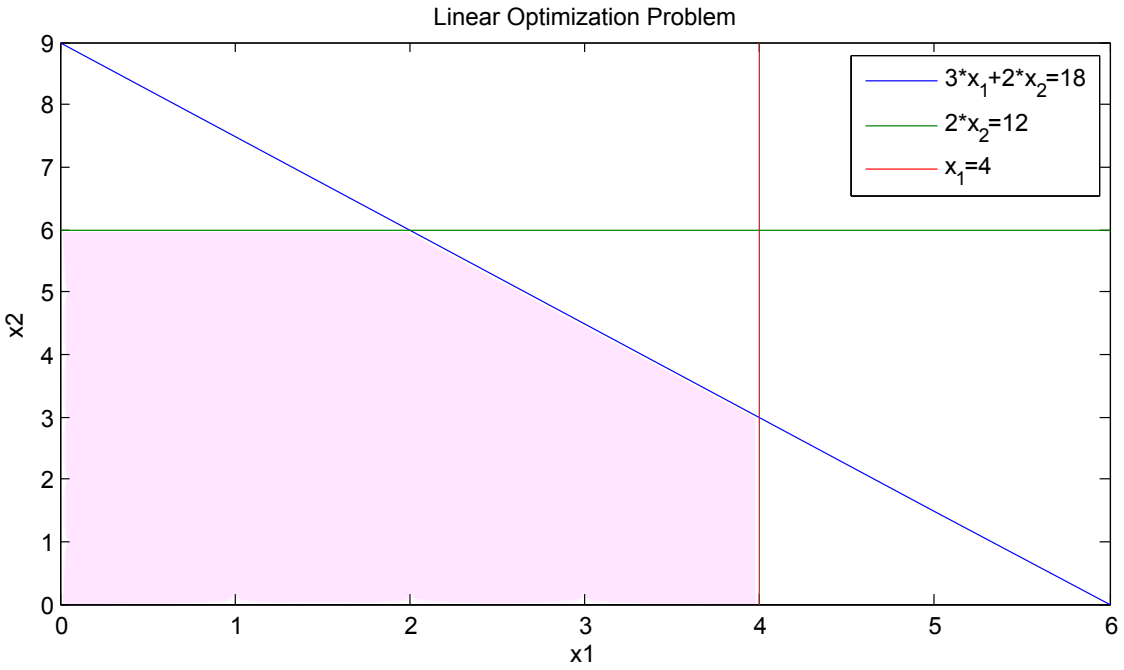
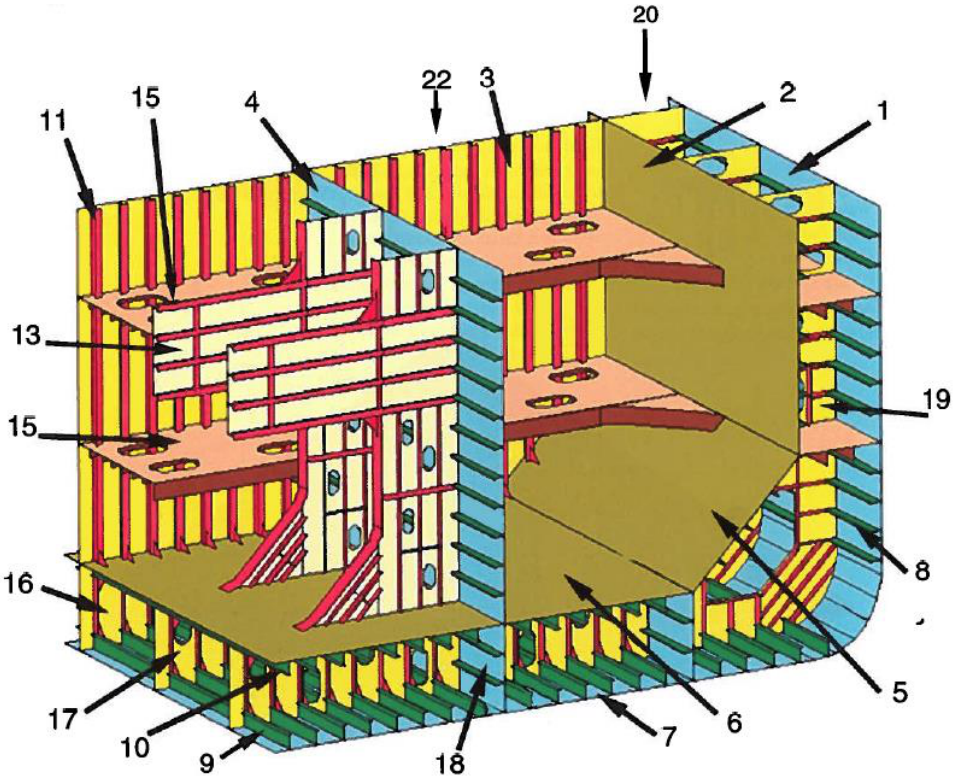


Figure 2.2: Graphical presentation of linear programming problem. Feasible area shown as coloured area.



Plating	Stiffenings on the plating	Plate-stiffeners	Holds
1. Shell	8. Side longitudinals	13 Tie beam or cross-tie	20. Wing ballast tank
2. Longitudinal bulkhead (of the inner hull)	9. Bottom frame / Longitudinal	14. Stringer	21. Double bottom
3. Transverse bulkhead	10. Inner bottom longitudinal	15. Stringer deck	22. Cargo tank
4. Longitudinal bulkhead	11. Bulkhead stiffener	16. Watertight floor	
5. Lower hopper	12. Stiffener with brackets	17. Full floor	
6. Tanktop		18. Watertight side keelson	
7. Bottom		19. Web frame	

Figure 2.3: Drawing of double-bottom tanker with longitudinal framing system. [Dokkum, 2003]

The cross section in Figure (2.3), is built of of multiple panels. A panel is in this thesis used as a name for a plate with stiffeners attached. Each panel element has a length in the y- or z- plane, a thickness and stiffeners attached. On each panel the number and size of the stiffeners may wary.

In defining the optimisation it must be decided what is user supplied information and which parameters the program should optimise. Assuming that the geometry of the cross section in the cargo area will be constant through the cargo area, and that the designers will know the basic dimensions of the ship, like length, breadth,

draught, depth and block coefficient, this is easily provided input. Looking at Figure(2.3), the section between the transverse stiffening of the ship, enumerated as 17 in the bottom and 19 in the side, provide a natural unit to focus the optimisation on. By finding value for the optimisation defined in value per meter, one can extrapolate the value to represent the whole cross section.

The structural constraints of the problem will be the same either if one wants to optimise with regards for weight or cost, this is because the ship need to fulfil the base requirements set by a classification society, in our case DNV, to be allowed to be built and sailed. The structural constraints for the cargo area are found in the DNV rules [DNV, 2013]. Instead of making the user supply all the parameters needed to calculate the constraint, simplifications are made so not all the functions in the rules must be used.

In terms of the object function, variables defining, and parameters weighting the cost and weight contribution for stiffeners, plates and frames are needed. As both the weight and the cost of the cross section depend on area of plates and stiffeners and also the number of stiffeners, we deduce that the design variables should be the thickness of the plates and the distance between the stiffeners of each panel. The stiffener type is set to be dependent of the stiffener distance, and therefore not a design variable, this is done to reduce the number of variables used in the optimisation.

To account for the fact that the yard will not make special stiffeners, the available and reasonable stiffener dimensions should be taken into account. The stiffeners which are set to be available in the program are found in Table (2.1), normally to be set by user, are chosen by a function in the program after finding the requirement for the stiffener. In the optimisation process, the user will be able to specify which values the cross-sections scantlings can to be chosen from. This is done by defining the feasible set of values for each variable used in the optimisation, as seen in Chapter (7).

The variables for the optimisation can either be continuous, chosen to be a random number within a set region, or the variables can be discrete, then the value of the variable must be taken from a list of allowed values. The last type of variables are the most interesting for the task because this allows the designers to choose the values which are available and favourable for the building process.

To sum up what is needed for the cross section optimisation:

- Object functions: weight and cost.
- Weighing factors for the object function: Cost parameters for steel, labour and consumables. Geometry of the cross section.

- Constraints: requirements connected to the structural integrity, the DNV rules.
- Variables: plate thickness and distance between stiffeners for every panel.

Panel nr.	Type	Section modulus Z $[cm^3]$	Area of stiffener A $[cm^2]$
1	HP 100x6	38	7.74
2	HP 100x8	45	9.74
3	HP 120x6	54	9.31
4	HP 120x8	63	11.7
5	HP 140x8	87	13.8
6	HP 140x9	93	15.2
7	HP 160x8	118	16.2
8	HP 160x9	126	17.8
9	HP 180x9	166	20.7
10	HP 200x9	225	23.6
11	HP 220x10	302	29
12	HP 240x10	368	32.4

Table 2.1: HP profiles allowed in the optimisation program, obtained from Appendix A

Chapter 3

The constraints of ship design

In this chapter the constraints of the problem, namely the DNV rules for ships over a 100 meters are presented. The choice of rules from DNV was made on the basis that Rolls-Royce Marine uses DNV for classification of their design. When referring to the DNV rules later in the text, it will always be the rules over 100 meter unless stated otherwise. [DNV, 2013]

In a design process, the engineers will base their design on the rules stated by the classification society. These rules are often based on analytical structural theory and empirical data that the classification company has gathered over the years. To approve the design, the classification company uses finite element analysis on the structure the designers have chosen. This is an important part of the validation process, because even though the independent segment in the structure are checked against the rules, there might be three dimensional aspects that might have been overlooked. Figure (3.1) shows some of the dependence of the constraints. The

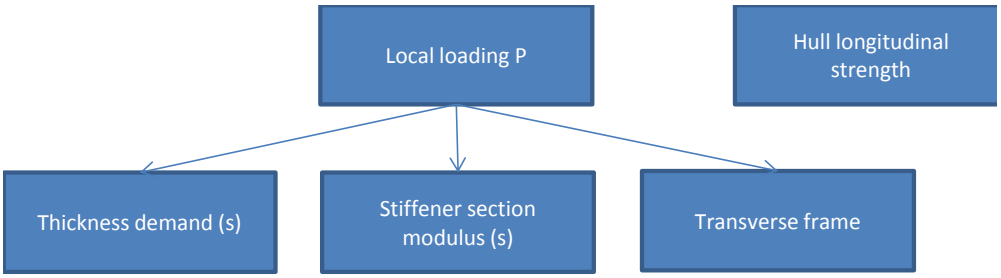


Figure 3.1: Display of constraint dependence

local loading is made from information from the cross section geometry. The local loading in turn shaping the thickness demand for the plates and the section modulus

demand of the stiffener, which both depend on the distance between the stiffeners, s. In addition the local pressure shape the transverse frame demand. The hull longitudinal strength is only dependent on the ship overall parameters.

3.1 Local loading on panels

The DNV rules regarding the pressure on each panel is based on extreme condition values, experience and structural analysis. The panels can be influenced by both internal and external pressures, including static and dynamic- sea pressure, pressure from liquids in tanks and dry cargo. Also the position of the panels affect what type of loading the panels will be exposed to, through the influence of gravity and the acceleration forces. [Mürer, 1996]

In the following text, the different pressure options are presented, but not described in detail. The reader is for this referred to the DNV rules [DNV, 2013]. Because of the wish to keep the choice of pressure conservative, the point where the load is calculated, the load point, is set to be at the lowest position of the element in question. The different load points defined in the DNV rules are based on the behaviour of the components in question. [DNV, 2013]

Bottom Structure

Bottom structure of the ship includes the keel, outer and inner bottom, and bottom floors. The keel runs along the centreline and have a higher minimum thickness demand then the rest of the bottom plating. Both outer bottom and the keel are categorized under p_1 in Figure (3.2) which represents dynamic sea pressure.

The inner bottom plating have more pressure options. Depending on the tank contents, p_4 to p_{15} are the relevant pressures. If the panel is specified to be in the cargo area, p_4 is used. p_4 is dependent on the cargo density and stowing height. The a_v parameter represent vertical acceleration, which is constant in the midship area. If the panel is not loaded by dry cargo, the highest of pressure of p_5,p_7 or p_8 is chosen because of the assumption that the side tanks most often are ballast tanks. p_6 is not considered to limit the user input requirement in the early design phase.Pressures representing liquid cargo is also neglected.

For longitudinal girders the largest of p_{13} or p_{14} is chosen, representing pressure on tank boundaries. p_{15} is not considered because it is dependent on knowledge about the damaged condition. Again this is to reduce the user input needed in the early design optimisation.

Table B1 Design loads		
Structure	*Load type*	*p (kN/m²)*
Outer bottom	Sea pressure	$p_1 = 10\,h_0 + p_{dp}$ (kN/m²) 1)
	Net pressure in way of cargo tank or deep tank	$p_2 = \rho\,(g_0 + 0.5\,a_v)\,h_s - 10\,T_M$ $p_3 = \rho\,g_0\,h_s + p_0 - 10\,T_M$
Inner bottom	Dry cargo in cargo holds	$p_4 = \rho\,(g_0 + 0.5\,a_v)\,H_C$
	Ballast in cargo holds	$p_5 = (10 + 0.5a_v)h_s$ $p_6 = 6.7(h_s + \phi b) - 1.2\sqrt{H\,\phi\,b_t}$ 2) $p_7 = 0.67(10h_p + \Delta p_{dyn})$ $p_8 = 10h_s + p_0$
	Liquid cargo in tank above	$p_9 = \rho\,(g_0 + 0.5a_v)\,h_s$ $p_{10} = \rho g_0[0.67(h_s + \phi b) - 0.12\sqrt{H\,\phi b_t}]$ 2) $p_{11} = 0.67(10h_p + \Delta p_{dyn})$ $p_{12} = \rho g_0 h_s + p_0$
Inner bottom, floors and girders	Pressure on tank boundaries in double bottom	$p_{13} = 0.67\,(10\,h_p + \Delta\,p_{dyn})$ $p_{14} = \rho\,g_0 h_s + p_0$
	Flooded condition	$p_{15} = 10h_b$
1) For ships with service restrictions the last term in p_1 may be reduced by the percentages given in Sec.4 B202. 2) p_6 and p_{10} to be used in tanks/holds with largest breadth > 0.4 B.		

Figure 3.2: Pressure on bottom structure, yellow marking the considered pressures.[DNV, 2013, page 89]

Side Structure

In the side structure definition there is a choice of inner and external structure.

p_1 and p_2 from Figure (3.3) is used for the external panels. Both pressures are affected of possible service restrictions of the ship.

On the internal sides, pressures p_3, p_4 or p_5 are relevant. All of them are dependent on a parameter h_b which represent the distance to the ballast waterline. With guidance from Rolls Royce, this parameter was set equal to the ship draught.

Table B1 Design loads		
Load type		*P (kN/m²)*
External	Sea pressure below summer load waterline	$p_1 = 10\ h_0 + p_{dp}$ 1)
	Sea pressure above summer load waterline	$p_2 = (p_{dp} - (4 + 0.2\ k_s)\ h_0)$ 1) minimum $6.25 + 0.025\ L_1$
Internal	Ballast, bunker or liquid cargo in side tanks in general	$p_3 = \rho\ (g_0 + 0.5\ a_v)\ h_s - 10\ h_b$ $p_4 = \rho\ g_0\ h_s - 10\ h_b + p_o$ $p_5 = 0.67\ (\rho\ g_0\ h_p + \Delta\ p_{dyn}) - 10\ h_b$
	Above the ballast waterline at ballast, bunker or liquid cargo tanks with a breadth > 0.4 B	$p_6 = \rho g_0[0.67(h_s + \phi b) - 0.12\sqrt{H\ \phi b_t}]$
	Above the ballast waterline and towards ends of tanks for ballast, bunker or liquid cargo with length > 0.15 L	$p_7 = \rho g_0[0.67(h_s + \theta l) - 0.12\sqrt{H\ \theta\ l_t}]$
	In tanks with no restriction on their filling height 2)	$p_8 = \rho\left[3 - \frac{B}{100}\right]b_b$
1) For ships with service restrictions, p_2 and the last term in p_1 may be reduced by the percentages given in Sec.4 B202.		
2) For tanks with free breadth b_s > 0.56 B the design pressure will be specially considered according to Sec.4 C305.		

Figure 3.3: Pressure on side structure, yellow marking the considered pressures..[DNV, 2013, page 103]

Deck Structures

In the uppermost continuous deck, the strength deck, the pressure is set to p_1 in Figure (3.4), accounting for load of sea water on deck. Other deck structures is defined by p_6, p_7 or p_8, in the same figure. The first of these is valid if the deck can be counted as a tank bottom, the two others are valid both as tank top and bottoms.

Table B1 Design loads		
Structure	*Load type*	*p (kN/m²)*
Weather decks [1]	Sea pressure	$p_1 = a(p_{dp} - (4 + 0,2k_s)h_0)$ [2], minimum 5.0
	Deck cargo	$p_2 = (g_0 + 0.5\ a_v)\ q$
Cargo 'tweendecks	Deck cargo	$p_3 = \rho_c\ (g_0 + 0.5\ a_v)\ H_C$
Platform deck in machinery spaces	Machinery and equipment	$p_4 = 1.6\ (g_0 + 0.5\ a_v)$
Accommodation decks	Accommodation in general	$p_5 = 0.35\ (g_0 + 0.5\ a_v)$, see also Sec.4 C401
Deck as tank bottom in general	Ballast, bunker or liquid cargo	$p_6 = \rho\ (g_0 + 0.5\ a_v)\ h_s$ $p_7 = 0.67\ (\rho\ g_0\ h_p + \Delta\ p_{dyn})$ $p_8 = \rho\ g_0\ h_s + p_0$
Deck as tank top in general		$p_7 = 0.67\ (\rho\ g_0\ h_p + \Delta\ p_{dyn})$ $p_8 = \rho\ g_0\ h_s + p_0$
Deck as tank boundary in tanks with breadth > 0.4 B		$p_9 = \rho g_0[0.67(h_s + \phi b) - 0.12\sqrt{H\phi b_t}]$
Deck as tank boundary towards ends of tanks with length > 0.15 L		$p_{10} = \rho g_0[0.67(h_s + \theta l) - 0.12\sqrt{H\theta l_t}]$
Deck as tank boundary in tanks with breadth > 0.4 B [3]		$p_{11} = \rho\left(3 - \frac{B}{100}\right) b_b$
Deck as tank boundary in tanks with length > 0.1 L [4]		$p_{12} = \rho\left(4 - \frac{L}{200}\right) l_b$
Watertight decks submerged in damaged condition [5]	Sea pressure	$p_{13} = 10\ h_b$

1) On weather decks combination of the design pressures p_1 and p_2 may be required for deck cargo with design stowage height less than 2.3 m.

2) For ships with service restrictions p_1 may be reduced with the percentages given in Sec.4 B202. C_W should not be reduced

3) To be used for strength members located less than 0.25 b_b away from tank sides in tanks with no restrictions on their filling height. For tanks with free breadth (no longitudinal wash bulkheads) $b_b > 0.56$ B the design pressure will be specially considered according to Sec.4 C305

4) To be used for strength members located less than 0.25 l_b away from tank ends in tanks with no restrictions on their filling height. For tanks with free length (no transverse wash bulkheads or transverse web frames in narrow tanks) $l_b > 0.13$ L the design pressure will be specially considered according to Sec.4 C305

5) The strength may be calculated with allowable stresses for plating, stiffeners and girders increased by 60 f_1.

Figure 3.4: Pressure on deck structure, yellow marking the considered pressures..[DNV, 2013, page 120]

Bulkhead structure

In a cargo ship, the inner sides may be defined as longitudinal bulkheads if they run through the whole cargo area. The relevant pressures found in Figure (3.5). The pressures chosen there, $p3$, $p4$ and $p5$ are not so different from the pressures chosen for the inner side structure, only missing the ballast waterline reduction, they are identical to the pressures for decks as tank components.

Table B1 Design loads			
Structure		*Load type*	*p (kN/m²)*
Watertight bulkheads		Sea pressure when flooded or general dry cargo minimum	$p_1 = 10\ h_b$
Cargo hold bulkheads		Dry bulk cargo	$p_2 = \rho_c\ (g_0 + 0.5\ a_v)\ K\ h_c$
Tank bulkheads in general		Ballast, bunker or liquid cargo	$p_3 = \rho\ (g_0 + 0.5\ a_v)\ h_s$ $p_4 = 0.67\ (\rho\ g_0\ h_p + \Delta\ p_{dyn})$ $p_5 = \rho\ g_0\ h_s + p_0$
Longitudinal bulkheads as well as transverse bulkheads at sides in wide tanks	In tanks with breadth > 0.4 B		$p_6 = \rho g_0 [0,67(h_s + \phi b) - 0,12\sqrt{H\ \phi\ b_t}]$
	Note 1)		$p_7 = \rho\left[3 - \frac{B}{100}\right] b_b$
Transverse bulkheads and longitudinal bulkheads at ends in long tanks	In tanks with length > 0.15 L		$p_8 = \rho g_0 [0,67(h_s + \theta l) - 0,12\sqrt{H\ \theta\ l_t}]$
	Note 2)		$p_9 = \rho\left[4 - \frac{L}{200}\right] l_b$
Longitudinal wash bulkheads			$p_7 = \rho\left[3 - \frac{B}{100}\right] b_b$
Transverse wash bulkheads			$p_9 = \rho\left[4 - \frac{L}{200}\right] l_b$

1) To be used for strength members located less than 0.25 b_b away from tank sides in tanks with no restrictions on their filling height. For tanks with free breadth (no longitudinal wash bulkheads) $b_b > 0.56$ B the design pressure will be specially considered according to Sec.4 C305.

2) To be used for strength members located less than 0.25 l_b away from tank ends in tanks with no restrictions on their filling height. For tanks with free length (no transverse wash bulkheads or transverse web frames in narrow tanks) $l_b > 0.13$ L the design pressure will be specially considered according to Sec.4 C305.

Figure 3.5: Pressure on bulkhead structure, yellow marking the considered pressures..[DNV, 2013, page 128]

3.2 Plate thickness demand

The DNV rule for plate thickness,t, is based on simple plate strip theory. To get the right unit on the thickness in Equation (3.1), taken from the DNV rules, the

parameters inserted must be correct.

$$t_{d1} = \frac{15.8 \cdot s \cdot \sqrt{p}}{\sqrt{\sigma}} + t_c \qquad [mm] \qquad (3.1)$$

t_{d1} = Minimum plate thickness demand, dependent on s

s = Distance between stiffeners $[m]$

p = Pressure on the plate $[\frac{kN}{m^2}]$

σ = Allowed bending stress $[\frac{N}{mm^2}]$

t_c = Corrosion addition $[mm]$

There is a second formula for restraining the minimum thickness, shown in equation (3.2). This formula is based on DNV's experience with different ships. In both the thickness requirements, the contribution from a corrosion coefficient, t_c, is dependent on the contents of the rooms the panel is facing. This value must therefore be given as user input for each panel. In addition, t_c is only added in the weight and cost calculations, and will not contribute to the strength calculations. [DNV, 2013, page 23]

In Figure (3.6), the thickness minimum demand depending on the t_{d1}, t_{d2}, distance between stiffeners, s, and length of the stiffener, l_s, is shown. The figure indicates that a high stiffener distance will give a high thickness demand and that over a certain stiffener spacing the frame distance also adds to the thickness demand.

$$t_{d2} = t_l + \frac{k_t \cdot L_1}{\sqrt{\sigma}} + t_c \qquad [mm] \qquad (3.2)$$

t_l = Thickness depending on panel location $[mm]$

k_t = Factor depending on panel location $[\ ]$

L_1 = Length of ship, max 300 $[m]$

t_c = Corrosion addition $[mm]$

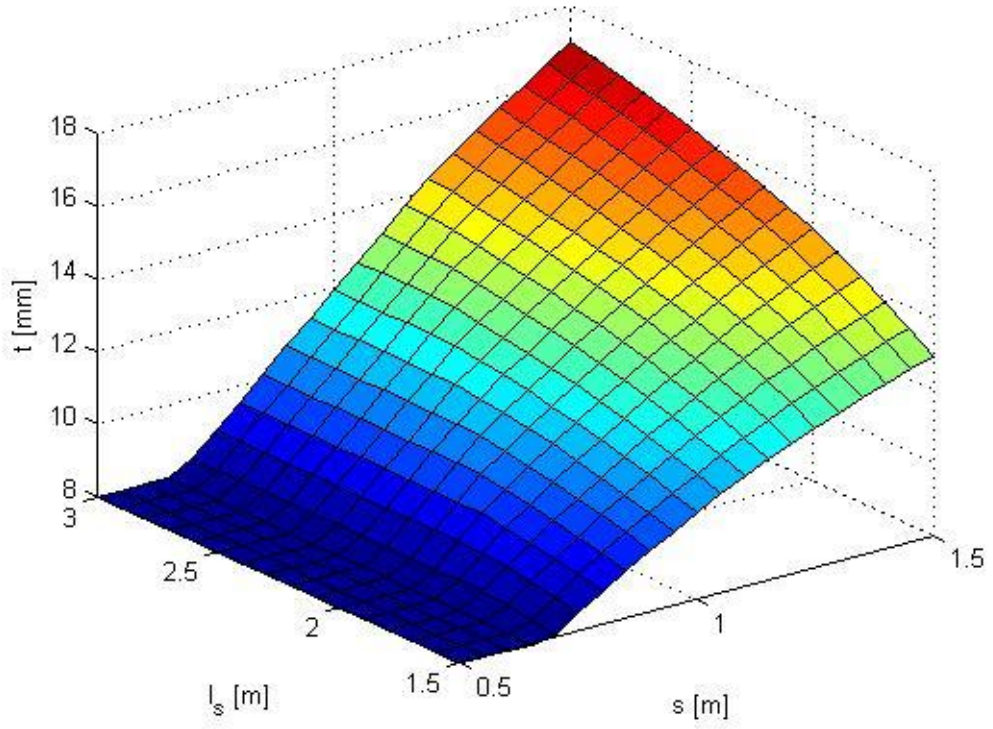


Figure 3.6: Max thickness demand from the DNV formulas. Dependent on stiffener length and spacing

3.3 Stiffener options

In this thesis one assume that all the stiffeners are longitudinal directed, this is convenient as the regulation for these stiffeners are simpler and more straightforward to calculate then the transverse stiffeners. The DNV rules have restrictions for the minimum stiffener strength, found in Equation (3.3). The section modulus of the stiffener is chosen after how much load it must withstand. The stiffeners are usually placed with equal spacing on a plate, but with some deviations near the ends to get the distance to fit.

$$Z_{lok} = \frac{83 \cdot l^2 \cdot s \cdot p}{\sigma} \quad [cm^3] \tag{3.3}$$

$\sigma =$ Allowable bending stress $[\frac{N}{mm^2}]$

$p =$ Dimensioning pressure $[\frac{kN}{m^2}]$

$s =$ Distance between the stiffeners $[m]$

$l_s =$ Length of the stiffener $[m]$

3.4 Transverse frame

Following the distribution of the load on the ship, the longitudinal stiffeners will distribute their loading into the transverse frames in the ship. The transverse frames consists of a transverse girder in the bottom, in the form of the floors in the double bottom, and girders in the side, in form of plates between external and inner side, as kan be seen in Figure (2.3). In the side structure the section modulus demand of the girder is given by the DNV rules in Equation (3.4), but the strength demand for the bottom structure must be found by performing structural analysis. [DNV, 2013] [Amdahl, 2009]

When one analyses the cross section structure with a simple beam analysis, one can, because of symmetry assume the bottom structure to be rigidly fastened in the centre line. The ship corner can not be counted as a rigid corner because of the deflection the ship side may allow for. As the corner will have a rotation, a simply supported end is not a perfect representation of the structural behaviour, but this assumption is used in the simplification of the transverse girder. By representing the longitudinal girder as springs, the maximum moment expected on the bottom beam is expected to be as given on a general form in Equation (3.9) and shown Figure (3.7). [Amdahl, 2009] The Longitudinal girders will affect the stresses in the beam, therefore one should, as a part of the total design process preform a more in-depth direct strength calculation. Information for this procedure is discussed in section 12 of [DNV, 2013], but only a rough simplification will be included in this optimisation program.

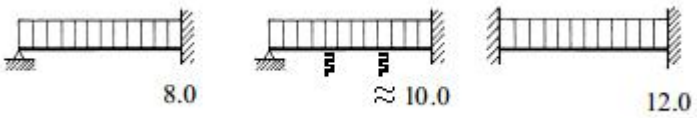


Figure 3.7: The different clamping alternatives of beams with the connected moment coefficient, m

$$Z_{DNV_s} = \frac{100 \cdot S^2 \cdot b \cdot p}{\sigma} \tag{3.4}$$

$S =$ Length of beam
$p =$ Pressure
$b =$ Load area breadth
$\sigma =$ Stress level in side

$$M = \frac{q}{m} \tag{3.5}$$

$$Q = p \cdot b \tag{3.6}$$

$$M_{fixed} = \frac{Q \cdot L^2}{12} \tag{3.7}$$

$$M_{simple} = \frac{Q \cdot L^2}{8} \tag{3.8}$$

$$M_{bottom} = \frac{Q \cdot L^2}{10} \tag{3.9}$$

$p =$ Pressure
$b =$ Load area breadth
$Q =$ Force per meter
$L =$ Length of beam

$$Z_{bottom} = \frac{M_{bottom}}{\sigma} \tag{3.10}$$

$\sigma =$ Stress level in bottom

When calculating the actual section modulus of the bottom or side, transverse beam, one should use a effective breadth of the flange to account for the shear deformation on the flanges, as illustrated in Figure (3.8). To obtain this value, the width of the plate may according to the DNV rules be reduced with regards to an aspect ratio of the loaded area. Based on the table in [DNV, 2013, page 40]. Figure (3.9) was made of the data for the highest numbers of point load, to simulate continuous loading, and the effective breadth coefficient, C wasw made as a polynomial function of the plates aspect ratio, shown in Equation (3.14). In the simplification for calculating the section modulus of the transverse beams, the web thickness of the transverse girder equals the requirement for the floor thickness,which can be found in the DNV rules in a simple demand in Equation (3.12). The web thickness in the side can also be set as a constant defined by the DNV rules, the formula found in Equation (3.11). Another simplification made, is that the section modulus for the beam in bottom and side are dependent on the smallest plate flanges in the span of the beams. The girder's flanges are the bottom and inner bottom plates or the external and internal side plates. To ensure that the structure is strong enough, the section modulus of the beam cross section, calculated with effective flange, must be bigger then the one calculated in Equation (3.10).

$$t_{w_s} = 5 + \frac{0.02 \cdot L1}{\sqrt{f_1}} \tag{3.11}$$

$$t_{w_b} = 6 + \frac{0.02 \cdot L1}{\sqrt{f_1}} \tag{3.12}$$

t_{w_s} = Thickness of web in side

t_{w_b} = Thickness of web in bottom

$L1$ = Length of ship, not over 300

f_1 = Material factor

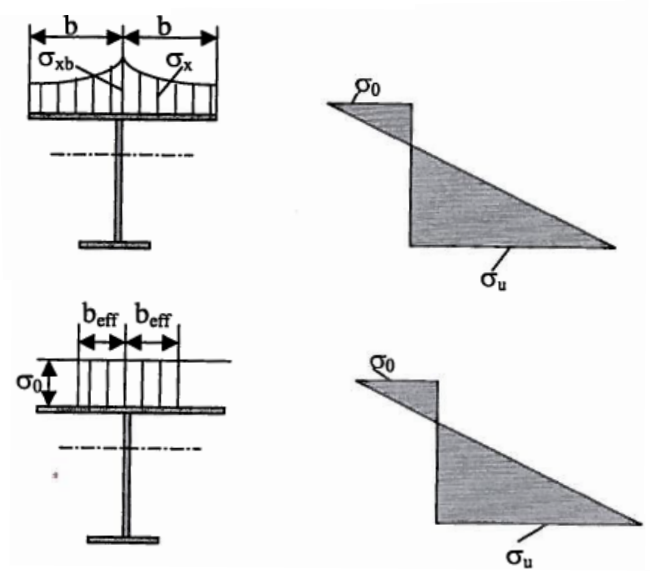


Figure 3.8: The effective breadth of flange, compensating for the variation of stress over the flange.

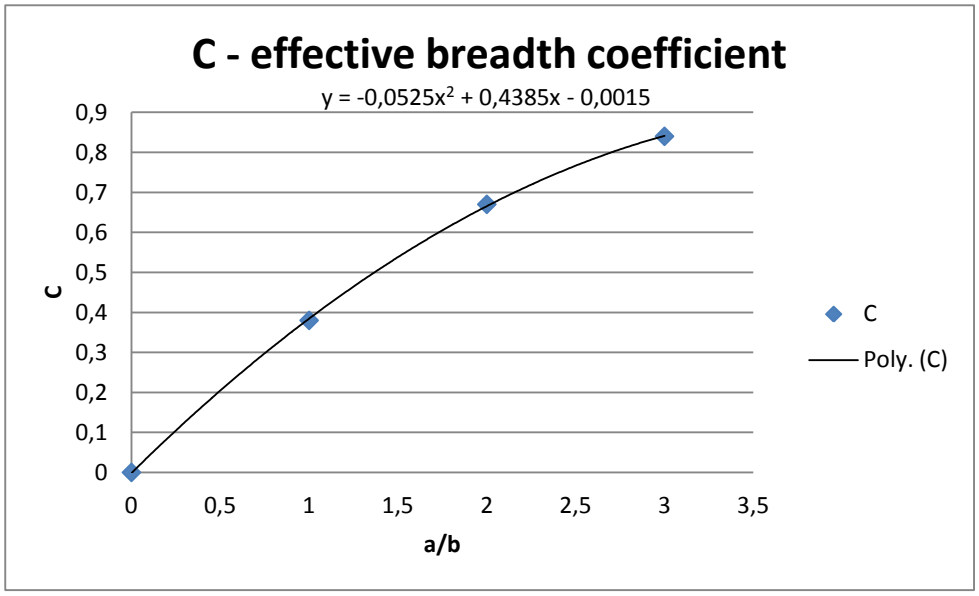


Figure 3.9: Effective width coefficient,the y-axis, based on aspect ratio of stiffener loading, the x-axis.

$$\frac{a}{b_f} = \frac{0.6 \cdot S}{b_f} \tag{3.13}$$

$$C = -0.0525 \cdot (\frac{a}{b_f})^2 + 0.4385 \cdot \frac{a}{b_f} - 0.0015 \tag{3.14}$$

$$b_e = C \cdot b_f \tag{3.15}$$

a = Distance between points of zero bending moments. Assumed fixed ends.

b_f = Flange breadth for the stiffener or girder.

S = Length of stiffener or girder in meters

C = Width coefficient. Based on number of load points over 6.

b_e = Effective flange breadth in meters

3.5 Hull longitudinal strength

In the section modulus for the whole midship cross section, Z_{cs}, the longitudinal stiffeners will contribute in addition to the decks and the side structure.

Globally the ship must endure stresses from moments caused by waves and still water loads which are given by a exceedance rate of 10^{-8}. This means it should be able to endure the loading by waves that statistically only occurs with 10^{-8} annual probability.[Mürer, 1996]

The required longitudinal bending moment midship is found in section 5 in the DNV rules [DNV, 2013]. Equation (3.16) and (3.17) give minimum stillwater moment (M_s) the ship should withstand. The equations are empirically developed and vary with the ships' length and fullness. [Amdahl, 2009]

$$Ms_{sag} = -0.065 \cdot C_w \cdot L^2 \cdot B \cdot (C_B + 0.7) \quad [kNm] \tag{3.16}$$

$$Ms_{hog} = C_{wu} \cdot L^2 \cdot B \cdot (0.01225 - 0.015 \cdot C_B \quad [kNm] \tag{3.17}$$

$$Mw_{sag} = -0.11 \cdot C_w \cdot L^2 \cdot B \cdot (C_B + 0.7) \quad [kNm] \tag{3.18}$$

$$Mw_{hog} = 0.19 \cdot C_w \cdot L^2 \cdot B \cdot C_B \quad [kNm] \tag{3.19}$$

C_{wu} = Wave coefficient accounting for restriction

C_w = Wave coefficient not accounting for restriction

L = Ship length in meters

B = Ship breadth in meters

C_B = Block coefficient

Equation (3.18) and (3.19) represent moments caused by wave effect, M_w, in normal speed. The formula is based on weather data form the North Atlantic.

The wave coefficient depend on the length of the ship, and is given by the calculations in Equations (3.20) to (3.22).

$$C_w = 10.75 - (\frac{300 - L}{100})^{\frac{3}{2}} \qquad \text{for L} < 300 \tag{3.20}$$

$$C_w = 10.75 \qquad \text{for } 300 < \text{L} < 350 \tag{3.21}$$

$$C_w = 10.75 - (\frac{L - 350}{150})^{\frac{3}{2}} \qquad \text{for L} > 355 \tag{3.22}$$

When looking at the requirement for the section modulus of the cross section, the rules mainly focuse on the midship area where the scantlings are held constant.

The method for calculating the cross section section modulus uses a the direct mathematical approach, based on cargo and ballast conditions. Using a combination of the moment contribution form stillwater M_s and wave bending moments M_w in either hog or sag one obtains the formula in Equation (3.23). Here M_s may be enlarged with 20-40% to be conservative[1]. This is the formula used in the optimisation.

$$Z_{cs} = \frac{M_s + M_w}{\sigma_l} \cdot 10^3 \qquad [cm^3] \tag{3.23}$$

$$\sigma_l = 175 \cdot f_1 \text{ Within 0.4 L amidship}$$

$$f_1 = \text{Material factor}$$

When calculating the ships own cross section modulus, Z_{built}, all the longitudinal elements will contribute. When adding the stiffeners in this calculation, one do not want to add each stiffener one by one, as it require a lot of work. For this reason one may include the stiffeners contribution by adding the area of the stiffeners to the panel as additional panel thickness. In this way, the stiffeners are merged with the panels contribution as a smoothly distributed additional thickness. This practice is called smearing, seen illustrated in Figure 3.10. [Leira et al., 2011]

3.6 Normalised constraints

In the optimisation it is necessary to look at the constraints, g, in normalised terms as seen in Equation (3.24). This is done to adapt the constraints to the

[1]Recommended by Evelin Tankovic

Figure 3.10: The cross section area of the stiffener smeared over the plate to form a equally distributed additional thickness.

optimisation algorithm. For the solution to be valid, g must be *less* then zero. The parameter b_a will represent the different values calculated as a function of, or set to, the design variables distance between stiffeners, s, and plate thickness, t. a_s is here the DNV and structural demands found in the equations presented earlier in the text. This is exemplified in Equation (3.25), which is used to test the thickness for every panel, while Equation (3.26) test the strength of the whole cross section.

$$g = \frac{a_s - b_a}{a_s + b_a} \tag{3.24}$$

$g =$ Normalised constraint value

$a_s =$ Strength demand

$b_a =$ Actual loading

$$\frac{t_{min} - t}{t_{min} + t} < 0 \tag{3.25}$$

$$\frac{Z_{cs} - Z_{built}}{Z_{cs} + Z_{built}} < 0 \tag{3.26}$$

3.7 Global loading conditions

In some simple structures, the local demand of the plates and stiffeners is enough to fulfil the global demand as well, but when the structure becomes more complex, frame and beam analyses using finite element method are necessary to ensure structural integrity. This is especially important when the ship does not have homogeneous loading, as some dry cargo ship do. When this is the case, direct

strength analysis of the cargo area is needed. In these analysis one must test the ship for different loading conditions to see which is the most critical. The loading conditions include fully loaded, partially loaded, ballasted and load during on and off loading of the ship. The beams in the bottom structure should be tested for stresses and deformations. The demands presented above is beyond the scope of this thesis.

3.8 Limitations of the program

As this thesis goal is to make a program applicable for general cargo multi-purpose ships that can carry dry bulk, containers or other cargo,the special calculations required for special classes of ships are not considered. The user supplies the geometry input of the cross section, the corrosion addition to the plates caused by the content of tanks. As a structural limitation, no plates can be represented or calculated as curved. The cost and weight calculations will not include the transverse bulkheads contributions. Also there will not be additions and reductions considering the brackets and cut outs which are normal in he ship hull.

Because of the need for making the programme simple and manageable, it was necessary to make simplifications. All the panels will therefore only have longitudinal stiffeners due to of the extent of the rules involved. It is also assumed that there is no stiffeners on the floors in the double bottom , but all the other panels will have stiffeners. The dimensioning pressure was chosen to be as conservative as possible. In addition, only a few of the local loading conditions has been considered, avoiding the formulas that require parameter not easily obtained in the early design phase.

Because of the choice of rule based design, no advanced analysis of beam system will be implemented. Also the buckling or shear strength requirements where not added in the code.

The DNV rules for ships over 100 meter are more complex than the rules for ships under 100 meters. The latter have simplifications regarding the scantlings and the need for direct calculations, because a shorter ship will experience the same magnitude in loading then a long ship. Using the program made in this thesis for a ship under 100 meters is possible, but one may end up with number which are higher then what one actually need.

Chapter 4

The objective function for ship cross section optimisation

Then optimising for weight, the assumption is that one will get the lowest weight when the structure has thin plates and close stiffeners. In contrast, the structure that costs the least will have thicker plates and fewer stiffeners, resulting in a heavier construction. In defining the object functions the main challenge is getting the influence of each component correctly described in the search of finding the optimum structure.

In this chapter the parameters affecting the design variables, t and s, in the object function will be presented.

4.1 Weight function

The stiffener are important both in both the weight and cost calculations. The stiffeners and plates all contribute to fulfilling the cross section section modulus demand, Z_{built}, constrained the DNV formula in Equation (3.23), which depend on the design variable s. In the weight calculation the stiffeners area is needed. To find this, an empirical formula dependent on the section modulus of the stiffener was made and can be found in Equation (4.5). By using the standard information available for the stiffeners, provided by Rolls-Royce, found in Appendix A, and the formulas in Equation (4.1) to (4.5), the area of the stiffener without the flange was found as a function of Z_{loc}. The relation seen in Figure (4.1) is made by fitting the points on the graph to a polynomial trend line. The excel spread sheet used is found in Appendix B. Equation (4.2) was found in chapter 3 of the buckling

compendium of Jørgen Amdahl. In the optimisation, the chosen stiffener will be the one which is closest to the required area of the stiffener. [Amdahl, 2005]

$$z_n = \frac{\sum d_i \cdot A_i}{\sum A_i} \tag{4.1}$$

z_n = Distance from lowest point to the neutral axis
d_i = Distance from bottom to local center of mass
A_i = Area of part

$$I_e = I_s + e_x^2 \cdot A_s \cdot (1 + \frac{A_s}{b_e \cdot t})^{-1} \tag{4.2}$$

I_e = Moment of inertia of effective flange
I_s = Moment of inertia of the HP-stiffener
e_x = Distance from bottom to the center of mass of stiffener
A_s = Cross section area of HP-stiffener
b_e = Effective width of the plate flange
t = Thickness of plate flange

$$Z_e = \frac{I_e}{max(z_n, H_{hp} - z_n)} \tag{4.3}$$

Z_e = Section modulus for stiffener and flange
H_{hp} = Height of stiffener

(4.4)

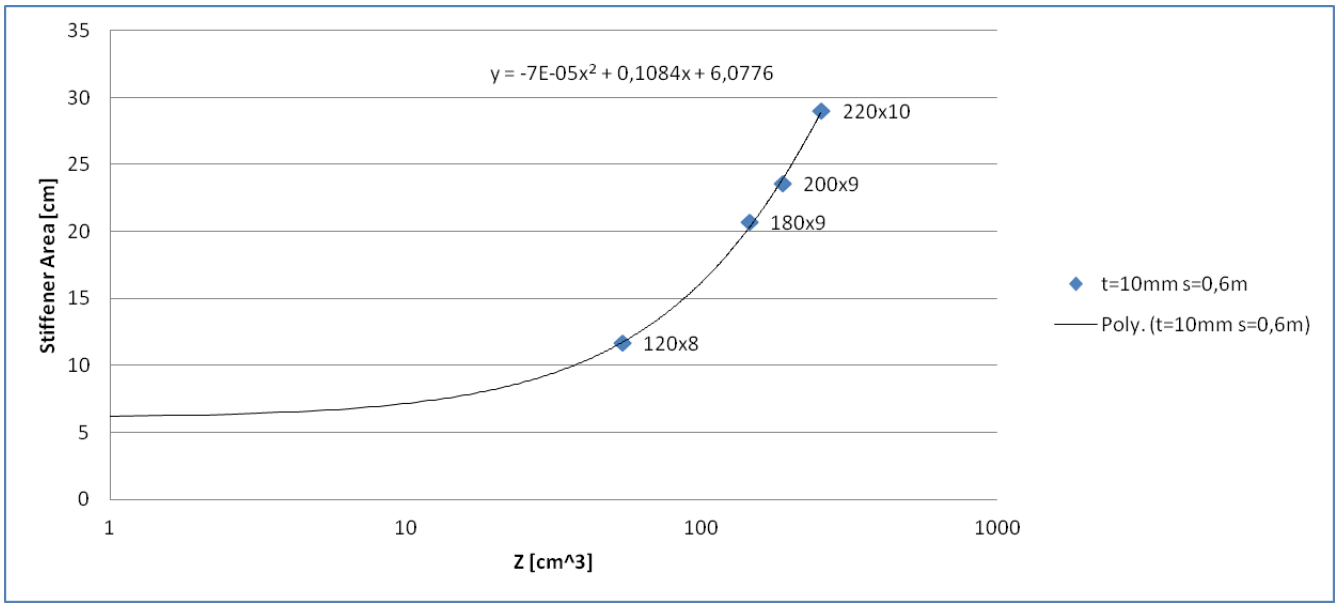


Figure 4.1: The stiffener area as a function of section modulus demand on stiffener with flange.

$$A_s = -7E{-}05 \cdot x^2 + 0.1084 \cdot x + 6.0776 \tag{4.5}$$
$$x = Z_{loc} \text{ demand}$$

The thickness and stiffener distance will differ from plate to plate. The numbers chosen in Appendix B, $t = 6mm$ and $b = 0.5m$, were chosen to be representative values to base the formula in Equation (4.2) on, so that it could be used to find the necessary area of all stiffeners in the ship. For the same reason, the effective width was set to 75% of the width in Equation (4.2).

In a simple weight optimisation of a panel, the stiffener distance is significant in both the formula for minimum thickness, in Equation (3.1), and in defining the section modulus demand for the stiffener, as seen in Equation (3.3). In Figure (4.2) and (4.3), the two minimum stiffness demands, t_{d1} and t_{d2}, are plotted together with the contribution from the stiffener area from Equation (4.5). The figures shows how the thickness development on a panel, subjected to different pressures, is dependent on the stiffener spacing. Considering the figures, the assumptions that the smallest stiffener distance will give the lowest weight is not completely true when constraints are considered. The thickness and stiffener spacing will be in a weight optimisation be chosen as the combination which gives the least effective thickness. The chosen allowed stiffeners will affect the choices.

In Figure (4.2) and Figure (4.3) the thickness restrictions for the same panel under different pressures are plotted. In addition the contribution from the stiffener smeared, t_a is plotted. If other variables in Equation (3.1) and (3.2) where changed, this would affect where the two minimum demands would intersect and thereby change the minimum weight point.

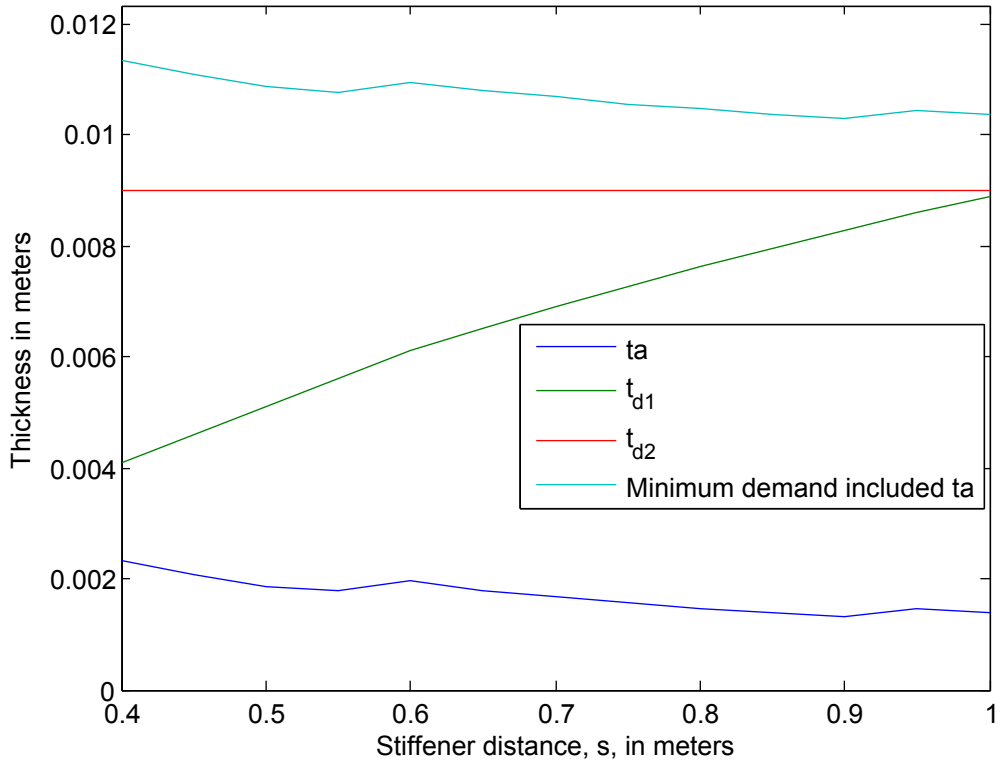


Figure 4.2: Graphs showing the effective thickness of a panel, thickness demand in addition to smeared thickness t_a , depending on the distance between stiffeners. Pressure set to 50 kNm.

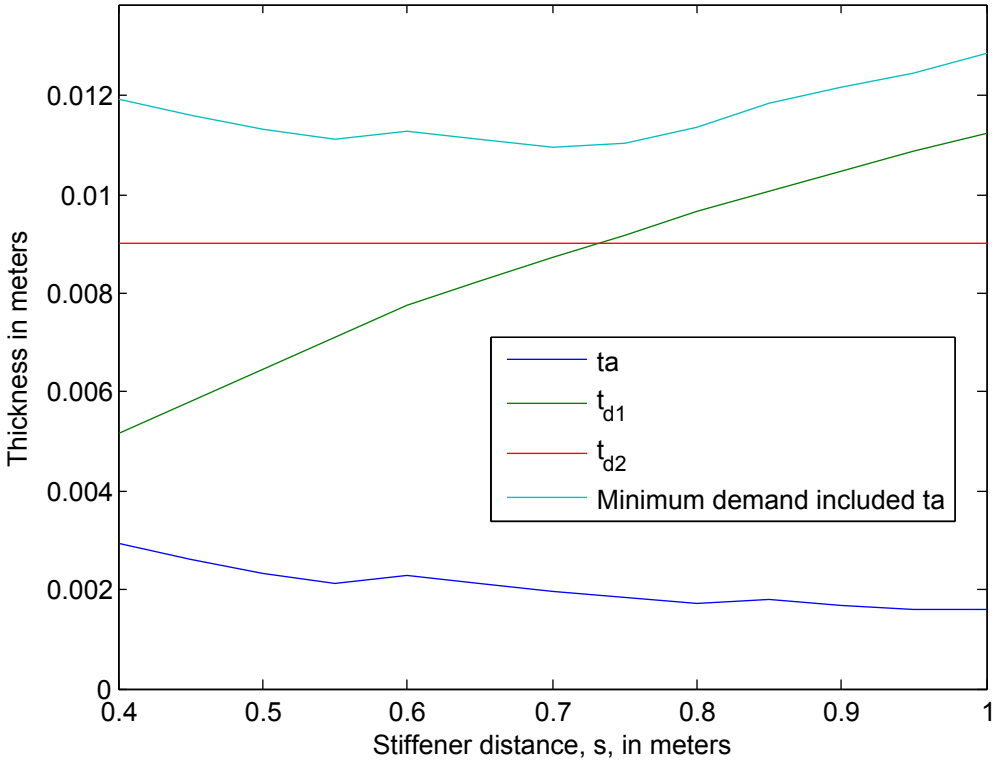


Figure 4.3: Graphs showing the effective thickness of a panel, thickness demand in addition to smeared thickness t_a , depending on the distance between stiffeners. Pressure set to 80 kNm.

The total weight will be calculated with contributions from plates, stiffeners and the frame, as seen in Equation (4.9).

$$W_p(i) = rho_s \cdot L(i) \cdot (t(i) + t_c(i)) \quad \text{kg/m} \quad (4.6)$$

$$W_s(i) = rho_s \cdot L(i) \cdot \frac{A_s(i)}{s(i)} \quad \text{kg/m} \quad (4.7)$$

$$W_F(j) = \frac{rho_s \cdot S(j) \cdot t_w(j) \cdot w_h(j)}{l_s} \quad \text{kg/m} \quad (4.8)$$

$$W = \sum_i W_p + W_s + \sum_j W_F \quad \text{kg/m} \quad (4.9)$$

i = Number of panels
j = Beam in side and beam in bottom
W_p = Weight of plate, including corrosion addition
rho_s = Steel density in kg per m^3
L = Panel length in meters
t = Panel thickness in meters
t_c = Corrosion addition in meters
W_s = Weight of stiffener
A_s = Area of stiffener in meters squared
s = Distance between stiffeners in meters
W_F = Weight of transverse frame
S = Length of beam in meters
t_w = Thickness web in meters
h_w = Height web in meters
l_s = Length of stiffener in meters

4.2 Cost function

The prices connected to ship building are not easy to get hold of as ship yards wish to keep these numbers to themselves for competitive reasons. Besides the cost of the actual steel, the cost of welding the stiffeners on to the plate must be considered. The labour cost involved will vary depending on where the work is done; an hour of weld work done in Norway will be much more expensive than if the work is done in Turkey or Poland. On top of this, the cost will vary with welding method and complexity of the weld work. [Rigo, 2001]

The first approach on the cost estimation, was made using a standard cost per meter weld, and scaling the cost dependent on the thickness of the weld as done

C_{plate}	0.8	euro/kg
$C_{stiffener}$	1.6	euro/kg
K	80	kg/hour
P_4^0	2.5	hour/m
δP_4	0.02	hour/m
P_{10}^0	0.15	hour/m^2
δP_{10}	0.04	hour/m^2
C_{8x}^0	2	euro/m
δC_{8x}	0.05	Variation of C_{8x} per mm
E_0	0.010	m
E_{0x}	0.010	m

Table 4.1: Cost parameters

in [Thanh, 2010]. This was not successful as the cost for few stiffeners where higher then the cost of having many, contradiction the initial assumption for weight optimisation. The problem may have been the lack of information on what the standard cost included, the insecurity of the number is enough to want another way to calculate the cost.

To get a reasonable formula for the cost, some simplifications where done. The formula made in this thesis will not distinguish between weld methods or the location of the weld in the ship. The formulas were adapted from Rigo's paper [Rigo, 2001], in this the example in Appendix C, provided by Sören Ehlers, was very helpful. In addition a standard number of steel cost of $3£/m$ provided by Rolls Royce[1] was used for the adapting the cost parameters to provide a reasonable total cost.

Testing the cost formulas on a example cross section the cost per kg landed on 2.84 euro per kg. Which was deemed to be sufficiently close to the number provided by Rolls Royce.

In table (4.1), C_{plate} is the cost per kg of steel used for the plates, $C_{stiffener}$ is the cost per kg of steel used for the stiffeners. K is a coefficient connecting work load in hours to material processed. P_4^0 represents the workload for welding longitudinal stiffener. P_{10}^0 is the workload required to prepare 1 m^2 plate. C_{8x}^0 represent cost of consumables per meter and δC_{8x}, $\delta P4$ and $\delta P10$ all add a cost for deviation from the preferred thickness of stiffener, E_0x, or plate, E_0.

As the cost of steel is different for stiffeners and plates, each panel will get two contributions related to the steel cost as shown in Equation (4.10). In the calculation

[1]Evelin Tankovic

of the total cross section in Equation (4.15), the cost of the steel for the transverse frame is added and divided by the frame distance to get the cost per meter.

$$C_{steel} = (A_{plate} \cdot C_{plate} + A_{stiffener} \cdot n_s \cdot C_{stiffener}) \cdot \rho \qquad \text{[euro/m]} \qquad (4.10)$$

C_{steel} = Cost of panel per meter
A = Area in m^2
C = Cost per kg material
n_s = Number of stiffeners fitted to the panel
ρ = Density of the steel in kg per m^3

The K parameter in Table (4.1) is used in the calculation of the labour cost, together with parameters related to the work load for each task. Here, the cost of deviating from preferred stiffener sizes is used. These parameters naturally depend on the yards ability to produce the needed material. Making a thickness dependent price variance should also be possible to do if one had the price of a couple of similar panels with varying thickness. As this information was not possible to obtain from Rolls Royce, one could either make a qualified guess or neglect it's influence in the cost estimation. To try to keep the formulas simple, the choose was made not to add this cost to the formula.

$$C_l = K \cdot C_{plate} \cdot L_i \cdot (\frac{1}{s} \cdot P_4 + P_{10}) \qquad (4.11)$$

$$P_4 = P_4^0 \cdot (1 + (d_x - E_{0x}) \cdot 10^3 \cdot \delta P_4) \qquad (4.12)$$

$$P_{10} = P_{10}^0 \cdot (1 + (\delta - E_0) \cdot 10^3 \cdot \delta P10) \qquad (4.13)$$

C_l = Cost labour, euro per meter
C_{plate} = Cost op plate
L_i = Length of element
P_4^0 = Workload for welding longitudinal stiffener
P_{10}^0 = Workload for welding required to prepare 1 m^2 plate
$\delta P4$ = Factor which punishes deviation from stiffener thickness
$\delta P10$ = Factor which punishes deviation from plate thickness
E_0x = Stiffener thickness standard
d_x = Actual stiffener thickness
E_0 = Plate thickness standard
δ = Actual plate thickness

The cost for consumables are related to the material needed to weld. It is dependent on the number of welds. The total cost of consumables is expected to be very small in comparison to the other costs involved.

$$C_c = L_i \cdot \frac{1}{s} \cdot C^0_{8x}(1 + \delta C_{8x} \cdot (d_x - E_{0x} \cdot 10^3)) \tag{4.14}$$

C_C = Cost consumables, euro per meter
L_i = Length of element
C^0_{8x} = Cost of consumables per meter
C_{8x} = Cost of deviance form standard thickness
E_0x = Stiffener thickness standard
d_x = Actual stiffener thickness

$$C_{tot} = \sum(C_l + C_c + C_s) + \sum W_F \cdot C_{plate} \tag{4.15}$$

C_{tot} = Total cost, euro per meter
W_F = Weight frame
C_{plate} = Cost of plate per kilo

4.3 Combination the object functions

To make it possible to consider both weight and cost in the optimisation, one can normalise the values and weight in the importance of the different goals by a parameter α set between one and zero. The best way to normalise the values is to find the weight and cost of a similar ship. If this is not available the user should do a quick assessment of the values expected and use these as the foundation for normalisation. Looking at Equation (4.16), we see that the object function will be varied around 1 dependent on the chosen variables.

$$f = \frac{C}{C_{init}} \cdot \alpha + \frac{W}{W_{init}} \cdot (1 - \alpha) \tag{4.16}$$

C = Cost of chosen variable combination (4.17)
C_{init} = Comparison cost (4.18)
W = Weight of chosen variable combination (4.19)
W_{init} = Comparison weight (4.20)
α = User parameter between 0 and 1 (4.21)

Chapter 5

The particle swarm optimisation

Particle Swarm Optimisation, PSO, is the algorithm used in this thesis to find the best combination of variables to get the lowest weight, price or a combination of the two. The optimisation code used in this thesis was provided by Sören Ehlers. The original source of his PSO is [Jalkanen, 2006], but was further used and developed by Sören Ehlers [Ehlers, 2012]. The algorithm was developed by [Kennedy and Eberhart]. Other optimisation methods that were looked into was a minimising MATLAB algorithm for non-linear constrained function: fmincon and the MATLAB algorithm based on genetic algorithm theory: GA. There algorithms either did not fit with large optimisation problem or proved hard to obtain as MATLAB code.

In an optimisation the object function will either target to become as big or as small as possible. The PSO will initially target to find the least value of a constrained object function, therefore the local and global best(optimal) values of the algorithm will be the smallest value of the object function that is possible to obtain while not breaking any of the constraints. If one rather wants the biggest value of the object function using the above mentioned algorithm, one only needs to run the algorithm with the negative of the object function. [Hiller and Lieberman, 2010]

To explain the PSO algorithm, one can think of it as a flock of birds looking for food. The flock share information of the location of the food and where they should land. The flock's aim is to land as close to were there is the most food as possible. In a programming perspective, each bird represents a particle which contains a set of parameters, it's location, which in the object function will give a value. Each particle has a memory of its best value. In addition all the particle share this information so there will be a "common knowledge" about the optimum value the flock has found. In our optimisation the location of each particle will be a vector

containing the design variables for the cross sections panels. The optimisation do not keep track of the individual variables, but the result of the combination for the cross section. [Kennedy and Eberhart, 1995]

The PSO is a meta-heuristic optimisation, which means that one is not assured that the solution found is the absolute best, but it may be able to find a good acceptable solution. The meta-heuristic algorithms was developed because some problem were to difficult to solve with more traditional methods, which guarantee the absolute optimal solution. That said, improvement is always good from an engineering point of view, and the PSO optimisation have proved to give good results in ship optimisation. [Hiller and Lieberman, 2010] [Ehlers, 2012]

The PSO works with a population that changes over time to find the optimum solution. In contrary to a genetic optimisation, which evolves it's population by mutation and cross-over parameters, the PSO changes it's population by varying the particles speed and location, accelerating them through the solution space. The speed varies with the local and global optimum value found so far and the position of the particle is a combination of the prior position and speed. This means that the PSO needs an initial feasible group of particles to start from to create a population. [Ehlers, 2012]

The formula for the speed in the particles next position is found by the formula in Equation (5.1). This velocity is used to find the new position of the particle as seen in Equation (5.2). The velocity is tested to see that it is within the limits set for the variables. The velocity term accelerates the particle towards a better solution, but this does not mean that the particle will land in a feasible area. As the velocity components, the particle components will also be checked that they are within the band of feasible parameters. In Figure () an example of how the particle moves and is effected by the parameters is shown. The inertia coefficient, w, is normally set between 0.8 and 1.4 and affects how far the particles are flung. In the optimisation trial this value was set to 1.4. The cognitive coefficient, c, affects how the local and global pull on the particle is weighted, normally set to 2 for each of the contributions.

$$v_{i+1} = w \cdot v_i + r_1 \cdot c_1 \cdot (P_i - v_i) + r_2 \cdot c_2 \cdot (P_g - v_i) \tag{5.1}$$

v_{i+1} = Velocity of the particle at the next position

v_i = Velocity of the particle at current position

w = Inertia coefficient.

r = random number between 0 and 1

c = Cognitive constants

P_i = Position of the particle best value

P_g = Position of the swarms best value

$$x_{i+1} = x_i + v_{i+1} \tag{5.2}$$

x_i = Location of the particle

x_{i+1} = Location of the particle in the next position

v_{i+1} = Velocity of the particle at the next position

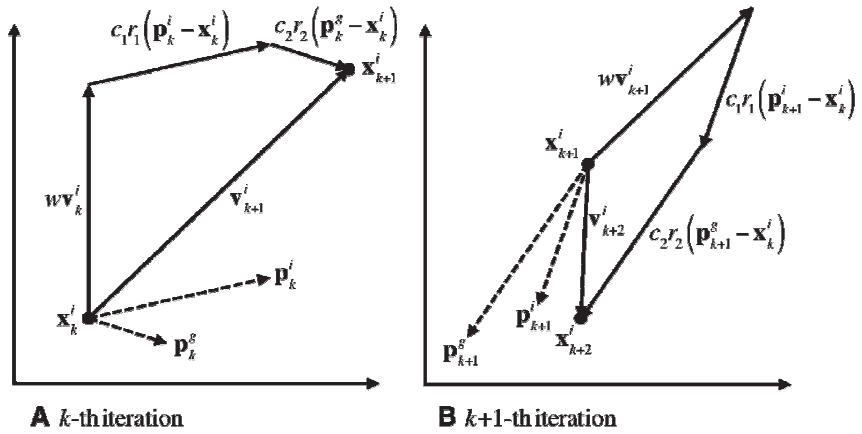


Figure 5.1: Development of a particles position [Ehlers, 2012]

Each member of the population is tested against the constraints of the problem at hand. The best values, both feasible and infeasible, of the object function is stored, and the best feasible is used as the basis for creating a new population.

In Figure (5.2) the flow chart of the procedure is presented. When starting the PSO, the user must specify a vector of feasible values for each of the variables.

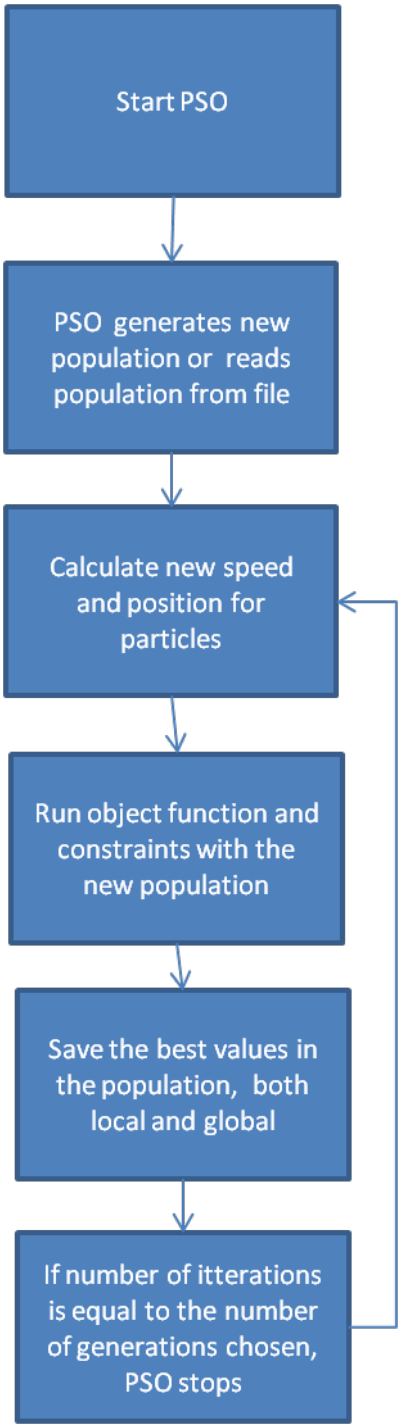


Figure 5.2: Particle swarm optimisation flow chart

Chapter 6

Program build-up

In creating the optimisation program for this thesis, code for the PSO procedure was obtained from Sören Ehlers. The code for the constraints and object function was built and adapted until it could fit into and around the PSO. Especially important was making good code for the geometrical input, so that the constraints and object function got the right input data. The flow chart of the program is found in Figure (6.1). All the code in found in Appendix F.

6.1 Input needed

In the start of a optimisation trial for a ship, the user must supply the geometry of the cross section and define each panel according to Table (6.1) to obtain the correct pressure. The cross section must be represented in such a way that each panel is an element, connected with a thickness and distance between the stiffeners, and the location of the elements nodes are given by the user from left to right and bottom to top. A simplification that must be done when defining the cross section geometry is that each element ends where another begins, with the exception of the hatch panel, numbered as 20 in Figure (7.4).

The user must also provide corrosion additions, t_c for each panel, in addition to two voluntary coefficients for per panel, the t_l and k coefficient in Equation (3.2), which will override the default setting in the program if they are given. The default settings if found in Appendix D. In the file *profiles.m* the standard HP profiles allowed to use can be modified. Last but not least all the basic ship parameters in Table (7.1) must be given in the *Userinput.txt* file.

Panel type	Description
KEEL	In a given width along the centreline
BOTTOM_OUT	Panel in the bottom
BOTTOM_IN_C	Double bottom panel under cargo
BOTTOM_IN_GIRDER	Longitudinal girder in double bottom
BOTTOM_IN	Double bottom panel not under cargo
SIDE_IN	Internal vertical panel
SIDE_EXT	External vertical panel
BULKHEAD_EXT	External vertical panel
DECK_ST	Strength deck
DECK	top or bottom of tank

Table 6.1: Panel input choices

6.2 Presentation of programs

In the main program file, *Run.m*, the names of the input files must be stated before the rest of the functions can be started. The user can in this file specify if he or she wants to do a direct run, or an optimisation. In the optimisation option one can choose to name a file which contains a feasible population to base the optimisation on, or make a feasible population with the program. Depending on the magnitude of variables in the problem, the search size for the feasible population may be changed. The search size decides how many combinations is to be tried to find a feasible population. The feasible population needs to contain as many feasible combinations for the cross section as there is variables in the optimisation. If one wants to run several optimisations one can use *Runalfa.m* which is essentially the same as *Run.m* but it contain loops which gives multiple solutions for chosen α in Equation (4.16).

The function *inputvalues.m* reads the input files and makes the information available for the rest of the program. It also records the length of each panel and checks that all the panels are connected, which is important for finding geometrical values later on.

The functions *globalsectionmod.m* and *platepressure.m* calculates, respectively, the section modulus demand for the cross section and the local pressure connected to each panel. These calculations are independent of the choices of the design variables.

When the user chose the optimisation option, the the function *StructOpt.m* is used. If there is no initial population presented, the code is made to produce its

own feasible population by looking for feasible combinations of the variables from a feasible set, like the on in on in Appendix E. The details for the optimisation algorithm is found in Chapter 5. What is important here is that the PSO finds the variables which is to be used to calculates the object function and the constraints of the problem in *particle_fun.m* and *paramfun.m* inside *StructOpt.m*.

Either for the optimisation or the direct run, the following functions is used to test the chosen variables. *stipar.m* is a function connecting the local demand of the stiffeners to the closest area of a approved stiffeners in the sub-function *profiles.m*. This function also defined the maximum allowed stiffener dependent on what the user has chosen. The restrictions are found in *constrfun.m*. The values returned are the section modulus calculated based on the variables, the minimum demand of the thickness and the section modulus relating to the transverse frame. *paramfun.m* uses the functions *weigth.m* and *costsimple.m* to find, based on a parameter α, the weighted object function. *weigth.m* and *costsimple.m* use the formulas explained in Chapter 4.

In the end, the variables are used to produce figures showing the cost distribution on the different segments. The development of the constraints and best values found in the PSO code is also plotted. If the user has specified it, the structures panels are plotted in a figure and the values from the optimisation are written to a text file.

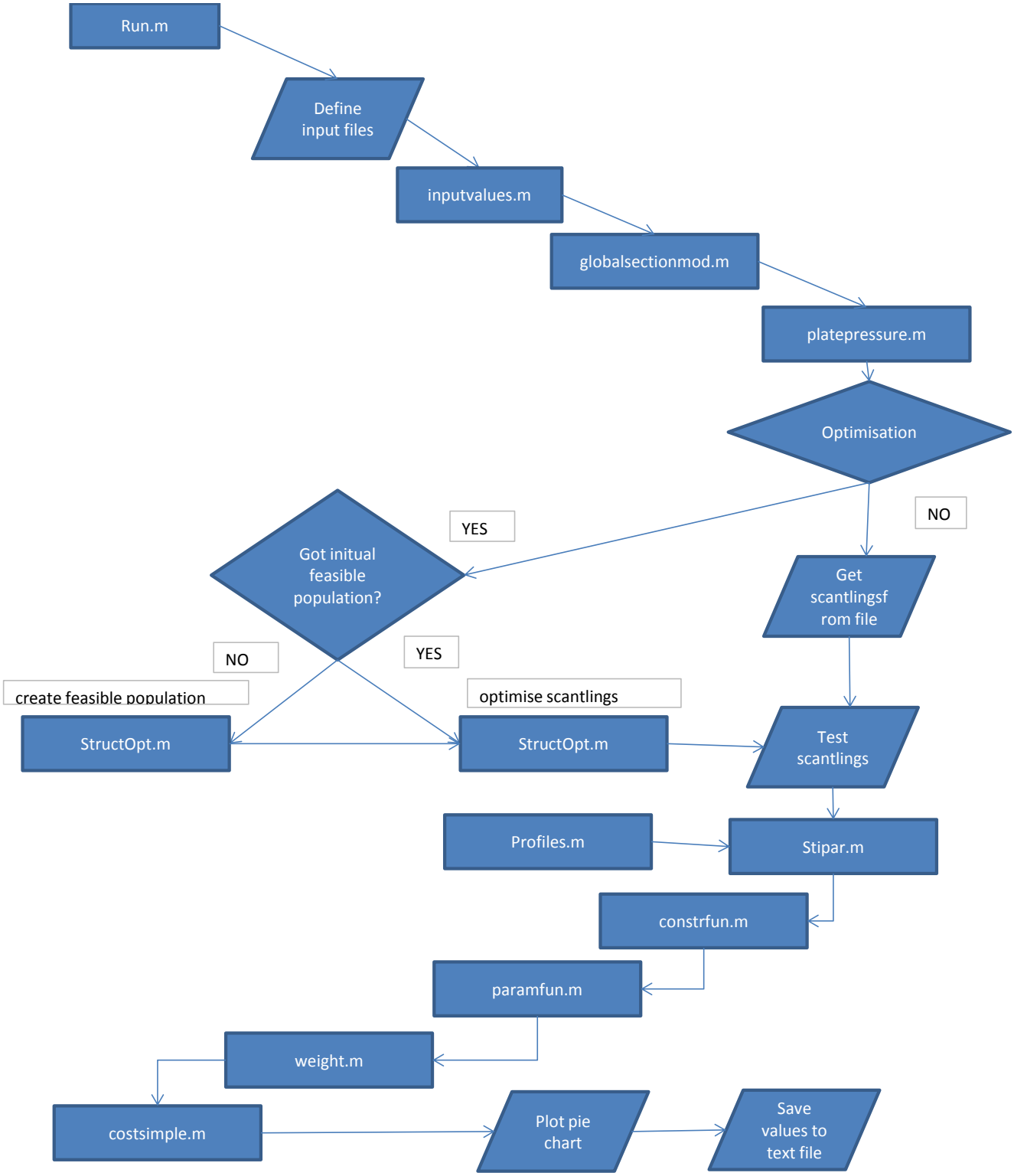


Figure 6.1: MATLAB code flow chart

Chapter 7

Optimisation case study

To test the optimisation program a example ship given by Rolls Royce was used as a base. The ship in question is a 100 meter long cargo ship, which therefore just fit into the rules over 100 meter. The ship provided is designed to carry lumber in the cargo hold and coils and stones on the hatch cover. The hatch calculations are not performed in this program. The cargo area is 57.4 meters long, it has four compartments and is estimated by Rolls Royce to weigh 751.8 ton. A drawing of the ship is found in Figure (7.1).

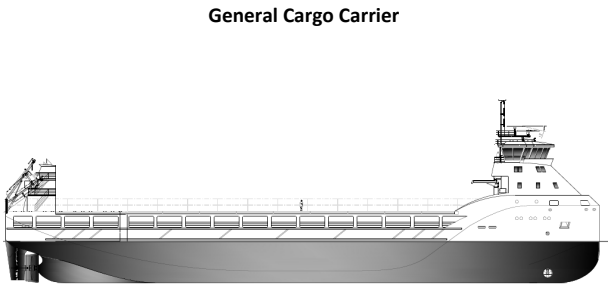


Figure 7.1: Drawing of cargo ship

The cargo ship cross section can be seen in Figure (7.2). The input file, shown in Figure (7.3), represents the geometry of the cross section and generates the simplified cross section in Figure (7.4). The parameters T_0 and K, the input for t_l and k in Equation (3.2), are set to zero, letting these parameters be defined by the default options, presented in Appendix D. The corrosion addition is for simplistic set to 1 mm for all panels. As seen in Figure (7.4), one only need half the cross section because of the symmetry expected for this kind of ship. If there is not symmetry about the center line, the whole cross section geometry and panel

definition must me given and the coefficient Mirrored in Figure (7.5) must be set to 1.

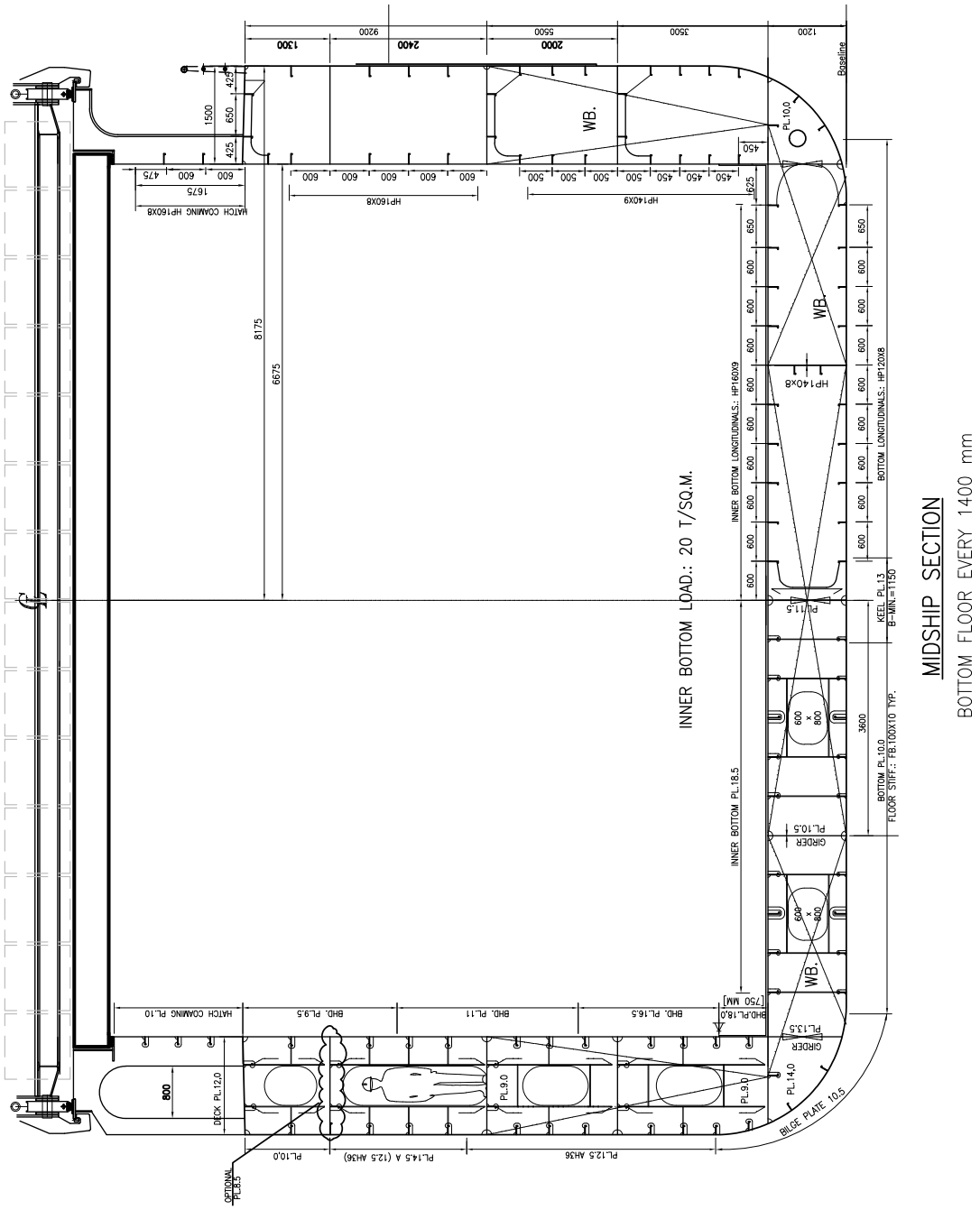


Figure 7.2: Cross section of mid ship area which is to be optimised

TAG	X1 [m]	Y1[m]	X2[m]	Y2[m]	T0[mm]	K[]	T_C[mm]
KEEL	0	0	0,575	0	0	0	1
BOTTOM_OUT	0,575	0	3,575	0	0	0	1
BOTTOM_OUT	3,575	0	6,65	0	0	0	1
BOTTOM_OUT	6,65	0	8,15	1,2	0	0	1
BOTTOM_IN_C	0	1,2	3,575	1,2	0	0	1
BOTTOM_IN_C	3,575	1,2	6,65	1,2	0	0	1
BOTTOM_IN	6,65	1,2	8,15	1,2	0	0	1
BOTTOM_IN_G	0	0	0	1,2	0	0	1
BOTTOM_IN_G	3,575	0	3,575	1,2	0	0	1
BOTTOM_IN_G	6,65	0	6,65	1,2	0	0	1
SIDE_EXT	8,15	1,2	8,15	4,7	0	0	1
SIDE_EXT	8,15	4,7	8,15	6,7	0	0	1
SIDE_EXT	8,15	6,7	8,15	9,1	0	0	1
BULKHEAD	6,65	1,2	6,65	4,7	0	0	1
BULKHEAD	6,65	4,7	6,65	6,7	0	0	1
BULKHEAD	6,65	6,7	6,65	9,1	0	0	1
DECK_ST	6,65	9,1	8,15	9,1	0	0	1
DECK_TWEEN	6,65	4,7	8,15	4,7	0	0	1
DECK_TWEEN	6,65	6,7	8,15	6,7	0	0	1
BULKHEAD	6,65	9,1	6,65	10,775	0	0	1

Figure 7.3: Geometric input from user

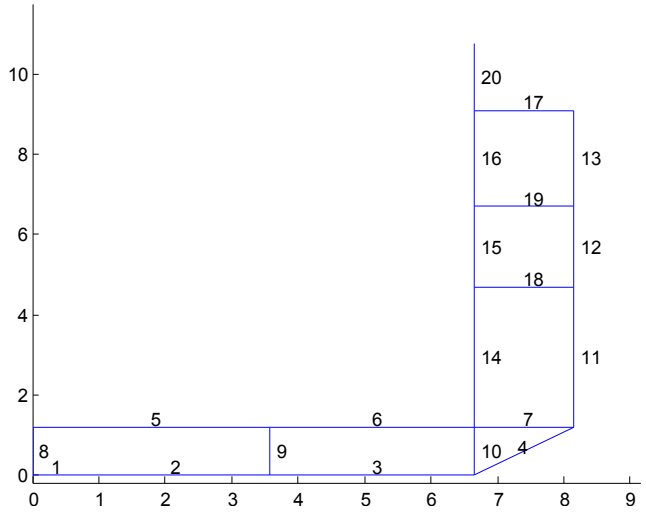


Figure 7.4: Geometry of cross section with panel numbering

The general ship information is found in Figure (7.5). This information is used to form the constraints in the optimisation. Note that the cargo density is given in ton/m^2 and this is accounted for in the pressure calculation.

Description	input
Rule Length [m]	100
Rule dept [m]	9,2
Rule draught [m]	7
Rule breadth [m]	16,3
Speed [knot]	12
Block coef. []	0,757
Material factor []	1
Restriction R []	2
alfa []	0,8
Steel dencity [kg/m^3]	7800,00
Cargo dencity [t/m^2]	20
Heigth of air pipe over main deck [m]	1,6
Mirrored [1=no or 2=yes]	2
length of stiffeners [m]	1,4

Figure 7.5: Ship information given by user

The initial cost and initial weight of the cross section is needed to normalise the optimisation. If the user has initial scantlings, these can be used in the direct run option in the code to get the initial weight and cost. If this is done, it is important not to add corrosion in Figure (7.4) it the scantings chosen already have included this. If initial scantlings are not available, the user can insert reasonable numbers from experience. The initial scantlings are introduced as shown in Figure (7.6), with thickness, stiffener spacing and HP profile area for each panel, in a file named *initalscantlings.txt*. The order of the scantlings must be in the same order as the geometry has been represented. The values generated by the direct run is found in Table (7.1). The cost distribution is shown in Figure (7.7). Seen here, the labour is the major expense and the weight calculated per meter is fairly close to the weight given per meter by Rolls Royce. It should be noted that the weight deviance in (7.1) is caused by that the program does not include the transverse bulkheads in the cargo area. If the five bulkheads in the cargo area have a thickness of 10 [mm], the calculated weight and the weight given by Rolls Royce is almost the same. This means that the neglection of cut outs and brackets evens out in the weight calculation.

	Calculated by programme	Values given by Rolls Royce	unit
Weight	12401.4	13098	$\frac{kg}{m}$
Cost per meter	35276.5		$\frac{euro}{m}$
Cost per kilo	2.74	3	$\frac{euro}{kg}$

Table 7.1: Values from example ship provided by Rolls Royce

Tag	plate [mm]	stiffener [m]	area [cm^2]
KEEL	13	0,6	11,7
BOTTOM_OUT	10	0,6	11,7
BOTTOM_OUT	10	0,6	11,7
BOTTOM_OUT	10,5	0,6	11,7
BOTTOM_IN_C	18	0,6	17,8
BOTTOM_IN_C	18	0,6	17,8
BOTTOM_IN	18	1	17,8
BOTTOM_IN_G	12	0	0
BOTTOM_IN_G	10,5	0,6	13,8
BOTTOM_IN_G	10	0	0
SIDE_EXT	12,5	0,45	15,2
SIDE_EXT	12,5	0,5	15,2
SIDE_EXT	14,5	0,6	16,2
BULKHEAD	15	0,45	15,2
BULKHEAD	15	0,5	15,2
BULKHEAD	12	0,6	16,2
DECK_ST	18,5	0,45	16,2
DECK_TWEEN	9,5	0,45	15,3
DECK_TWEEN	10,5	0,45	16,2
BULKHEAD	13	0,6	16,2

Figure 7.6: Initial scantlings for the example ship

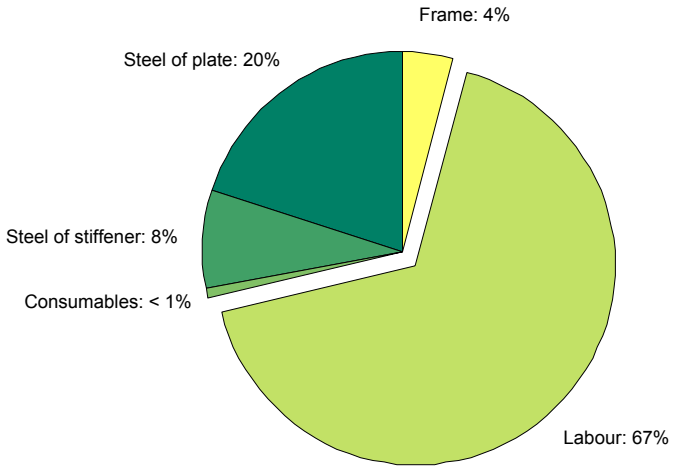


Figure 7.7: Cost distribution of one frame in the example ship

In the optimisation, the plate thickness and the distance between stiffeners for each

panel is chosen as design variables. To make the discrete optimisation, a feasible area for the variables must be defined. As there is 2 variables per panel, there must be defined 40 feasible vectors, in MATLAB this is easily done in a loop. The feasible set for the variables was set for thickness from the largest of 6 mm and the minimum thickness demand, up to 27 mm, with 1 mm steps. The feasible set for the stiffeners was set from 0.4 m, with 0.05 m in step distance, to the smallest of the length of the panel or 1 m. The feasible search area applied in the optimisation trial is found in Appendix E.

Chapter 8

Results from the optimisation trial

In this section the behaviour of the PSO will be explained, in terms of the development of the optimal solution and the dependence on the initial population.

The results from the optimisations with regards to lowest weight, lowest cost and a combination of these two previous objectives will be presented. The scantlings found in there optimisations will be compared to the values chosen by Rolls Royce in Chapter 7 to see if the optimisation program is able to obtain better solutions.

8.1 PSO behaviour

When the particles in the population is develop, the best solution found in each iteration may improve the value of the object function, as seen exemplified in Figure (8.1). Each particle contains alternative cross section scantlings and each change in the graph represent cross section found which improves the value of the object function. If all the particle in the particles in the population at each generation whas plotted, there would be scatted values above and below the plotted values, representing values not feasible or higher than the best. The graph flattens out when spread of the particles no longer find better solutions and all the particles has been drawn towards the best position. The development of the highest constraint connected to the particle giving the best object value can be seen in Figure (8.2). Only the highest constraint of the particle which provides the best solution is plotted, as this ensures that all the other constraints are also in the feasible range.

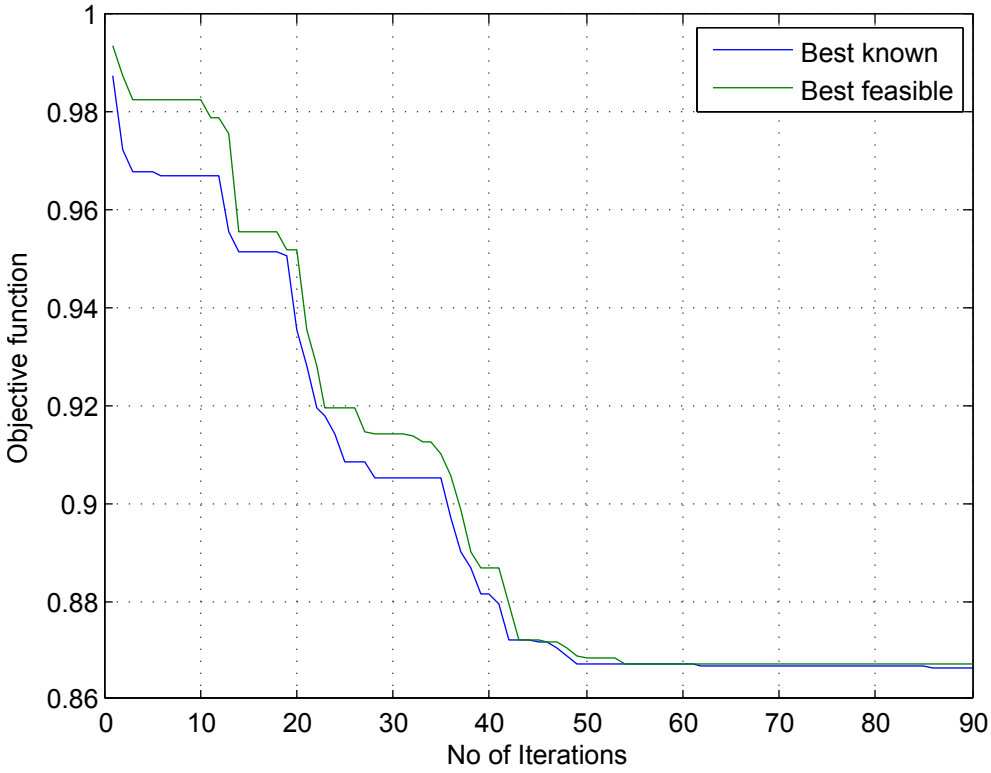


Figure 8.1: Development of best value of a object function

When one operates with as may variables as in a cross section optimisation, there is a small chance of finding the same combination of variables each time the optimisation is run, even though one uses the same initial feasible population. With 40 variables and a wide range of allowed values for each variable, there is a extreme amount of possible combinations to be found. This means that presenting the improvement of weight and cost in % from the initial values, is only an indication of what the improvement may be. Running a optimisation multiple times, based on the came initial optimisation, might give a better or worse number than the first trial. As seen in Figure (8.3), the weight optimisation, using $\alpha = 0$ in Equation (4.16), have two clusters of solutions, containing five optimisations each, where one has managed to find numbers which reduced the weight. These clusters is caused by using different feasible initial areas, which affect the search area of the optimisation. In the case of the weight optimisation the values are spread around 0.84 and 0.89 on the weight axis. The reason why they do not find the same optimal value in the different runs is because of the acceleration through the space, described in Equation (5.1), which is affected by the local and global best position of the particles. If the optimisation finds a local minimum, it might get stuck there, if not a better solution is found to pull it out. The optimisation for cost,

and the optimisation where cost and weight is combined, also have these clusters of solutions where the improvement of the object function varies.

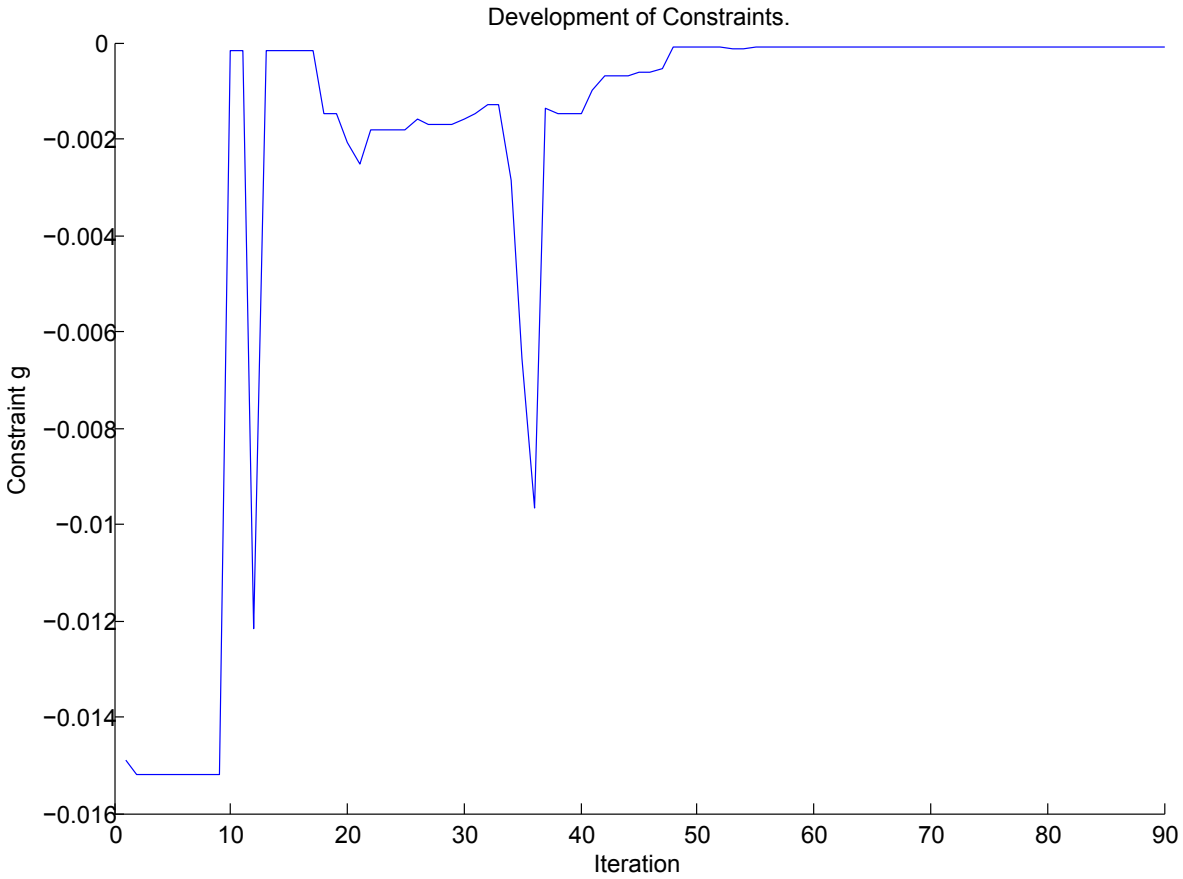


Figure 8.2: Development of the constraint with the highest value for the best particle at each iteration

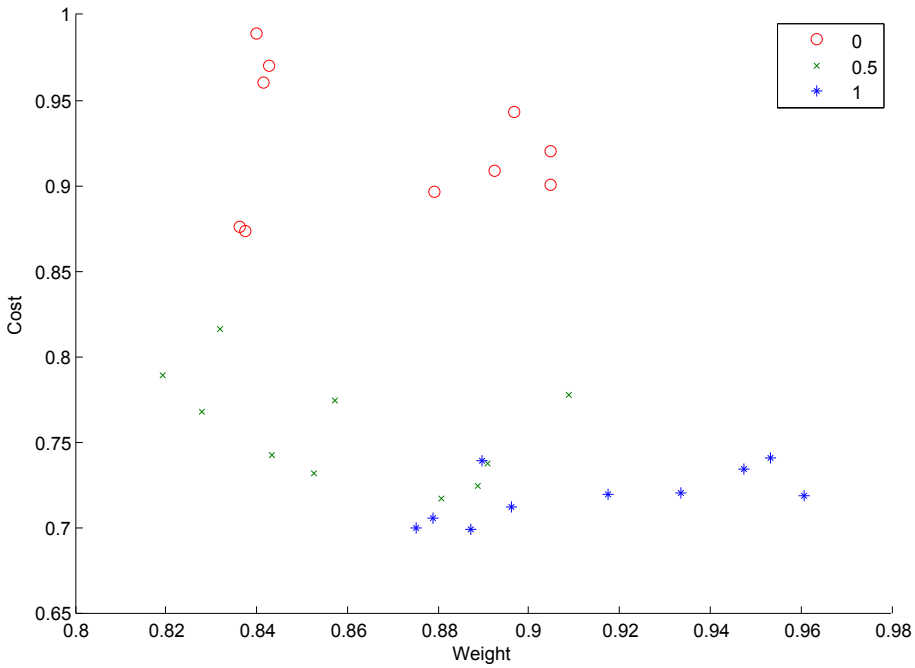


Figure 8.3: Optimisation of different object functions represented by the α factor in Equation (4.16)

8.2 Results from the optimisations

Weight optimisation

In Table (8.2), the scantlings for the optimisation when α is set to zero, creating a pure weight optimisation, is found. The scantling in this table is presented with plate thickness without corrosion addition, t_p, distance between stiffener, s, HP stiffener dimensions and the thickness requirements t_{d1} and t_{d2}. Here t_{d2} refers to the formula in Equation (3.2), and is rounded up to the closest whole number and used as the lower bound in the feasible area. t_{d1} is calculated from Equation (3.1) and is in this optimisation small because of the low stiffener distance.

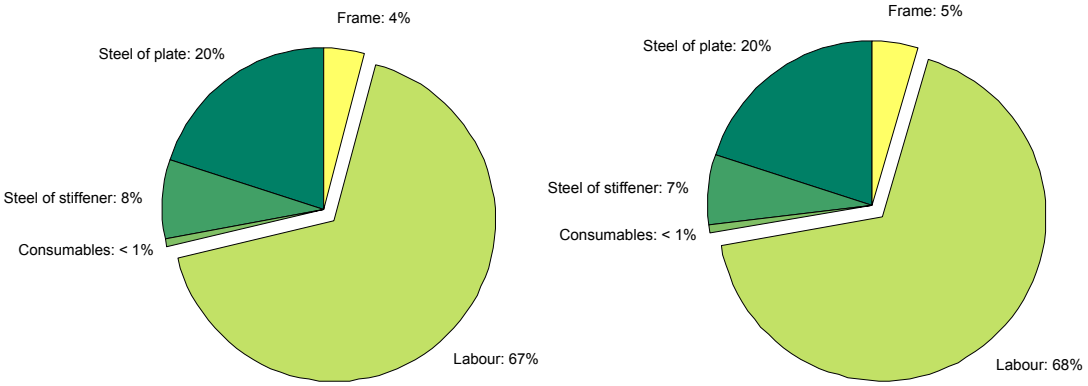
	Result	Reduction
weight [kg/m]	11068.9	10.7%
cost [euro/m]	31543.6	10.5%
Cost cargo area [euro/kg]	2.85	-1.8%

Table 8.1: Results for weight optimisation , $\alpha = 0$

P_n	t_p [mm]	s [m]	HP stiffener [mm]	t_{d2} [mm]	t_{d1} [mm]
1	12	0.40	120 x 6	12.0	4.78
2	9	0.40	120 x 6	9.0	4.78
3	9	0.70	120 x 8	9.0	7.95
4	9	0.40	120 x 6	9.0	4.88
5	17	0.95	240 x 10	10.0	16.99
6	10	0.40	160 x 9	10.0	8.27
7	9	0.80	140 x 8	9.0	8.81
8	10	0.40	120 x 8	10.0	5.23
9	8	0.40	120 x 8	8.0	5.23
10	9	0.70	140 x 8	8.0	8.70
11	10	1.00	140 x 9	9.0	9.92
12	9	1.00	120 x 8	9.0	7.59
13	9	1.00	120 x 6	9.0	5.87
14	8	0.75	140 x 8	8.0	7.89
15	8	0.85	120 x 8	8.0	7.26
16	8	1.00	120 x 8	8.0	7.43
17	8	1.00	100 x 6	7.5	4.18
18	27	1.00	140 x 8	7.5	9.31
19	8	0.40	120 x 6	7.5	3.84
20	27	0.40	120 x 6	8.0	3.23

Table 8.2: Scantlings for weight optimisation , $\alpha = 0$

In Table (8.1) weight and cost of the cross section is presented and compared to the initial values in Chapter 7. There can be see that in lowering the the weight also has reduced the cost. In Figure (8.4) where the cost distribution of the initial cross section and the cross section produced by the optimisation is shown, one sees that the change from the initial distribution is a small increase of the labour cost involved. The cost per kg is increased because of a larger decrease in weight than cost.



(a) Initial distribution (b) Distribution after optimisation

Figure 8.4: Comparison of cost distribution of weight optimisation

Cost optimisation

The scantlings for the cost optimisation, seen in Table (8.4), show that the most significant change from the initial scantlings is that the stiffener distance is pushed up towards the top boundary of the feasible set. This causes bigger HP- profiles than in the weight optimisation. In this optimisation, the t_{d1} constraint become mores significant and affect some of the plate thickness requirements.

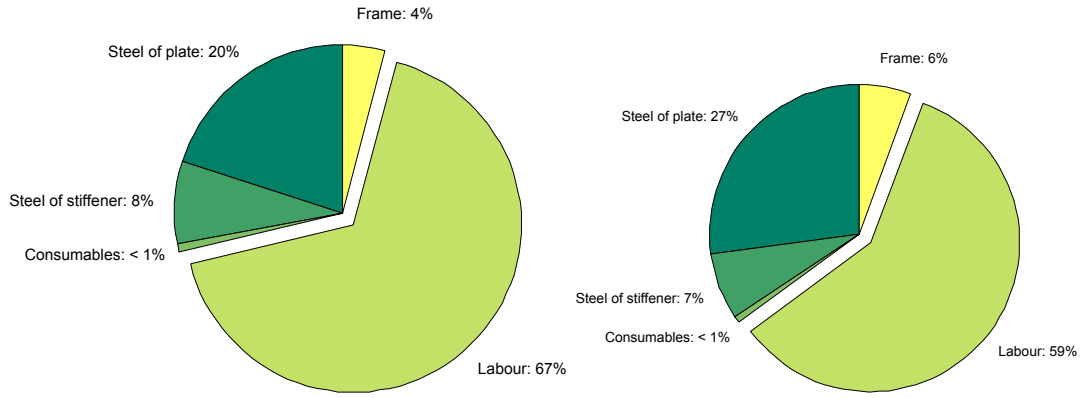
In Table (8.3) it becomes apparent that the cost reduction also causes a reduction in the weight. The cost of the labour in the total cost is in this optimisation reduced, as seen in Figure (8.5).

	Result	Reduction
weight [kg/m]	11536.5	7%
cost [euro/m]	25268.2	28.3%
Cost cargo area [euro/kg]	2.2	23%

Table 8.3: Results for cost optimisation , $\alpha = 1$

P_n	t_p [mm]	s [m]	HP stiffener [mm]	t_{d2} [mm]	t_{d1} [mm]
1	12	0.40	120 x 6	12.0	4.78
2	11	1.00	140 x 8	9.0	10.14
3	10	0.95	140 x 8	9.0	9.82
4	11	1.00	140 x 9	9.0	10.36
5	16	0.80	220 x 10	10.0	15.15
6	17	0.95	240 x 10	10.0	16.99
7	14	1.00	140 x 9	9.0	10.21
8	13	1.00	160 x 8	10.0	11.11
9	11	0.95	160 x 8	8.0	10.76
10	8	0.60	140 x 8	8.0	7.74
11	10	1.00	140 x 9	9.0	9.92
12	9	1.00	120 x 8	9.0	7.59
13	9	1.00	120 x 6	9.0	5.87
14	10	1.00	140 x 9	8.0	9.57
15	8	0.95	140 x 8	8.0	7.81
16	9	1.00	120 x 8	8.0	7.43
17	20	1.00	100 x 6	7.5	4.18
18	18	1.00	140 x 8	7.5	9.31
19	8	0.90	120 x 8	7.5	7.63
20	19	1.00	120 x 8	8.0	6.86

Table 8.4: Scantlings for cost optimisation , $\alpha = 1$



(a) Initial distribution (b) Distribution after optimisation

Figure 8.5: Comparison of cost distribution of cost optimisation

Mixed optimisation

In a third optimisation run, the object function is a combination of the weight and cost function by using α equal to 0.5 in Equation (4.16). The scanlings produced in Table (8.6) have a mix of low and high stiffener distances, and is in that way more similar to the initial cross section than the scantlings produced by the two other optimisation. Most of the panels tend towards the top feasible range for s.

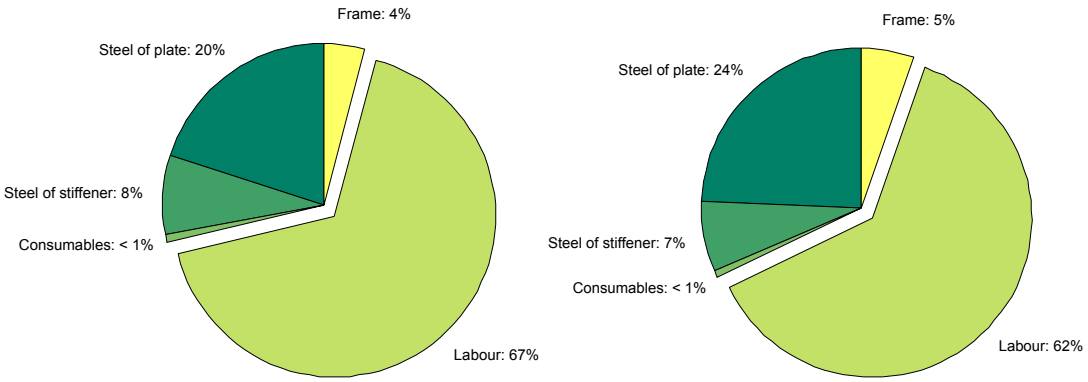
	Result	Reduction
weight [kg/m]	11163.7	10%
cost [euro/m]	26848.7	23.9%
Cost cargo area [euro/kg]	2.4	15.4%

Table 8.5: Results for mixed optimisation , $\alpha = 0.5$

The cost distribution shown in Figure(8.6) show a reduction in the cost related to the labour compared to the initial cross section, and the results in Table (8.5) indicate a significant reduction both in weight and cost has been achieved in this optimisation.

P_n	t_p [mm]	s [m]	HP stiffener [mm]	t_{d2} [mm]	t_{d1} [mm]
1	12	0.40	120 x 6	12.0	4.78
2	11	1.00	140 x 8	9.0	10.14
3	11	1.00	140 x 8	9.0	10.14
4	11	1.00	140 x 9	9.0	10.36
5	17	0.95	240 x 10	10.0	16.99
6	10	0.40	160 x 9	10.0	8.27
7	10	0.40	100 x 8	9.0	4.81
8	12	1.00	160 x 8	10.0	11.11
9	12	1.00	160 x 8	8.0	11.11
10	11	0.95	160 x 8	8.0	10.76
11	9	0.85	140 x 8	9.0	8.93
12	9	0.95	120 x 8	9.0	7.35
13	9	1.00	120 x 6	9.0	5.87
14	10	1.00	140 x 9	8.0	9.57
15	8	0.95	140 x 8	8.0	7.81
16	8	1.00	120 x 8	8.0	7.43
17	27	1.00	100 x 6	7.5	4.18
18	8	0.75	120 x 8	7.5	7.68
19	15	1.00	120 x 8	7.5	8.16
20	13	1.00	120 x 8	8.0	6.86

Table 8.6: Scantlings for mixed optimisation , $\alpha = 0.5$



(a) Initial distribution (b) Distribution after optimisation

Figure 8.6: Comparison of cost distribution of mixed optimisation

8.3 Discussion

The influence of the constraints

Regarding the thickness constraint, one can see that in the weight optimisation in Table (8.2) that most of the t values are near, or on the lowest feasible value, defined by t_{d2} rounded up to a whole number. In the pure cost optimisation i Table (8.4), t_{d1} starts effecting the choice of plate thickness because of the increased distance between the stiffeners. The latter also happens in the mixed optimisation.

As seen in the different scantling found by the optimisations above, there are some t values which are close or on the feasible upper limit in the feasible set. The feasible set for the t values was made so that the behaviour of the optimisation would be seen. The t values was not forced there by thickness requirement, but rather the requirement for fulfilling the cross sections modulus. The panels which get these "freak" values seems random and include both vertical and horizontal plates, both close and far away from the neutral axis. If the panels chosen were only vertical panels placed away from the neutral axis, these would give a high contribution to the section modulus because of the parallel axis theorem.This would allow the other thicknesses to stay close to the minimum demand.

The demand on the transverse girders do not seem to be critical for any of the optimisations. This can be seen in Table (8.7) were the values for the DNV demand for the cross section, Z_{cs} is presented. The demand for the bottom transverse beam,Z_{dnv_b}, and the side demand Z_{dnv_s} is also presented, together with the actual values based on the scantlings in the different optimisations. For all the cross sections, the section modulus's are close to the DNV demand.

	Z_{cs}	Z_{dnv_s}	Z_{dnv_b}
DNV demand	2.0894	0.0044	0.0139
α	Z_{built}	Z_s	Z_b
0	2.1129	0.0197	0.0178
0.5	2.0915	0.0197	0.0204
1	2.1068	0.0202	0.0195

Table 8.7: Transverse beam restrictions and values. All values in 10^6

	stiffener area $[m^2]$	plate area $[m^2]$	Total$[m^2]$
Initial	0.23	1.13	1.36
Weight opt.	0.18	1.01	1.19
Cost opt.	0.14	1.09	1.24
Mixed opt.	0.15	1.03	1.18

Table 8.8: Area of cross section contribution

The change in the values and scantlings from the initial calculations, and error sources

The improvement in the different optimisations have all both weight and cost reduction to some extent. The reason for this can be seen in Table(8.8), the numbers here are developed by the calculated cost for each of the optimisations and the cost distribution. Because of the areas influence on the object function, the reduction of both stiffener and plate area, improve the weight and the cost of the ship. The deviance in the values from the initial scantlings and the optimisation chosen values, may be due to the limited sett of constraints from he DNV rules used in the program. The initial values chosen by Rolls Royce may have been based on other rules, or the values may have been adapted on the base of the designers experience with similar ships.

An factor that influences the cost values produced, is that the parameter K in Equation (4.11), representing the connecting the amount of steel to hours work. This value might have been sett to high, seeing that the labour cost always is much higher than 50% of the total cost of the build. This value would most correctly be sett by the yard in question, the same goes for the rest of the cost parameters.

In a late stage, it was discovered that the corrosion addition was not added in the plate preparation parameter P_{10}. This error could both reduce and increase the cost of the labour, depending how close the thickness variable is to the reference thickness in Equation (4.11). The error was corrected in the code and the effect was tested in a direct run with the scantlings produced for the mixed optimisation. The resulting increase in weight by only 0.3%. With this low magnitude, the error seems not to effect the result produced in a critical way. Another mistake discovered was an error in the code representing the area of the stiffeners in Equation (4.5). In the code the part of the equation which was to be in the power of two was neglected not represented correctly. Investigating the error, showed the effective error was very small and on the conservative side, causing in a few cases the HP-stiffeners chosen to be higher than needed. The deviation of the wrong representation, here noted as x, and the correct representation, x^2, can bee seen in Figure (8.7).

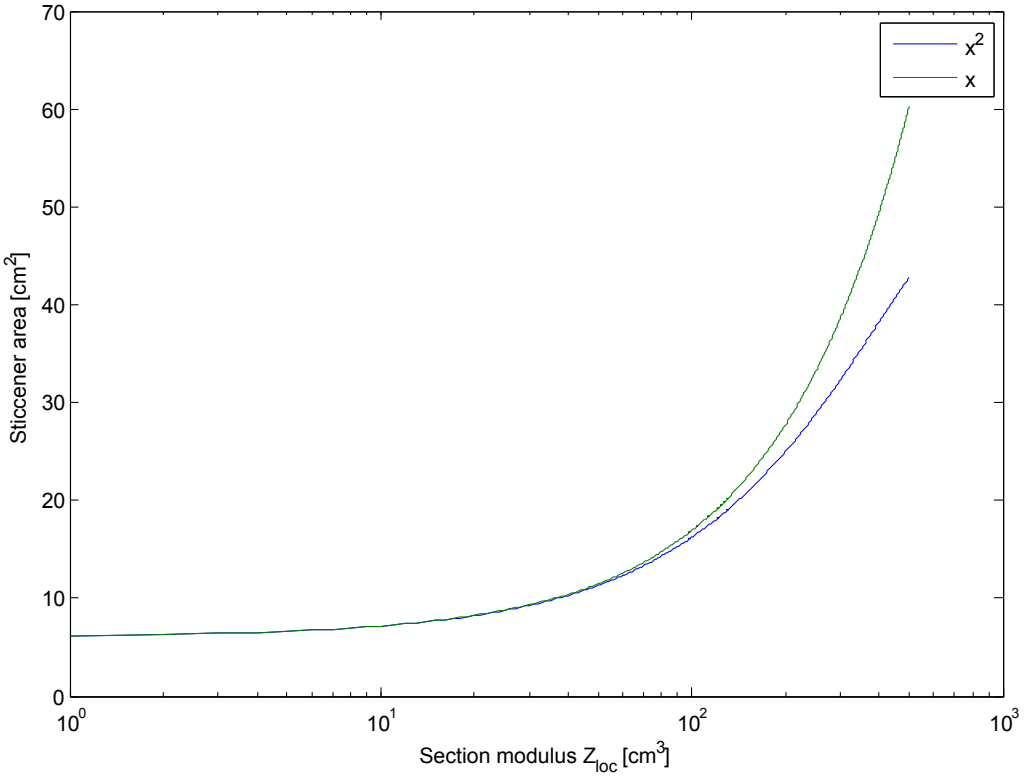


Figure 8.7: Divians in Area formula with and without the first part in the power of two.

The weight optimisation effect on the scantlings

In Table (8.2) the optimisation follows the initial assumption and chooses low s with a few exceptions. The exceptions are either if the minimum weight point is affected by the constraints, or that the optimisation was unable to find a better particle containing a lower number. Examples for these panels are panel 3, shown in Figure(8.8), panel 5 in Figure(8.9) and 12 in Figure(8.10). Panel 3 finds its minimum weight point where the added thickness from the stiffener is in a local dip, though the change in the stiffener contribution is very small. In panel 5 the minimum weight point is in the cross section between the two thickness demand. In panel 12, the minimum demand set by empirical formula is so high that the minimum weight point occurs when the stiffener contribution is at it's lowest, namely at the highest s. This is directly affected by the choice of allowed stiffeners. If there larger leaps in he areas of the allowed stiffeenrs, this would affect the minimum weight point position than what is seen in these graphs, there the stiffener contribution varies with under 1 mm from the smallest s to the largest.

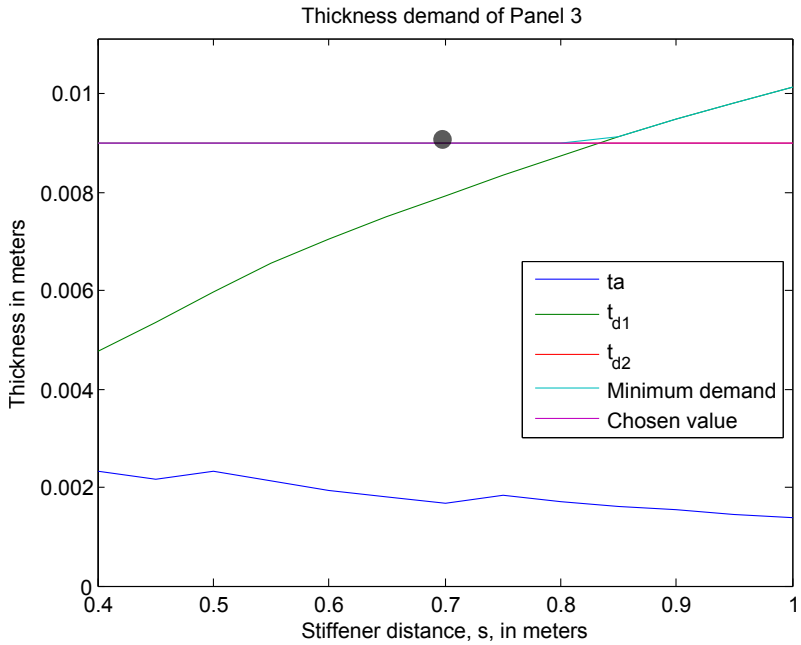


Figure 8.8: Thickness depending on s for panel 3, weight optimisation

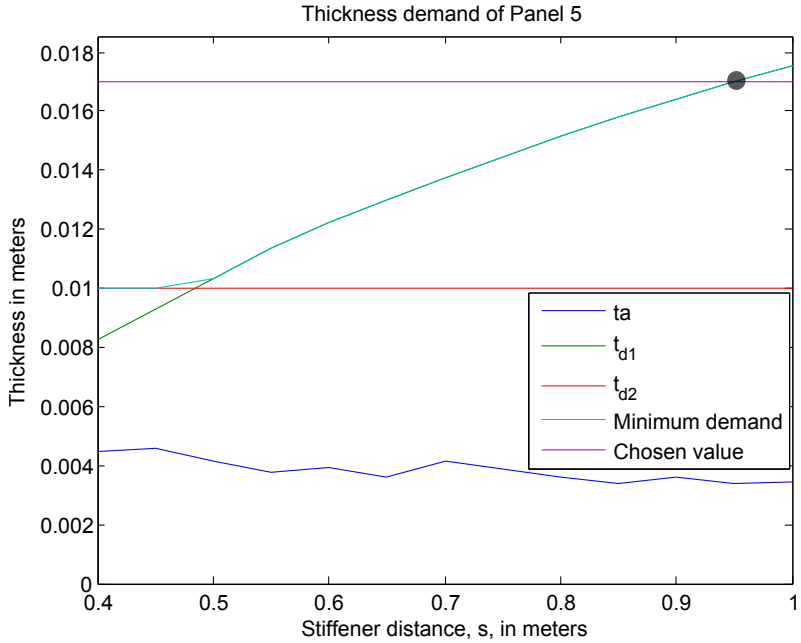


Figure 8.9: Thickness depending on s for panel 5, weight optimisation

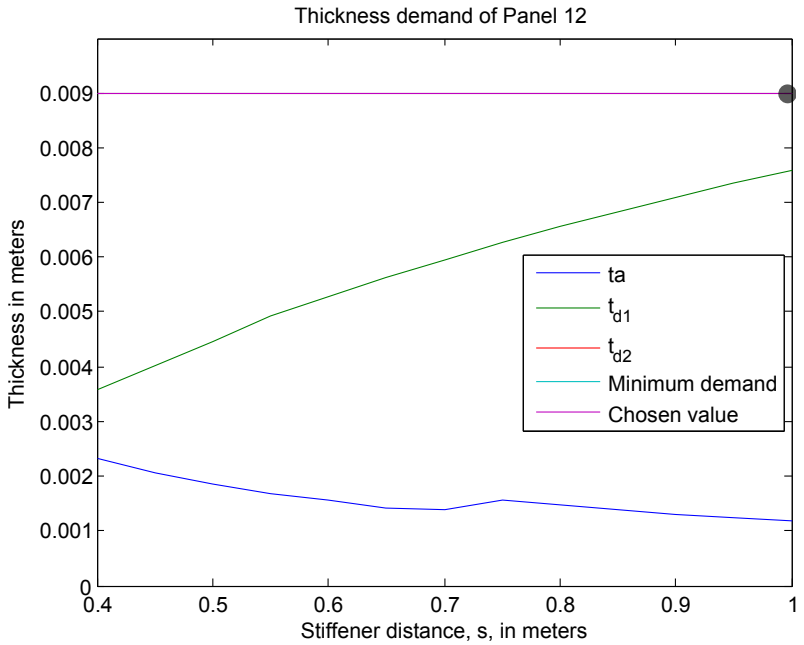
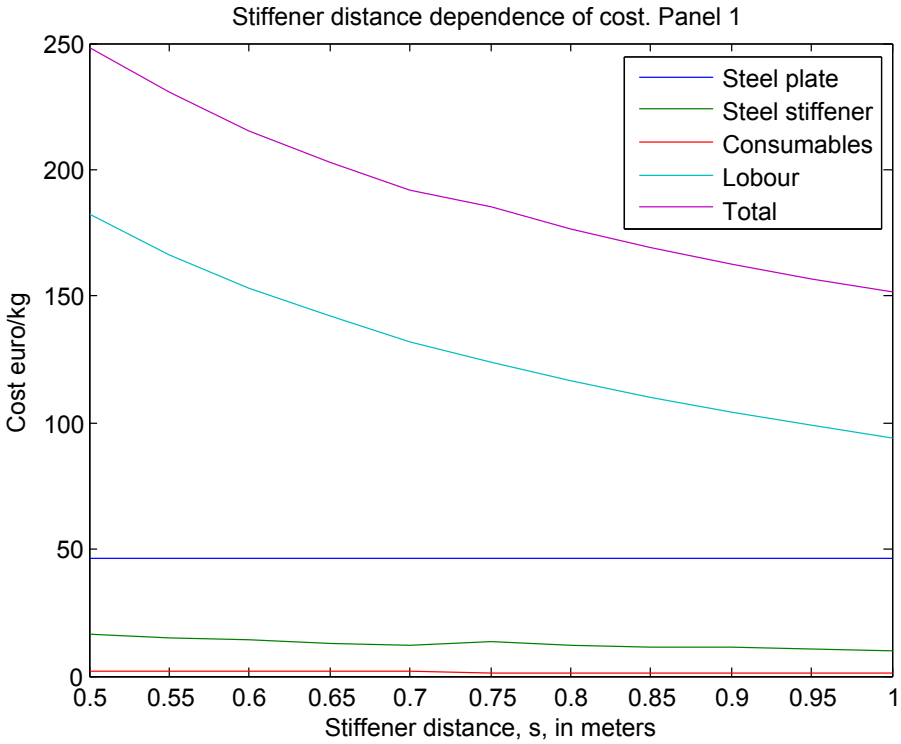


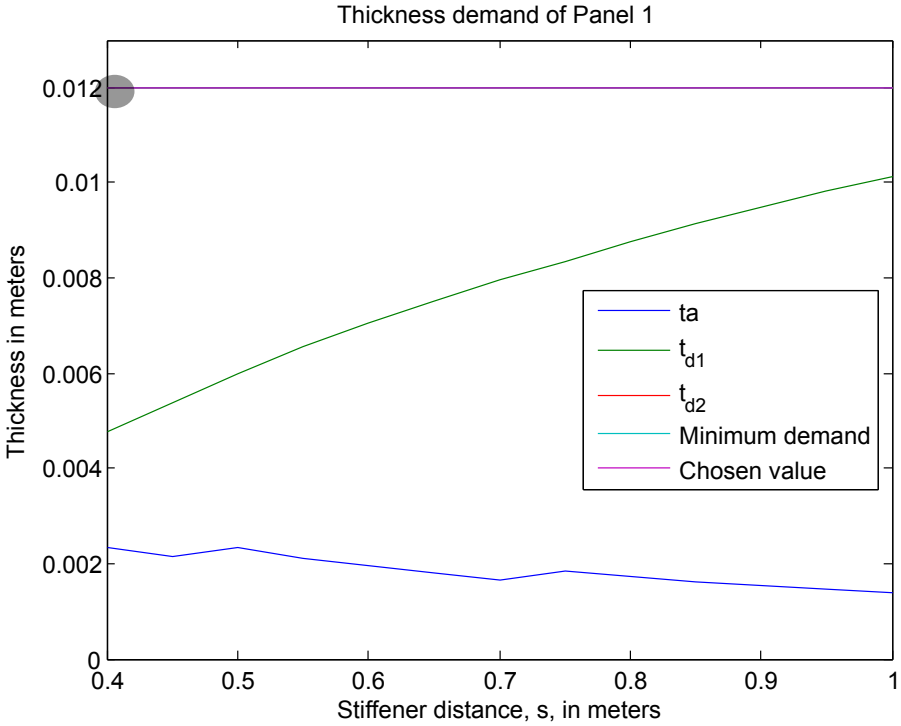
Figure 8.10: Thickness depending on s for panel 12, weight optimisation

The cost optimisations effect on the scantlings

In the cost optimisation, the scantlings in Table (8.4) primarily follow our assumption; that the cost is reduced by using high stiffener distances. Also here there are a few exceptions with panels with low stiffener distance. In panel 1, shown in Figure(8.11), the stiffener spacing is set to be the lowest and most expensive stiffener distance. The cost distribution in Figure (8.11a) is an example of the prevalent trend of all the panels; the higher the distance between the stiffeners, the lower the cost. In panel 5, seen in Figure(8.12) and in panel 10 in Figure (8.13) the stiffener distance chosen does not give the minimum weight, but it has avoided the smallest stiffener distance. The case here is that the particles in the generations has not been flung in a direction where these panels have been set to the largest spacing and have been able to generate a better, feasible, object function value.

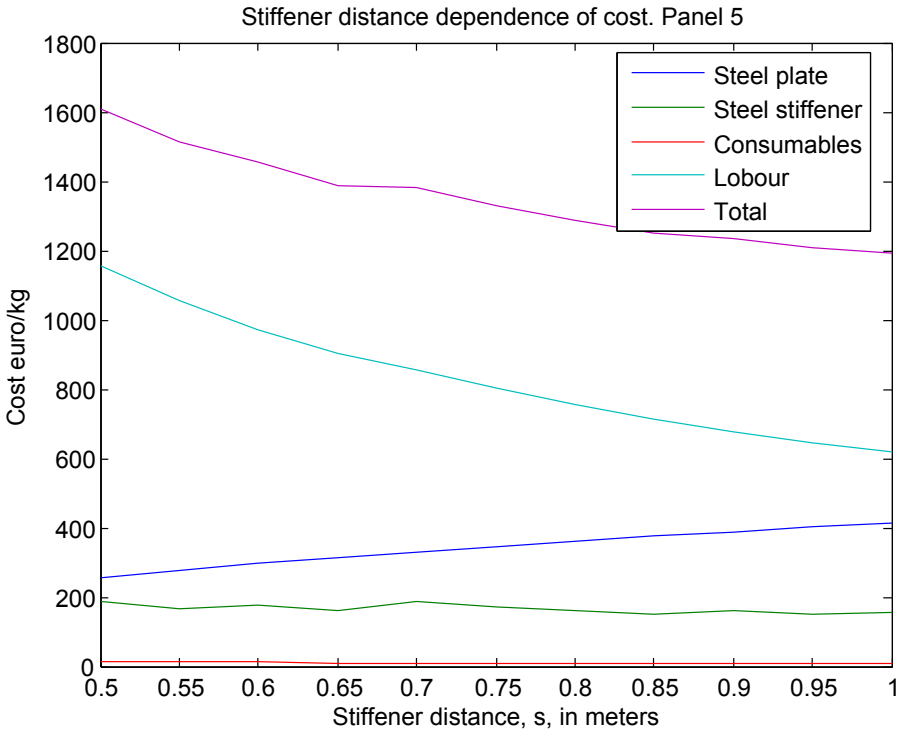


(a) Cost depending on s

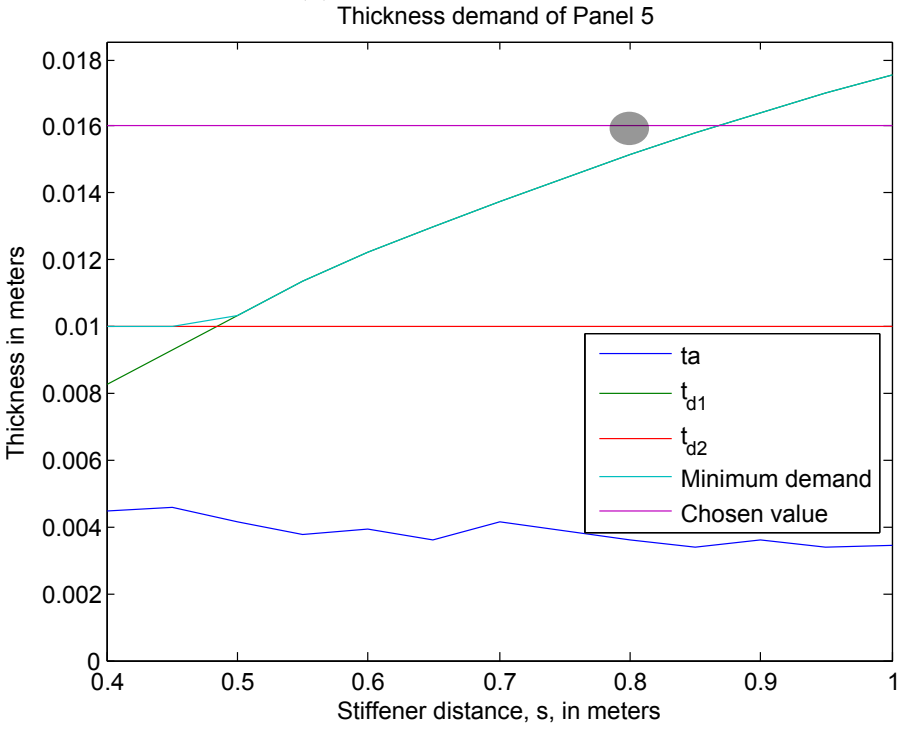


(b) Thickness depending on s

Figure 8.11: Weight and cost of panel 1, cost optimisation

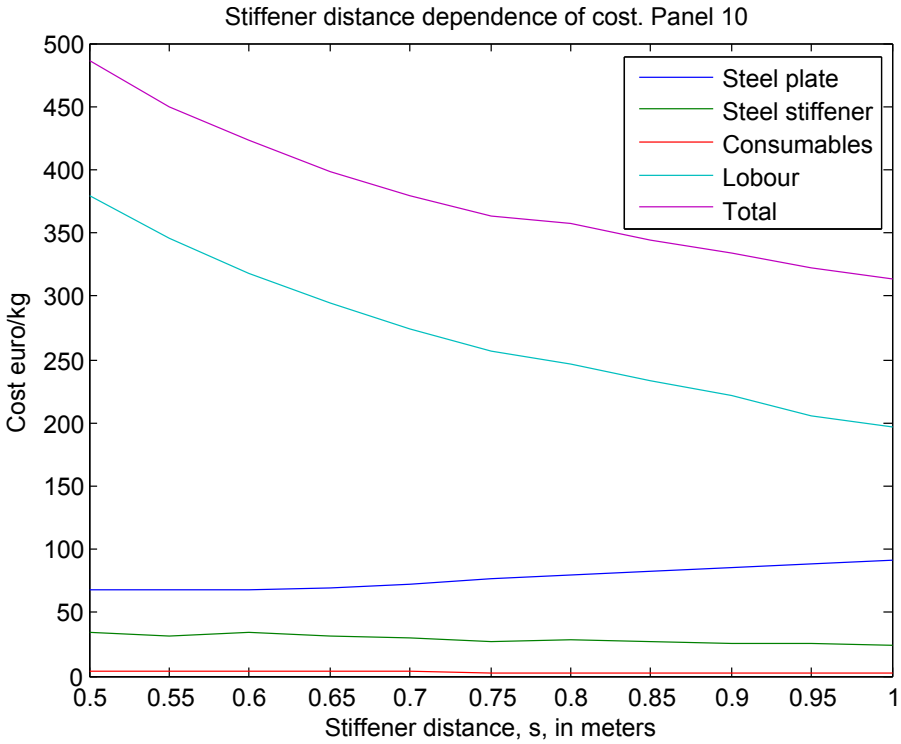


(a) Cost depending on s

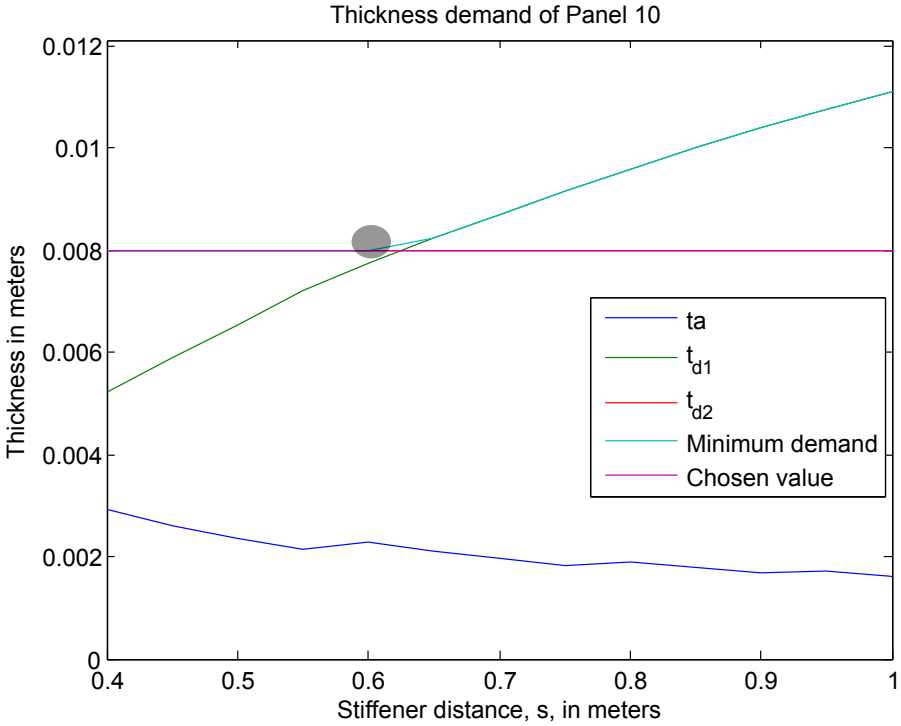


(b) Thickness depending on s

Figure 8.12: Weight and cost of panel 5, cost optimisation



(a) Cost depending on s



(b) Thickness depending on s

Figure 8.13: Weight and cost of panel 10, cost optimisation

Mixed optimisation

The optimisation done with mixed objectives look more like the initial scantlings than the other optimisations, as there is more variation in s, and a more even distribution of t. The main difference lays in that the initial scantlings tend to have a lot of 0.6 meter distance between the stiffeners in contrary to the mixed optimization, were most of the distances between the stiffeners are set to 1 meter. This may be because cost is more sensitive than the weight reduction and therefore affect the object function more, meaning that an improvement towards lower cost gives a larger gain in the object function than an improvement towards lower weight. Examples of the variables in this optimisation is found in panel 4, Figure(8.14), panel 6, Figure(8.15), and panel 11, Figure(8.16).

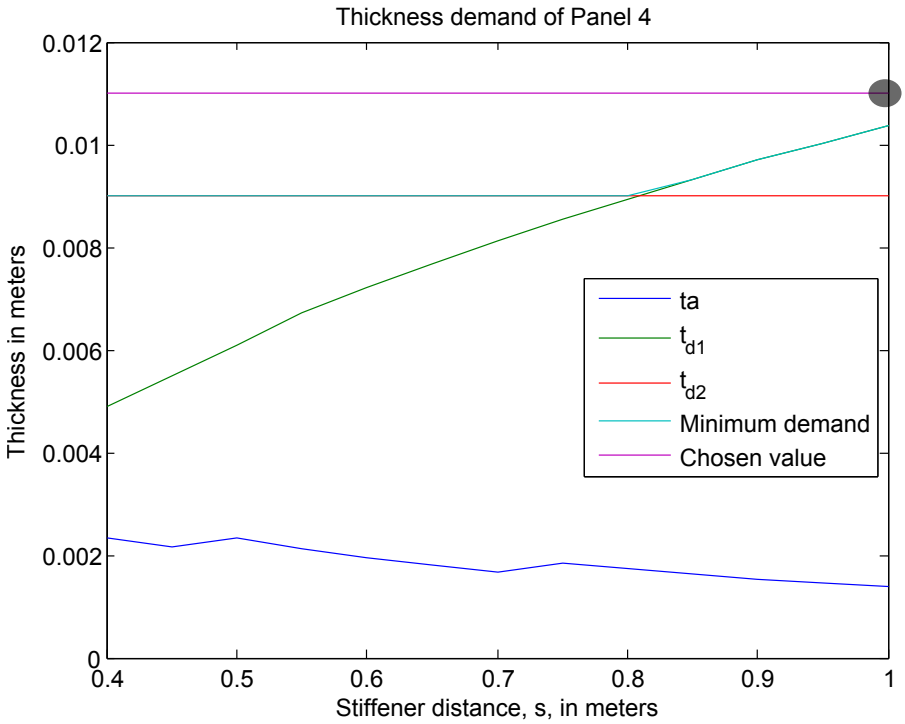


Figure 8.14: Thickness depending on s for panel 4, mixed optimisation

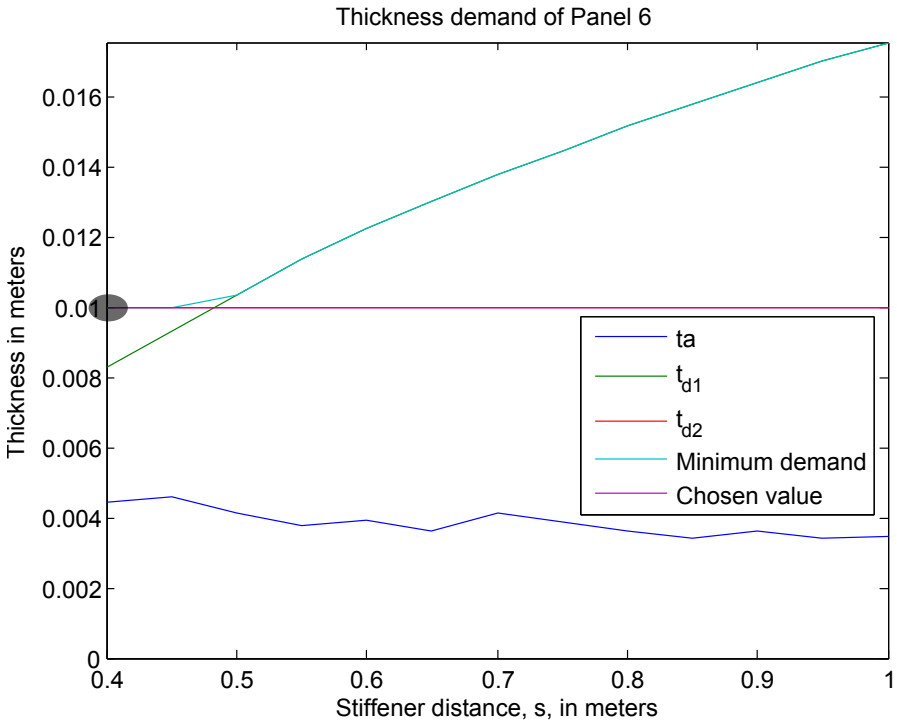


Figure 8.15: Thickness depending on s for panel 6, mixed optimisation

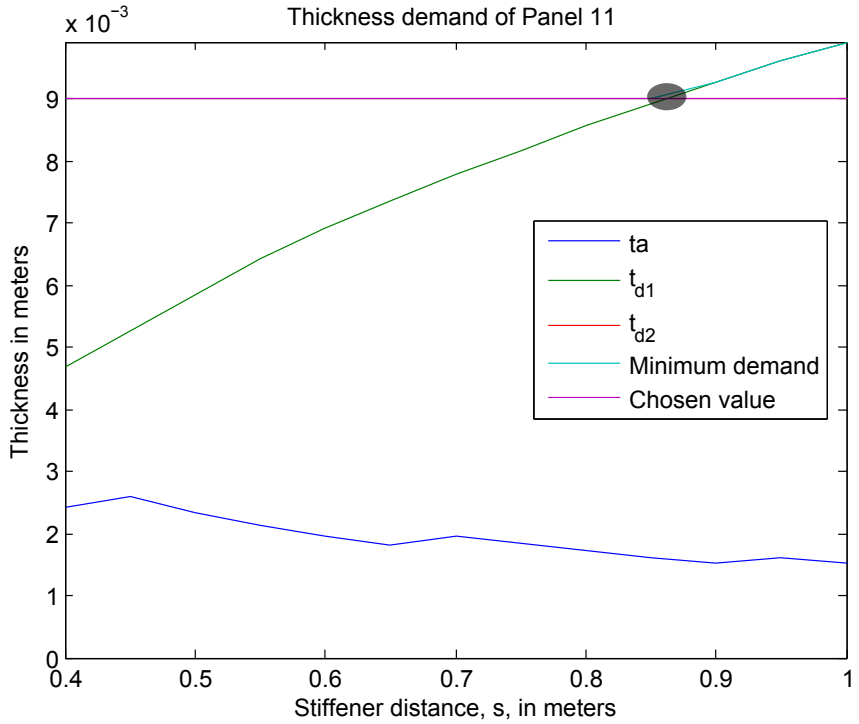


Figure 8.16: Thickness depending on s for panel 11, mixed optimisation

The buckling problem

In extra trials the maximum stiffener distance in the feasible area was increased,to test the effect in the cost and weight. The results in Table(8.9) shows that for weight optimisation, a lower cost could be achieved by increasing the stiffener distance further.

α	0	0	1	1
s_{max}	Weight	Cost	Weight	Cost
1 m	10662.7	36825.4	11787.4	25360.9
1.2 m	11529.0	38242.1	12673	24102.07
1.4 m	11397.8	33288.5	13154.2	23173.4
2.1 m	10804.4	27372.2	13439.3	22813.9

Table 8.9: Results for different stiffener distances

When the critical stress, dependent on the thickness ans stiffener spacing of a panel, become smaller than the actual stress in the panel, buckling will occur. because of this effect, high stiffener distances are avoided in ship building. The critical level of stress on a section is presented in the formula in Equation (8.1), found in [Amdahl, 2005]. To test for buckling Equation (8.2) is used with the assumption that the stress in the deck is 175 [N/mm^2][1]. Trying the formula on a plate in the deck with increasing stiffener distance, one can see in Table (8.10) that the critical stress becomes lower and lower, driving the usage factor over one,and into the buckling stress area. The example is relevant for panel in the top deck, because it is there the longitudinal stress is highest. To ensure that the connection between plate thickness and stiffener spacing is safe, one should include the buckling requirements found in Chapter 13 in the Rules for over 100 meter found in [DNV, 2013] into the program.

[1]Value found in the DNV rules for longitudinal strength

s	t[mm]	σ_c[N/mm^2]	η [ ]
0.4	8	298	0.59
1	16	190	0.92
1	14	146	1.19

Table 8.10: Critical section modulus

$$\sigma_c = \frac{\pi^2 \cdot E}{12(1-\nu^2)}(\frac{t}{b})^2 \cdot k \qquad \text{N}/mm^2 \tag{8.1}$$

σ_c = Critical compressive buckling stress
E = Young modulus, 2.06e10^5 N/mm^2
ν = Poisson's ratio, 0.3
t = Plate thickness in meters
b = Stiffener distance in meters
k = Buckling coefficient=4

$$\eta = \frac{\sigma_a}{\sigma_c} \tag{8.2}$$

η = usage factor, should be under 1
σ_a = Actual compressive stress
σ_c = Critical compressive buckling stress

(8.3)

Effect of different frame distances

As the frame distance must be the same for all the panels, the frame distance could not be set as a design variable. By using some time to experiment with different frame distances for the optimisation, an indication on how the frame distance change the weight and cost of the cross section in Table (8.11). The numbers were obtained by running 4 optimisations with α=0 and α=1 for each frame length, except l_s=1.4, where the numbers from the optimisation trial are used.

By increasing the frame distance, the transverse frames will need to become stronger because of the increased load area. This may lead to increased thickness in the

	Weight optimisation		Cost optimisation	
α	0	0	1	1
l_s	Weight	Cost	Weight	Cost
1.2 m	12189.7	32596.0	13125.7	27481.0
1.4 m	10662.7	36825.4	11787.4	25360.9
1.6 m	11096.0	38560.2	11905.0	26217.2
2.1 m	11071.8	41430.7	12197.4	31355.3

Table 8.11: Results for different frame distances

beam flanges, because the web thickness is set as a constant. Also the stiffeners will be affected as they need to be stronger when the load area increases.

Because of the restrictions on the stiffeners allowed, the stiffener distance must be decreased to keep the DNV section modulus demand for the stiffeners down. This cause the cost to go up, but the cost for frame distance on 1.6 meters is lower then when the distance is 1.2 meters. At frame distance 1.2 meter in the cost optimisation, the weight is high because of multiple high plate thicknesses in the best scantlings found in this run. At the cost optimisation for 2.1 meters frame distance, the stiffener distance and there by the weight, is pushed down because of the stiffener restrictions. It is interesting to see that the numbers indicate that the frame distance used in the case study, 1.4 meters, gives the lowest weight both for the weight and the cost optimisation. To find the ideal frame distance more optimisations should be run to verify the numbers and find the trend in the development of the cost and weight.

The program as a design tool

Running one optimisation while making the feasible population takes 15 minutes, on a NTNU school computer, for one specified α. This time includes the storing of the values and creating a cost chart. As there is a spread in the solutions found when running the optimisation, one would like to run several optimisations to find the best solution. By using a loop in the code the user will get the best of 10 optimisation runs in under two hours, using the feasible set in Appendix E.

The PSO optimisation has proved it can handle multiple variables in a short time. Looking at the scantlings produced in the different optimisations, the program do not always manage to reach the optimum value for the individual panels, but over all, it can be seen in Figure (8.3) that there is found many different combinations which has improved the cross section drastically. The optimisation knows the

contents of the best local and global particles, but this may not be enough when the particles are accelerated towards the best values. As the best values for the particle as a whole is remembered and not the specific value of the individual parameters.

As there is no buckling control included in the code, the designers need to test this after the scantlings are found. Also design refinement like the thickness transition between plates, and some analyses of the beam structure must evaluated

Chapter 9

Conclusion

By creating this optimisation routine, based on the DNV rules and the PSO algorithm, we get scantlings that differ from the design chosen variables of the initial cross section.

- The PSO optimisation for a object function give a range of feasible solutions, even though it is run from the same initial feasible population. Using several feasible populations gives a larger range in the results and helps avoid the optimisation getting stuck in a local optimum.
- One optimisation, counting creation the initial feasible population takes 15 minutes. Which is a relative short time to come up with a design alternative to continue working on.

Based on the limitations of the thesis we get the following results, which coincides with the assumption for low weight and low cost structure.

- The best weight was reduced with 10.7% by lowering the distance between stiffeners and the thickness on several plates. The optimisation also gave a reduction in cost because of reduced amount of steel used and therefore outweighed the extra labour involved involved.
- The best cost optimisation reduced the costs by 28.4% and the weight 7%. Here the labour and the cross section area was reduced.
- In combining the cost and weight optimisation, the weight was reduced by 10% and the cost by 23.9%.

Basing a conclusion on the the points above, a designer can, only by using some time defining the cross section and filling in the ship information, create a range of feasible combination of scantlings which differ from the regular path the designer

normally would take. The program has still things to be desired, but considering that it is a early design tool, and that by only using some hours one can get a range of different alternative solutions, the program is useful in the design stage.

Chapter 10

Further work

In the process of making the program, several thing where found that could improve the code and task, but was the work was deemed to be done in a later stage.

- Include the rules under 100 meters, making the program chose between what code the program should depend on
- Include the buckling rules in the constraints, to avoid critical stress on the structure.
- Make the program more user friendly by using pop up menu for creating the cross section and defining the user input.
- Create an optimisation where the type of stiffeners is added as a design variable for each panel.
- Add more experience based constraints, like a correlation for coherent panels to avoid large leaps in plate thickness.
- Re-evaluate the cost function with more experience based numbers to get a more correct cost optimisation.
- Implement the constraints for transverse bulkhead dimensioning to optimise a larger section of the cargo area at a time.
- Improve the calculation formula for the transverse beams.
- Look further into how the optimisation also can be used to find the optimum frame distance and make this a integrated or additional program in the optimisation.

- Make the optimisation able to adjust the tank sizes after structural and capacity demands.

Bibliography

J. Amdahl. *TMR 4205 Buckling and Ultimate Strength of Marine Structures.* NTNU, 2005.

J. Amdahl. *TMR 4167 MARIN TEKNIKK 2 del 1.* Norges teknisk-naturvitenskapelige universitet. Institutt for marin teknikk, 2009.

C. A. Carlsen and D. Kavlie. Indets - integrated design of tanker structures. *Norwegian Maritime Research*, 13(1):2–17, 1975.

DNV. *Rules for Classification of Ships - January 2013. Pt.3 Ch.1.* Det Norske Veritas AS, 2013.

K. V. Dokkum. Ship knowledge a modern encyclopedia. 2003.

S. Ehlers. A particle swarm algorithm-based optimization for high-strength steel structures. *Journal of Ship Production and Design*, 28(1):1–9, 2012.

S. Ehlers, H. Remes, A. Klanac, and H. Naar. A multi-objective optimisation-based structural design procedure for the concept stage - a chemical product tanker case study. *Ship Technology Research*, 57(3):182–197, 2010.

F. Hiller and G. Lieberman. *INTRODUCTION TO OPERATIONS RESEARCH.* McGraw-Hill International edition, 2010.

O. Hughes, M. F., and Z. V. A practical method for rational design of ship structures. *Journal of Ship Research*, 24(2):101–113, 1980.

J. Jalkanen. Particle swarm optimization of load carrying structures. *Journal of Structural Mechanics*, 2:23–35, 2006.

D. G. Karr, P. Rigo, S. S. Na, and R. Sarghiuta. Comparison of rational-based and rule-based optimum design of ship structures. *SNAME Transactions*, 110: 435–451, 2012.

J. Kennedy and R. Eberhart. Particle swarm optimisation.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings., IEEE International Conference onNeural Networks*, volume 6, pages 1942–1948. IEEE CONFERENCE PUBLICATIONS, December 1995.

B. J. Leira, K. Syvertsen, J. Amdahl, and C. M. Larsen. *TMR 4170 Marine konstruksjoner, grunnkurs*. Norges teknisk-naturvitenskapelige universitet. Institutt for marin teknikk, 2011.

C. Mürer. *DNV STRUCTURAL RULES. DEVELOPMENT BACKGROUND MOTIVES.* Det Norske Veritas, 1996.

K. Nerem. *Hovedoppgave: Dimensjonering og optimalisering av en seksjon i en tråler*. Inst. for marine konstruksjoner, Marinteknisk Senter, 1990.

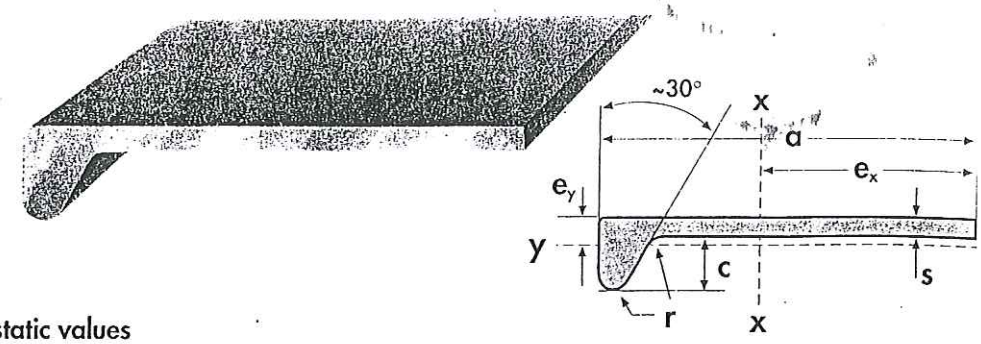
P. Rigo. Least-cost structural optimization oriented preliminary design. *Journal of Ship Production*, 17(4):202–215, 2001.

N. C. Thanh. *Optimum layout of stiffened panels subjected to lateral pressure and inplane loads*. Norges teknisk-naturvitenskapelige universitet. Institutt for marin teknikk, 2010.

Appendix A

HP profiles

Bulb Flats
ROLLED



Dimension range, weight/m and static values

Width a mm	Thickness s mm	Height c mm	Radius r mm	Area A cm²	Weight kg/m	e_x cm	I_x cm⁴	e_y cm	I_y cm⁴	W_x* cm³	Plate cross sectional area cm²
60	4	13	3.5	3.58	2.81	3.82	12.2	0.44	0.61	13	
	5	13	3.5	4.18	3.28	3.70	14.4	0.47	0.74	14	
	6	13	3.5	4.78	3.75	3.62	16.4	0.51	0.88	16	
80	5	14	4	5.40	4.24	4.89	33.8	0.47	1.02	23	
	6	14	4	6.20	4.87	4.78	39.0	0.50	1.20	25	
	7	14	4	7.00	5.50	4.69	43.3	0.54	1.42	27	
	Delivery by special agreement. Standard lengths 6–12 m										
100	6	15.5	4.5	7.74	6.08	5.98	76.1	0.51	1.71	38	
	7	15.5	4.5	8.74	6.86	5.87	85.3	0.55	1.99	41	
	8	15.5	4.5	9.74	7.65	5.78	94.3	0.59	2.31	45	
120	6	17	5	9.31	7.31	7.20	133	0.53	2.34	54	60
	7	17	5	10.5	8.25	7.07	148	0.56	2.70	59	
	8	17	5	11.7	9.19	6.96	164	0.60	3.10	63	
140	7	19	5.5	12.4	9.74	8.31	241	0.59	3.81	80	
	8	19	5.5	13.8	10.8	8.18	266	0.63	4.33	87	
	9	19	5.5	15.2	11.9	8.07	291	0.67	4.91	93	
160	7	22	6	14.6	11.4	9.66	373	0.65	5.84	110	
	8	22	6	16.2	12.7	9.49	411	0.68	6.54	118	
	9	22	6	17.8	14.0	9.36	448	0.71	7.30	126	
	12				15.2						
180	8	25	7	18.9	14.8	10.9	609	0.74	9.92	157	
	9	25	7	20.7	16.2	10.7	663	0.77	10.95	166	
	10	25	7	22.5	17.6	10.6	717	0.81	12.07	177	
	11				19.0						
200	9	28	8	23.6	18.5	12.1	941	0.84	15.75	225	
	10	28	8	25.6	20.1	11.9	1020	0.87	17.20	237	
	11.5	28	8	28.6	22.5	11.7	1126	0.92	19.62	255	
220	10	31	9	29.0	22.8	13.4	1400	0.93	23.85	302	
	11.5	31	9	32.3	25.4	13.1	1550	0.98	26.91	323	
240	10	34	10	32.4	25.4	14.7	1860	1.00	32.36	368	
	11	34	10	34.9	27.4	14.6	2000	1.03	34.83	391	
	12	34	10	37.3	29.3	14.4	2130	1.06	37.45	406	
260	10	37	11	36.1	28.3	16.2	2430	1.07	42.8	455	
	11	37	11	38.7	30.3	16.0	2610	1.10	45.9	474	
	12	37	11	41.3	32.4	15.8	2770	1.13	49.1	493	
280	11	40	12	42.6	33.5	17.4	3330	1.17	59.4	566	100
	12	40	12	45.5	35.7	17.2	3550	1.19	63.3	590	
300	11	43	13	46.7	36.7	18.9	4190	1.24	75.7	671	
	12	43	13	49.7	39.0	18.7	4460	1.26	80.4	701	
	13	43	13	52.8	41.5	18.5	4720	1.29	85.3	728	
320	12	46	14	54.2	42.5	20.1	5530	1.34	101	819	
	13	46	14	57.4	45.0	19.9	5850	1.36	107	849	
340	12	49	15	58.8	46.1	21.5	6760	1.41	125	947	
	14	49	15	65.5	51.5	21.1	7540	1.46	139	1014	
370	13	53.5	16.5	69.6	54.6	23.5	9470	1.55	177	1210	
	15	53.5	16.5	77.0	60.5	23.0	10490	1.59	195	1278	
400	14	58	18	81.4	63.9	25.5	12930	1.68	243	1580	
	16	58	18	89.4	70.2	25.0	14220	1.72	267	1666	
430	15	62.5	19.5	94.1	73.9	27.4	17260	1.81	328	1935	150
	17	62.5	19.5	103.0	80.6	26.9	18860	1.85	356	2036	

* Inclusive plate as noted

Standard lengths
6 – 18 m
220 – 430 mm profiles available up to 25 m lengths by special agreement.

Orders must include
measurements, a x s
lengths
quantity (minimum 3000 kg)

Appendix B

Excel sheet for stiffener area

Section modulus vs Area of stiffener

Section modulus with plate flange, stiffenere area without.

stiffener distance				0,5	m	l		1,5			
b				0,5		a		0,9			
plate thickness				0,006	m	a/b		1,08			
be				0,375		C>7		**0,75**			
Ae				0,00225							
A				0,003						steiners bidrag dominerande	
HP	h	A [m^2]	ex [m]	I_s	x	ΣxiAi [m^3]	ΣAi [m^2]	zn [m]	I_e	Z [m^3]	
120X8	0,12	0,00117	0,0696	1,64E-06	0,0756	0,000097452	0,00417	0,02337	5,36873E-06	5,23114E-05	
140X9	0,14	0,00152	0,0807	2,91E-06	0,0867	0,000140784	0,00452	0,031147	8,81788E-06	7,67753E-05	
160X8	0,16	0,00162	0,0949	4,11E-06	0,1009	0,000172458	0,00462	0,037329	1,25924E-05	9,78648E-05	
180X9	0,18	0,00207	0,107	6,63E-06	0,113	0,00024291	0,00507	0,047911	1,89735E-05	0,0001374	
200X9	0,2	0,00236	0,121	9,41E-06	0,127	0,00030872	0,00536	0,057597	2,62741E-05	0,000177046	
220X10	0,22	0,0029	0,134	0,000014	0,14	0,000415	0,0059	0,070339	3,67501E-05	0,00023609	

Plot data in [cm^2] and [cm^3]

HP	A	Z
t=10mm s=0,6m		
120X8	11,7	54,4033
140X9	15,2	
160X8	16,2	
180X9	20,7	146,2999
200X9	23,6	189,3153
220X10	29	254,1105

Appendix C

Example cost values

Modelling of the production cost - a basic cost module

The total production cost will be the sum of three components:

$F_C = F_{MAT} + F_{CONS} + F_{LAB}$ (1)

where

Symbol	Description	Unit
F_C	The total production cost	€
F_{MAT}	The cost of materials	€
F_{CONS}	The cost of consumables	€
F_{LAB}	The cost of labor	€

The Cost of Materials

The cost of materials means the steel acquisition cost. For a stiffened panel, this cost is directly derived from the structural weight using the following formula:

$$F_{MAT} = \gamma . L . B . \left[C_1 . \delta + C_2 . \frac{(h.d + w.t)_X}{\Delta_X} [1 + DW_2] + C_3 . \frac{(h.d + w.t)_Y}{\Delta_Y} [1 + DW_3] \right] \quad (2)$$

where

Symbol	Description	Unit
F_{MAT}	The cost of materials – for a stiffened panel	€
γ	Steel specific weight	N/m²
L	Stiffened panel length	M
B	Stiffened panel width	M
δ	Stiffened panel plate thickness	M
H	Web height	M
D	Web thickness	M
w	Flange width	M
t	Flange thickness	M
Δ_X	Longitudinal stiffeners spacing	M
Δ_Y	Transversal frames spacing	M
X	Index of longitudinal stiffeners	-
Y	Index of transversal frames	-
C_1	Cost / Kg of a plate with δ thickness	€ / Kg
C_2	Cost / Kg of longitudinal stiffeners	€ / Kg
C_3	Cost / Kg of transversal frames	€ / Kg
DW_2	Corrective factor of long. stiffeners weight	-
DW_3	Corrective factor of transversal frames weight	-

The values of the parameters C_1, C_2, C_3, should be calculated using the formulas:

$$\begin{aligned} C_1 &= C_1^o \left[1+\Delta C_1 \left(\delta - E_0\right)\ 10^3\right] \\ C_2 &= C_2^o \left[1+\Delta C_2 \left(d_X - E_{0X}\right)\ 10^3\right] \\ C_3 &= C_3^o \left[1+\Delta C_3 \left(d_Y - E_{0Y}\right)\ 10^3\right] \end{aligned} \qquad (3)$$

where

Symbol	Description	Unit
δ	stiffened panel plate thickness - actual	M
d_X	Long. stiffeners web thickness - actual	M
d_Y	Transversal frames web thickness - actual	M
E_0	Reference thickness for plate cost assessment	M
E_{0X}	Reference web thickness for long. stiffeners	M
E_{0Y}	Reference web thickness for transversal frames	M
C_1^0	Cost / Kg of a plate with E_0 thickness	€ / Kg
C_2^0	Cost / Kg of longitudinal stiffeners with E_{0X} web thickness	€ / Kg
C_3^0	Cost / Kg of transversal frames with E_{0Y} web thickness	€ / Kg
ΔC_1	Variation of C_1 per mm	1 / mm
ΔC_2	Variation of C_2 per mm	1 / mm
ΔC_3	Variation of C_3 per mm	1 / mm

The Cost of Consumables

The cost of consumables means the cost of welding except the labour cost and it is composed by the cost of energy, gas, electrodes, provision for equipment depreciation. The cost of consumables for a stiffened panel will be calculated as follows:

$$F_{CONS} = L \,.\, B \,.\, \left(\left[\frac{2-\alpha_X}{\Delta_X} \right] . C_{8X} + \left[\frac{2-\alpha_Y}{\Delta_Y} \right] . C_{8Y} \right) \qquad (4)$$

Symbol	Description	Unit
F_{CONS}	The cost of consumables – for a stiffened panel	€
L	Stiffened panel length	M
B	Stiffened panel width	M
Δ_X	Longitudinal stiffeners spacing	M
Δ_Y	Transversal frames spacing	M
α_X	Binary coeff. related to stiffeners manufacturing	-
α_Y	Binary coeff. related to frames manufacturing	-
C_{8X}	Cost / meter of the consumables related to long. stiffeners welding	€ / m
C_{8Y}	Cost / meter of the consumables related to transversal frames welding	€ / m

The values of the parameters C_{8X} and C_{8Y} should be calculated as follows:

$$C_{8X} = C^0_{8X} \ [1 + \Delta C_{8X} (d_X - E_{0X}) \ 10^3]$$

$$C_{8Y} = C^0_{8Y} \ [1 + \Delta C_{8Y} (d_Y - E_{0Y}) \ 10^3] \quad (5)$$

where

Symbol	Description	Unit
d_X	Long. stiffeners web thickness - actual	M
d_Y	Transversal frames web thickness - actual	M
E_{0X}	Reference web thickness for long. stiffeners	M
E_{0Y}	Reference web thickness for transversal frames	M
C_{8X}^0	Cost / meter of consumables for longitudinal stiffeners with E_{0X} web thickness	€ / m
C_{8Y}^0	Cost / meter of consumables for transversal frames with E_{0Y} web thickness	€ / m
ΔC_{8X}	Variation of C_{8X} per mm	1 / mm
ΔC_{8Y}	Variation of C_{8Y} per mm	1 / mm

The Labor Cost

The labor cost is related to the workload for welding and welding surface preparation. For a stiffed panel, the labor will be estimated as follows:

$$F_{LAB} = \eta \, . \, k \, . \, C_1^o \, . \, WLoad \quad (6)$$

Symbol	Description	Unit
F_{LAB}	The labor cost – for a stiffened panel	€
η	Efficiency parameter for the considered production plan	-
k	Plate weight equivalent to a man-hour of the considered shipyard	Kg / man-hour
C_1^0	Cost / Kg of a plate with E_0 thickness (see above – Cost of materials)	€ / Kg
WLoad	Workload required for the fabrication of the stiffened panel	Man-hour

The amount of workload should be calculated with the formula:

$$WLoad = L \,.\, B. \left[\begin{array}{l} \frac{1}{\Delta_X} . P_4 + \frac{1}{\Delta_Y} . P_5 \\ + \frac{1}{\Delta_X \,.\, \Delta_Y} \left(P_6 + \beta_X \,.\, \beta_Y \,.\, P_7 \right) \\ + \frac{1}{\Delta_X} . P_{9X} + \frac{1}{\Delta_Y} . P_{9Y} \\ + P_{10} \end{array} \right] \quad (7)$$

where

Symbol	Description	Unit
WLoad	Workload required for the fabrication of the stiffened panel	Man-hour
L	Stiffened panel length	M
B	Stiffened panel width	M
Δ_X	Longitudinal stiffeners spacing	M
Δ_Y	Transversal frames spacing	M
P_4	Workload per meter for the welding of long. stiffeners web on the plate (preparation included)	Man-hour / m
P_5	Workload per meter for the welding of transversal frames web on the plate (preparation included)	Man-hour / m
P_6	Workload required for the welding and preparation of one intersection between long. stiffeners and transversal frames	Man-hour / intersection
P_7	Workload required for fixing the brackets at one intersection between long. stiffeners and transversal frames	Man-hour / intersection
P_{9X}	Workload required to built 1 meter of long. stiffener – assembly of web - flange (preparation + welding)	Man-hour / m
P_{9Y}	Workload required to built 1 meter of transversal frame – assembly of web - flange (preparation + welding)	Man-hour / m
P_{10}	Workload required for the preparation of 1 m² of plate (cutting, positioning)	Man-hour / m²
β_X	Ratio between the amount of intersections requiring long. brackets and the total amount of intersections	-
β_Y	Ratio between the amount of intersections requiring transversal brackets and the total amount of intersections	-

The values of the unitary cost parameters involved in the equation (7) should be calculated as follows:

$$P_4 = P_4^o \left[1 + (d_X - E_{OX}).10^3 . \Delta P_4\right]$$

$$P_5 = P_5^o \left[1 + (d_Y - E_{OY}).10^3 . \Delta P_5\right]$$

$$P_{9X} = P_{9X}^o \left[1 + (d_X - E_{OX}).10^3 . \Delta P_{9X}\right]$$

(15)

$$P_{9Y} = P_{9Y}^o \left[1 + (d_Y - E_{OY}).10^3 . \Delta P_{9Y}\right]$$

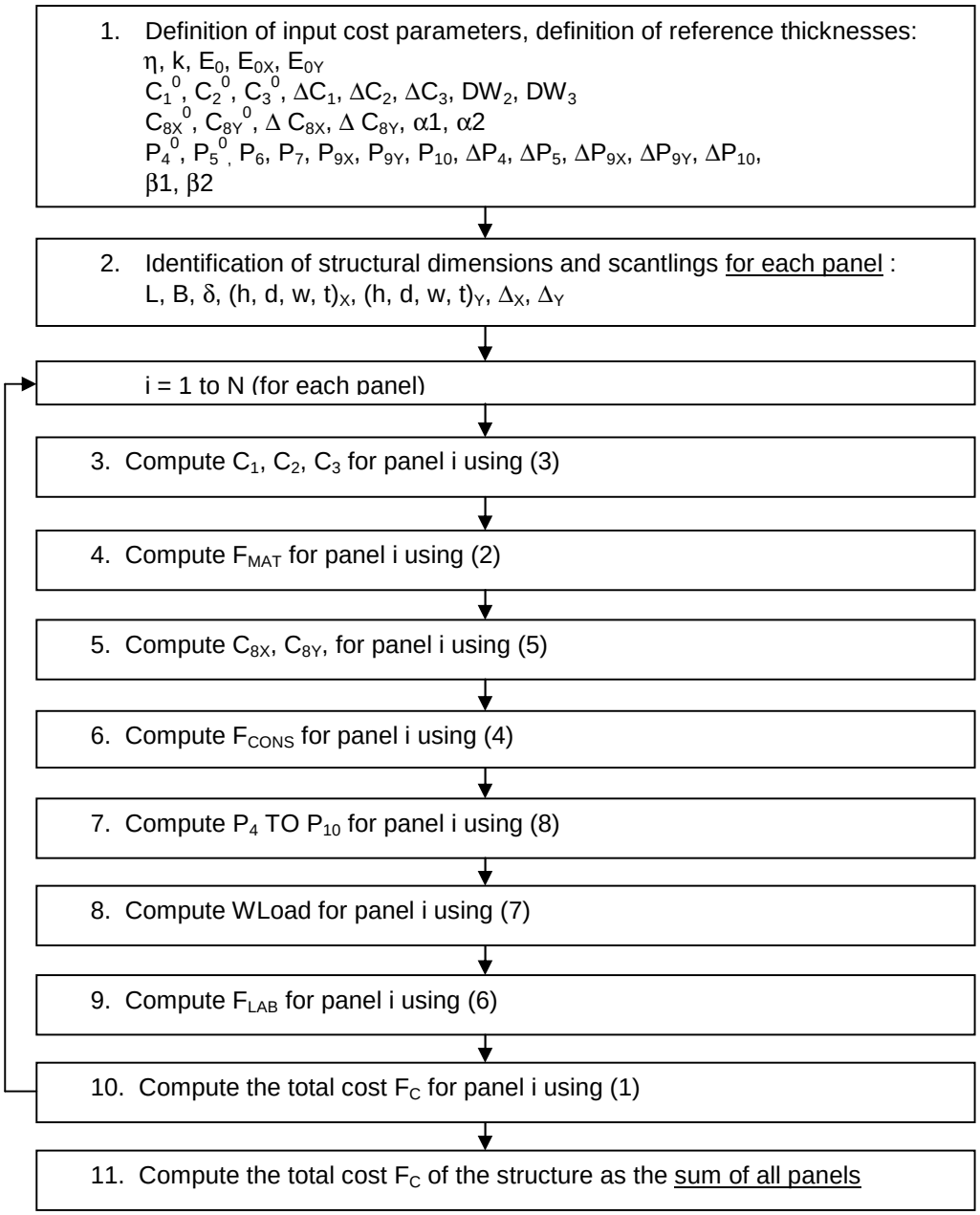
$$P_{10} = P_{10}^o \left[1 + (\delta - E_o).10^3 . \Delta P_{10}\right] \quad (8)$$

where

Symbol	Description	Unit
δ	stiffened panel plate thickness – actual	M
d_X	Long. stiffeners web thickness – actual	M
d_Y	Transversal frames web thickness – actual	M
E_0	Reference thickness for plate cost assessment	M
E_{0X}	Reference web thickness for long. Stiffeners	M
E_{0Y}	Reference web thickness for transversal frames	M
P_4^0	Workload per meter for the welding of long. stiffeners web (E_{0X} thickness) on the plate	Man-hour / m
P_5^0	Workload per meter for the welding of transversal frames web (E_{0Y} thickness) on the plate (preparation included)	Man-hour / m
P_{9X}^0	Workload required to built 1 meter of long. stiffener – assembly of web (E_{0X} thickness) - flange (preparation + welding)	Man-hour / m
P_{9Y}^0	Workload required to built 1 meter of transversal frame – assembly of web (E_{0Y} thickness) - flange (preparation + welding)	Man-hour / m
P_{10}^0	Workload required for the preparation of 1 m² of plate with E_0 thickness	Man-hour / m²
ΔP_4	Variation of P_4 per mm	1 / mm
ΔP_5	Variation of P_5 per mm	1 / mm
ΔP_{9X}	Variation of P_{9X} per mm	1 / mm
ΔP_{9Y}	Variation of P_{9Y} per mm	1 / mm
ΔP_{10}	Variation of P_{10} per mm	1 / mm

Step-wise description of production cost model

Before the start of the production cost calculation, the considered structure should be divided in several flat stiffened panels. Considering that the total number of stiffened panels is N, the calculation will follow the following steps:



Example values

For the Material cost

C_1	Cost / Kg of a plate with □ thickness	0.3500	€ / Kg
C_2	Cost / Kg of longitudinal stiffeners	0.4500	€ / Kg
C_3	Cost / Kg of transversal frames	0.4000	€ / Kg
DW_2	Corrective factor of long. stiffeners weight	0.0000	-
DW_3	Corrective factor of transversal frames weight	0.0000	-
ΔC_1	Variation of C_1 per mm	0.0000	1 / mm
ΔC_2	Variation of C_2 per mm	0.0000	1 / mm
ΔC_3	Variation of C_3 per mm	0.0000	1 / mm

For the labour cost

E_0	Reference thickness for plate cost assessment	0.0100	m
E_{0X}	Reference web thickness for long. Stiffeners	0.0100	m
E_{0Y}	Reference web thickness for transversal frames	0.0100	m
P_4^0	Workload per meter for the welding of long. stiffeners web (E_{0X} thickness) on the plate	0.5000	Man-hour / m
P_5^0	Workload per meter for the welding of transversal frames web (E_{0Y} thickness) on the plate (preparation included)	1.2000	Man-hour / m
P_{9X}^0	Workload required to built 1 meter of long. stiffener – assembly of web (E_{0X} thickness) - flange (preparation + welding)	0.0000	Man-hour / m
P_{9Y}^0	Workload required to built 1 meter of transversal frame – assembly of web (E_{0Y} thickness) - flange (preparation + welding)	0.0000	Man-hour / m
P_{10}^0	Workload required for the preparation of 1 m² of plate with E_0thickness	0.1000	Man-hour / m²
ΔP_4	Variation of P_4 per mm	0.0200	1 / mm
ΔP_5	Variation of P_5 per mm	0.0200	1 / mm
ΔP_{9X}	Variation of P_{9X} per mm	0.0000	1 / mm
ΔP_{9Y}	Variation of P_{9Y} per mm	0.0000	1 / mm
ΔP_{10}	Variation of P_{10} per mm	0.0400	1 / mm
P_6	Workload required for the welding and preparation of one intersection between long. stiffeners and transversal frames	0.2500	Man-hour / intersection
P_7	Workload required for fixing the brackets at one intersection between long. stiffeners and transversal frames	1.3000	Man-hour / intersection
β_X	Ratio between the amount of intersections requiring long. Brackets and the total amount of intersections	1.0000	-
β_Y	Ratio between the amount of intersections requiring transversal brackets and the total amount of intersections	1.0000	-

For the consumable cost

α_X	Binary coeff. related to stiffeners manufacturing	1.0000	-
α_Y	Binary coeff. related to frames manufacturing	0.0000	-
C_{8X}	Cost / meter of the consumables related to long. stiffeners welding	2.0000	€ / m
ΔC_{8X}	Variation of C_{8X} per mm	0.0500	1 / mm

```
Simplifciation to assess the cost per strake:

k=3500

Cost_per_strake=7000*t*L+2*HP*n*L+4500*t+2*L*n+6+.2*k*L*n+k*L;
```

For HP types:

1- 100x6
2- 120x8
3- 140x8
4- 160x8
5- 180x10
6- 200x10
7- 220x10
8- 240x10
9- 260x12
10- 280x12
11- 300x12
12- 320x13
13- 340x14
14- 370x13
15- 400x16
16- 430x15

Appendix D

Program chosen parameters

Panel type	σ $[N/mm^2]$	t_l [mm]	k [ ]	Comment
KEEL	120	7	0.05	
BOTTOM_OUT	120	5	0.04	
BOTTOM_IN_C	140	7	0.03	
BOTTOM_IN_GIRDER	130	6	0.04	Center girder
BOTTOM_IN_GIRDER	130	6	0.02	Out of center girders and floors
BOTTOM_IN	140	6	0.03	
SIDE_IN and SIDE_EXT	140	5	0.04	Below 4.6 m over waterline
SIDE_IN and SIDE_EXT	140	5	0.03	
SIDE_IN and SIDE_EXT	140	5	0.02	
SIDE_IN and SIDE_EXT	140	5	0.01	Above 11.5 from waterline
BULKHEAD_EXT	160	5	0.03	
DECK_ST	120	5.5	0.02	
DECK	120	5.5	0.00	

Table D.1: Panel parameters available

p

Appendix E

Feasible values for trial optimisation

1	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27				
1	0,4	0,45	0,5	0,55	0,6															
2	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
2	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
3	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
3	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
4	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
4	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
5	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		
5	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
6	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		
6	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
7	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
7	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
8	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27		
8	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
9	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
9	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
10	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
10	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
11	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
11	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
12	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
12	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
13	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	
13	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
14	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
14	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
15	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
15	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
16	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
16	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
17	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
17	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
18	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
18	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
19	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
19	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							
20	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
20	0,4	0,45	0,5	0,55	0,6	0,65	0,7	0,75	0,8	0,85	0,9	0,95	1							

Appendix F

MATLAB code

```
%Start file for running either direct or optimisation evaluation.
%Is excel available? yes=1, no=0
tic
%empty the files for old values
    %fidOUT = fopen('move_val.txt','w');
    %fclose(fidOUT);
    %User input:
    inn1='Userinput.xlsx'; %ship values
    inn2='Geoinputinit.xlsx'; %geometry
    yinit = 12401.4;
    Cinit =35276.5;
%excel availeble =1, if not = 0
excel=0;
%choose what the program thould do
user='dir'; %'dir' or 'opt'

%to avoid generating new fasible pop when nothing else is changed:
initfile='';
%initfile='initial_feasible_population_disc.txt'; %initial population file
%if one wantd to print the geometry fig=1, else fig=0
fig=0;

%save scantlings and values connected
saveval='Optimal_val_0_7j.txt';

%initial file or spesifyed run?
exin=1;
% Scantling input file exin=1
name='initscantlings.xlsx'; %t s As for all panels in a matrix
%Direct from Mat file exin=0
ip=15; %run number in matrix
a=load('Matnew_2feas.txt'); % alfa, t s for all panels in a long vector
[nnn nmm]=size(a);
Matdir=a(ip,2:nmm);
%%
%Input for the direct run and optimication info
[Ls, D, T, B, v, C_B, f_1, R_, alfa, rho_s,rho_c, H_air, mirror, l_s,...
    coo,Li, T0, K, T_C, paneldef,war] = inputvalues(inn1,inn2,excel);
if war==1
    break
end

L1=min(Ls,300);
n=length(coo(:,1)); %Number of panels (if the design varables change, so must this)
[Zg,C_w]=globalsectionmod(Ls,B,C_B,f_1,R_);
[P,sig,t_add] = platepressure(coo, Ls, T, B, v, rho_c, H_air,C_B, ...
                f_1, C_w, R_, T0, K,paneldef,D,mirror);
Pi=P(:,1);

alfa=0;
if  strcmp(user,'opt')==1     %Run the optimication.
    %Define where the input information is found og store constants
    fidOUT = fopen('move_val.txt','w');
    fprintf(fidOUT,'%s ',inn1);
    fprintf(fidOUT,'%s ',inn2);
```

```
     fprintf(fidOUT,'%d ',yinit);
     fprintf(fidOUT,'%d ',Cinit);
    fprintf(fidOUT,'% d ',excel);
    fprintf(fidOUT,'% d ',C_w);
    fprintf(fidOUT,'% d ',Zg);
    fprintf(fidOUT,'% d ',alfa);
    fclose('all');

    %Run once to get the initial feasible population. Adapt n_swarm to get
    %a sufficiently large initial population.
    var_n=2*n; %Number of variables
    if isempty(initfile)==1
    search_pop=600; % select size
    %Create feasible population
    [~]=StructOpt(var_n,Li,1,search_pop,[],t_add, L1, f_1,0);

    a=load('generation_dev.txt'); %feasible_population.txt
    [nm mm]=size(a); %nm=searc_pop*2

    X=zeros(var_n,var_n);
    fe_a=a((a(:,1)==1),:); %feasible
    if length(fe_a(:,1)) < var_n
     fprintf('Run once more with higher swarm size to get enough input values\n')
    else
    for i=1:var_n
        [~,tn]=min(fe_a(:,3));
        fe_a(tn,3)=inf;
        X(i,:)=fe_a(tn,4:mm);
    end

    fid = fopen('initial_feasible_population_disc.txt', 'w');
    for i=1:var_n
        fprintf(fid, '% 9.3f ', X(i,:));
        if i<var_n
        fprintf(fid,'\r\n');
        end
    end
    fclose(fid);
    %Run a second time to find the best combination
    [best_x]=StructOpt(var_n,Li,2,0,[],t_add, L1, f_1,1);
    end

    else
       [best_x]=StructOpt(var_n,Li,2,0,initfile,t_add, L1, f_1,1);
    end

else
%use direct values
    if excel==0
      %if excel not installed:
      [num,~,~] = xlsread(name,'','','basic');
    else
        %if excel installed:
       [num,~,~] = xlsread(name);
    end
```

```
    x=zeros(1,length(num)*2);
    Fai=zeros(1,length(num)*2);
    ji=1;
    for hi=1:length(num(:,1))
    x(1,ji:ji+1)=num(hi,1:2);
    Fai(1,hi)=num(hi,3);
    ji=ji+2;
    end
    if exin==1
    best_x=x;
    [Fa,HP,HPn]=profiles(Fai/100^2);
    t_hp=HP(HPn,3)/1000; %[m]
    h_hp=HP(HPn,2);
    else
        best_x=Matdir;
        k=1;
        for j=1:2:length(best_x)-1
        t(k)=best_x(j)/1000;    %[m]
        s(k)=best_x(j+1);       %[m]
        k=k+1;
        end
        [~,~, Fa,t_hp,h_hp,~]= stipar(s,Li,Pi,l_s,f_1);
    end
end

%Stiffener area
t=zeros(n,1);
s=zeros(n,1);
k=1;
for j=1:2:length(best_x)-1
    t(k)=best_x(j)/1000;    %[m]
    s(k)=best_x(j+1);       %[m]
    k=k+1;
end

if strcmp(user,'opt')==1
[~,~, Fa,t_hp,h_hp,~]= stipar(s,Li,Pi,l_s,f_1);
%[h_hp t_hp]
end

[Z,tp,Z_b, Z_dnv_b,Z_s, Z_dnv_s,S,t_w,w_h] = constrfun(s,t,coo,L1,...
                                Li,Pi,sig,f_1,l_s,mirror,t_add,paneldef,D,Fa);

tp2=(t_add(:,1)+t_add(:,2)*L1/sqrt(f_1));

[y]= weight(s,t, Li,rho_s,Fa,T_C,S,t_w,w_h,l_s)*mirror;
[C_s1, C_s2, C_w, C_l, Frame]= costsimple(s,t,T_C, rho_s,Li,Fa,S,t_w,l_s,w_h,t_hp);
C=sum(C_s1+C_s2+C_w+C_l+Frame)*mirror;
pros=[sum(C_s1+C_s2) sum(C_w) sum(C_l)]*100/sum(C_s1+C_s2+C_w+C_l);
cm=C/y; %euro/kg

 %Pie chart
 X=[C_s1 C_s2 C_w C_l Frame];
x=sum(X);
```

```
explode=zeros(size(x));
[c,offset]=max(x);
explode(offset)=1;
figure
h=pie(x,explode);
colormap summer

textObjs = findobj(h,'Type','text');
oldStr = get(textObjs,{'String'});
val = get(textObjs,{'Extent'});
oldExt = cat(1,val{:});
%Create the new strings, and set the text objects' String properties to the new↙
strings:

Names = {'Steel of plate: ';'Steel of stiffener: ';'Consumables: ';'Labour: ';'Frame:↙
';};
newStr = strcat(Names,oldStr);
set(textObjs,{'String'},newStr)
%Find the difference between the widths of the new and old text strings and change the↙
values of the Position properties:
val1 = get(textObjs, {'Extent'});
newExt = cat(1, val1{:});
offset = sign(oldExt(:,1)).*(newExt(:,3)-oldExt(:,3))/2;
pos = get(textObjs, {'Position'});
textPos =  cat(1, pos{:});
textPos(:,1) = textPos(:,1)+offset;
set(textObjs,{'Position'},num2cell(textPos,[3,2]))
%%
% %Plot input of geometri
if fig==1

    figure
    hold on
% title('Simple geometry')
for i=1:n

    plot([coo(i,1), coo(i,3)],[coo(i,2), coo(i,4)])

    if coo(i,1) == coo(i,3)
    text(coo(i,1)+0.1,coo(i,2)+(coo(i,4)-coo(i,2))/2,num2str(i))
    elseif coo(i,2) == coo(i,4)
    text(coo(i,1)+(coo(i,3)-coo(i,1))/2,coo(i,2)+0.2,num2str(i))
    else
     text(coo(i,1)+(coo(i,3)-coo(i,1))/2-0.1,coo(i,2)+...
         (coo(i,4)-coo(i,2))/2+0.1,num2str(i))
    end
end
 xaxis=max(max(coo(:,1),coo(:,3))+1);     % Axis dimentioning
 yaxis=max(max(coo(:,2),coo(:,4))+1);     % Axis dimentioning
 axis([0 xaxis -0.2 yaxis])               % Sarboard side of the ship
end

%Create a output file
 fidOUT = fopen(saveval,'w');
 fprintf(fidOUT,'%s\t % 9.1f \r\n','weigth',y);
```

```
 fprintf(fidOUT,'%s\t % 9.1f \r\n','cost', C);
 fprintf(fidOUT,'%s\t % 9.1f \r\n','euro/kg', cm);
 fprintf(fidOUT,' % s\t % s\t\t % s\t % s\t % s\t\t % s\t\t % s \r\n','PN','t','s','HP↙
t','HP h','tp2','tp1');
 k=1;
 for i=1:n
 fprintf(fidOUT,'% d\t %9.1f\t % 9.2f\t % d\t % d\t % 9.2f\t % 9.2f \r\n',i,best_x(k),↙
best_x(k+1),h_hp(i),t_hp(i)*1000,tp2(i),tp(i)*1000);
 k=k+2;
 end
 fclose('all');
toc
```

```
%File enables multiple optimisations
%Start file for running either direct or optimisation evaluation.
%Is excel available? yes=1, no=0
tic
    %User input:
    inn1='Userinput.xlsx'; %ship values
    inn2='Geoinput.xlsx'; %geometry
    yinit = 12401.4;
    Cinit =35276.5;
%excel availeble =1, if not = 0
excel=0;
%choose what the program thould do
%xxx
user='opt'; %
%to avoid generating new fasible pop when nothing else is changed:
initfile='';
%initfile='initial_feasible_population_disc.txt'; %initial population file
%if one wantd to print the geometry fig=1, else fig=0
fig=0;
%May need to adjust serch if numner of variables or feasible range is
%increased
search_pop=2000; % select size

%fine names for saving the best and all the calues
savename='Optimal_val_6j%d.txt';%best with nr. of run connected later
matrix='Matnew_6j.txt';          %Saves all runs
%%
%Input for the direct run and optimication info
[Ls, D, T, B, v, C_B, f_1, R_, alfa, rho_s,rho_c, H_air, mirror, l_s,...
    coo,Li, T0, K, T_C, paneldef,war] = inputvalues(inn1,inn2,excel);
if war==1
    break
end

L1=min(Ls,300);
n=length(coo(:,1)); %Number of panels (if the design varables change, so must this)
[Zg,C_w]=globalsectionmod(Ls,B,C_B,f_1,R_);
[P,sig,t_add] = platepressure(coo, Ls, T, B, v, rho_c, H_air,C_B, ...
                 f_1, C_w, R_, T0, K,paneldef,D,mirror);
Pi=P(:,1);
             var_n=2*n; %number of variables
             alfa=[0 0 0.5 0.5 1 1];
             nalfa=length(alfa);
             np=4;
             Mat=zeros(nalfa*np,var_n+1); %obs
             tt=1;
             for  jj=1:nalfa
    %Define where the input information is found og store constants
    fidOUT = fopen('move_val.txt','w');
    fprintf(fidOUT,'%s ',inn1);
    fprintf(fidOUT,'%s ',inn2);
     fprintf(fidOUT,'%d ',yinit);
     fprintf(fidOUT,'%d ',Cinit);
    fprintf(fidOUT,'% d ',excel);
    fprintf(fidOUT,'% d ',C_w);
```

```
    fprintf(fidOUT,'% d ',Zg);
    fprintf(fidOUT,'% d ',alfa(jj));
    fclose('all');

    %Run once to get the initial feasible population. Adapt n_swarm to get
    %a sufficiently large initial population.

    %Create feasible population
    [~]=StructOpt(var_n,Li,1,search_pop,[],t_add, L1, f_1,0);

    a=load('generation_dev.txt'); %feasible_population.txt
    [nm mm]=size(a); %nm=searc_pop*2

    X=zeros(var_n,var_n);
    fe_a=a((a(:,1)==1),:); %feasible
    if length(fe_a(:,1)) < var_n
     fprintf('Run once more with higher swarm size to get enough input values\n')
    else

    for i=1:var_n
        [~,tn]=min(fe_a(:,3));
        fe_a(tn,3)=inf;
        X(i,:)=fe_a(tn,4:mm);
    end

    fid = fopen('initial_feasible_population_disc.txt', 'w');
    for i=1:var_n
        fprintf(fid, '% 9.3f ', X(i,:));
        if i<var_n
        fprintf(fid,'\r\n');
        end
    end
    fclose(fid);
    %Run a second time to find the best combination
        for ii=1:np
        [best_x]=StructOpt(var_n,Li,2,0,[],t_add, L1, f_1,0);
        Mat(tt,1)=alfa(jj);
        Mat(tt,2:(var_n+1))=best_x;
        tt=tt+1;
        end
    end
            end

%Evaluatin of best values
[nm m]=size(Mat);
t=zeros(n,1);
s=zeros(n,1);

f1=zeros(nm,1);
f2=zeros(nm,1);
f=zeros(nm,1);
y=zeros(nm,1);
C=zeros(nm,1);
```

```
for i=1:nm
alfa=Mat(i,1);
x=Mat(i,2:m);
k=1;
for j=1:2:length(x)-1
    t(k)=x(j)/1000;     %[m]
    s(k)=x(j+1);        %[m]
    k=k+1;
end
[~,~, Fa,t_hp,h_hp,~]= stipar(s,Li,Pi,l_s,f_1);

 [Z,~,~,~,~, ~,S,t_w,w_h] = constrfun(s,t, coo,L1, Li,...
     Pi,sig,f_1,l_s,mirror,t_add,paneldef,D,Fa);

y(i)= weight(s,t, Li,rho_s,Fa,T_C,S,t_w,w_h,l_s)*mirror;
%Cost estimate

[C_s1, C_s2, C_w, C_l, Frame]= costsimple(s,t,T_C, rho_s,Li,Fa,S,t_w,l_s,w_h,t_hp);
C(i)=sum(C_s1+C_s2+C_w+C_l+Frame)*mirror;
cm(i)=C(i)/y(i);

f2(i)=(C(i)/Cinit);
f1(i)=(y(i)/yinit); % multiplyed by 2 becaus of symetric structure.
f(i)=f1(i)*(1-alfa)+f2(i)*alfa;
end

figure
gscatter(f1,f2,Mat(:,1),[],'ox*',[],'on','Weight','Cost')

t=1;
row=0;
Fb=zeros(nm/np,1);
for i=1:np:nm
    [Ci,I] = min(f(i:(i+np-1)));
    Fb(t)=I+row;
    t=t+1;
    row=row+np;
end

figure
gscatter(f1(Fb),f2(Fb),Mat(Fb,1),[],'ox*',[],'on','Weight','Cost')

for i=1:length(Fb)
    Makepie(Mat(Fb(i),:));
     x=Mat(Fb(i),2:m);
     k=1;
for j=1:2:length(x)-1
    t(k)=x(j)/1000;     %[m]
    s(k)=x(j+1);        %[m]
    k=k+1;
end
 file=sprintf(savename,Fb(i));
 fidOUT = fopen(file,'w');
 fprintf(fidOUT,'%s \t % 9.1f \r\n','weigth',y(Fb(i)));
 fprintf(fidOUT,'%s \t % 9.1f \r\n','cost', C(Fb(i)));
```

```
 fprintf(fidOUT,'%s \t % 9.1f \r\n','euro/kg', cm(Fb(i)));

 [~,~, Fa,t_hp,h_hp,~]= stipar(s,Li,Pi,l_s,f_1);
 [Z,tp,Z_b,Z_dnv_b,Z_s,Z_dnv_s, ~,~,~] = constrfun(s,t,coo,L1,...
                                Li,Pi,sig,f_1,l_s,mirror,t_add,paneldef,D,Fa);

tp2=(t_add(:,1)+t_add(:,2)*L1/sqrt(f_1));
 %all the variables
fprintf(fidOUT,'%s \t % s\t % s \r\n','Z_built', 'Z_side','Z_bottom');
fprintf(fidOUT,'% 9.1f \t % 9.1f \t % 9.1f \r\n',Z, Z_s,Z_b);

fprintf(fidOUT,' % s\t % s\t\t % s\t % s\t % s\t\t % s\t\t % s \r\n','PN','t','s','HP↙
h','HP t','tp2','tp1');
 for ii=1:n
 fprintf(fidOUT,'% d\t % d\t % 9.2f\t % d\t % d\t % 9.1f\t % 9.2f \r\n',ii,t(ii)*1000,s↙
(ii),h_hp(ii),t_hp(ii)*1000,tp2(ii),tp(ii)*1000);
 end
 fclose('all');
end

 %save everything
 fid = fopen(matrix,'w');
  aaa=length(Mat(:,1));
 for j = 1:(aaa)
     fprintf(fid,'% 9.3f ',Mat(j,:));
     fprintf(fid,'\r\n');
 end
 fclose(fid);
toc
```

```
function [Ls, D, T, B, v, C_B, f_1, R_, alfa, rho_s, rho_c, H_air, mirror,...
    l_s,coo,Li, T0, K, T_C, paneldef,war] = inputvalues(name, name2,excel)
%Read in the user defines geometry for the panels (only one side of the
%ship). Remember to double where its needed. Stores and distributes the
%ship parameters

%Retreave ship information
if excel==0
%If excel not installed:
[num1,~,~] = xlsread(name,'','','basic');
else
%if excel installed:
[num1,~,~] = xlsread(name);
end
Ls  =num1(1);
D   =num1(2);
T   =num1(3);
B   =num1(4);
v   =num1(5);
C_B =num1(6);
f_1 =num1(7);
R_   =num1(8);
alfa  =num1(9);
rho_s=num1(10);       %Steel dencity
rho_c=num1(11);       %Cargo dencity
H_air=num1(12);          %Height of air pipe above main deck
mirror=num1(13); %2 for mirrord geom, 1 for not xxx user adaption
l_s=num1(14);

% Geometry input
if excel==0
%if excel not installed:
[num2,txt2,~] = xlsread(name2,'','','basic');
else
%if excel installed:
[num2,txt2,~] = xlsread(name2);
end

coln2=length(txt2(:,1));
paneldef=txt2(2:coln2, 1);
% column 1-4 has local coordinats(y,z)of the panel defined by each row
coo=num2(:,1:4);
%paneltype=num2(:,5);
for i=1:length(txt2(1,:))
    if strcmp(txt2(1,i),'T0[mm]')==1
        T0=num2(:,i-1);
    elseif strcmp(txt2(1,i),'K[]')==1
        K=num2(:,i-1);
    elseif strcmp(txt2(1,i),'T_C[mm]')==1
        T_C=num2(:,i-1);
    end
end
```

```
%% Use input geometry
%Test that all the nodes are accunted for
[war]=geotest(coo);
%Record panel lengths
n=size(coo(:,1));          %Number of rows
Li=zeros([n 1]);         %Empty vector for panel lengths
for i=1:n

    if (coo(i,4)-coo(i,2)) ==0
        Li(i)= abs(coo(i,3)-coo(i,1));
    elseif (coo(i,3)-coo(i,1))==0
        Li(i)=abs((coo(i,4)-coo(i,2)));
    else
        Li(i)=sqrt((coo(i,4)-coo(i,2))^2+(coo(i,3)-coo(i,1))^2);
    end

end
  fclose('all');
```

```
function [war]= geotest(coo)
%Test that every panel is connected, if not, make instructions to make
%new nodes there
npanel=length(coo(:,1));
war=0;
hmax=0;
for i=1:npanel
hmax=max(hmax,max(coo(i,2),coo(i,4)));
end

for j=1:npanel
    for i=1:npanel
    if (coo(j,1)==coo(i,1) && coo(j,2) ==coo(i,2) && j~=i) ||...
             (coo(j,1)==coo(i,3) && coo(j,2) ==coo(i,4)) || ...
             coo(j,1)==0
        test=0; %fins a match for panel j
        break
    else
        if max(coo(j,2),coo(j,4))<hmax
        test=1;
        else
            test=0;
        end
    end
    end
        if test==1
        fprintf('Missing node connection in panel %d \n', j);
        war=1;
        end
end
end
```

```
function [Zg, C_w]=globalsectionmod(Ls,B, C_B, f_1,R_)
%Calculate DNV's demand for the section modulus. DNV Rules for Ships over
%100 meter, January 2013. Pt.3 Ch.1 Sec.5

P_add=1.2; %Additional up to 40%
%Rules for Ships, Pt.3 Ch.1 Sec.4 B200
%Wave coefficient dependent on ship length
if Ls<=100
    C_w=0.0792*Ls;
    elseif 100<Ls<300
    C_w=10.75-((300-Ls)/100)^(3/2);
    elseif 300<=Ls<=350
    C_w=10.75;
    else
    C_w=10.75-((Ls-350)/150)^(3/2);
end

%Rules for Ships, Pt.3 Ch.1 Sec.5 B106 & B200
%Still water and wave bending monents, dependent on ship parameters [kNm]
C_wu=C_w*(1-R_/10);
 Ms_s= (-0.065* C_wu* Ls^2* B*(C_B+0.7))*P_add;
 Ms_h= (C_wu*Ls^2*B*(0.1225-0.015*C_B)); %xxx nor restricted

 C_B=max(C_B,0.6);
 Mw_s= (-0.11* C_w* Ls^2* B*(C_B+0.7))*P_add;
 Mw_h= ( 0.19* C_w* Ls^2* B* C_B);

%Rules for Ships, Pt.3 Ch.1 Sec.5 C304
%Global section modulus demand.

Zg=(max(abs(Ms_s+Mw_s),abs(Ms_h+Mw_h))*10^3/(175*f_1)); %[cm^3]
```

```
function [p, sig, t_a] = platepressure(coo, L,T, B,v, rho_ch, ...
                                       H_air,C_b, f_1, C_w, R_, T0, K, ...
                                       paneldef, D,~)
% This program calculates the pressure connected to the panels
%defined in the geometry matrix coo and other user input. Not all pressures
%has been concidered.  P=kN/m^2. DNV Rules for Ships over 100 meter,
%January 2013.

%INPUT;
%some innitial calculations sec.4
g_0=9.81;           %Constant of gravity    [m/s^2]
rho=1.025;          %Dencity of sea water   [t/m^3]
R=R_/10;            %Reduction parameter  [-]
L1=min(L,300);
%Create empty pressure vector
paneln=length(coo(:,1));%Number of panels
p=zeros(paneln,2);      %Dim. pressure and explanation
sig=zeros(paneln,1);    %Dim. plate tension

%thickness coefficients, some given as constants by uset input, others
%depend on position
t_a=zeros(paneln,2);

%% Initial calculations needed later
%Height at top corner, y-coordinat.
    H_ship=D;
%Air pipe heigth over baceline [m]
h_air=H_air+H_ship;
%Section 4 Design loads
%B203 Common acceleration parameter
C_v=min(0.2,sqrt(L)/50);
C_v1=max(0.8, v/sqrt(L));
%Not affected by restricted service
a0=3*C_w/L+C_v*C_v1;
%B600 Combined vertical acceleration. Hold kv constant as a simplification
%interrested in.
kv=0.7;                         %Constant between 0.3-0.6 from A.P
av=kv*g_0*a0/C_b;
%C200 Sea pressure calculations
%Constant between 0.2L and 0.7L from AP
ks=2;
%The smallest of T and the vertical distance from waterline to top of ship side,max 0.8↙
*Cw
kf=min(T,min(0.8*C_w,H_ship-T));
% Parameter for calculating p_dp
if v/sqrt(L)<=1.5
    p_l=ks*C_w+kf;
else
    p_l=(ks*C_w+kf)*(0.8+0.15*(v/sqrt(L)));
end
%Some simplifications of parameters. Descriprion found under the different
%Design load tables.
p_0=25;     % 25 in general, 15 in ballast for dry cargo vessels.
delta_p=25;  % 25 for ballast tanks, 0 for others tanks. (Page 58)
```

```
%% Pressure calculation
%Go through the panels and find the highest pressure for each panel
test=zeros(paneln,3);

for i=1:paneln
    %initial values for each panel - Assume load point at lowest point
    %Dynamic pressure
    p_dp=dynpressure(coo(i,:),B,T,p_l);
    %Vertical distance from waterline at draught T to load point
    h_0=abs(T-min(coo(i,2),coo(i,4)));
    %Distance from load point to top of tank and tank breadth, works for
    %horizontal beams
    [h_s,~]=tankheight(coo(i,1), coo(i,2),coo(i,3), coo(i,4), coo);
    %Distance from load point to top of air pipe
    h_p=h_air-coo(i,2);
    %Section 6 Bottom Structure
    if strcmp(paneldef(i),'KEEL')==1 || strcmp(paneldef(i),'BOTTOM_OUT')==1
        %Sea pressure P1 (sec 6, Table B1)
        p1=(10*h_0+p_dp)*(1-R);
        %Net pressure in way of cargo single bottom pressure? XXX
        %tank or deep tank
        %p2=rho*(g_0+0.5*av)*h_s-10*(0.35*T);
        %p3=rho*g_0*h_s+p_0-10*(0.35*T);
        p(i,1)=p1;
        sig(i)=120*f_1;  %When longitudenal frames constant

        if strcmp(paneldef(i),'KEEL')==1
        t_a(i,:)=[7 0.05];              %Keel parameters - konstant
        p(i,2)=1;
        else
        t_a(i,:)=[5 0.04];              %Outer bottom  - konstant
        p(i,2)=2;
        end

    %Inner bottom plating C400 (p4-p15)
    elseif strncmp(paneldef(i),'BOTTOM_IN',9)==1
        %inner bottom, floors and keel, horisontal elements

        %dry cargo in cargo holds: p4
        %Dependent on cargo heigth H_c
        if strcmp(paneldef(i),'BOTTOM_IN_C')==1
        p4=rho_ch*(g_0+0.5*av);
        p(i,1)=p4;
        p(i,2)=3;
        sig(i)=140*f_1; %when longditudenal frames within 0.4L
        t_a(i,:)=[7 0.03]; %Koefficient constant XXX finf fitting t_0

        elseif strcmp(paneldef(i),'BOTTOM_IN_G')==1%BOTTOM_IN_G
    %Plating in double bottom floors and longitudenal girder C500 (p13-p15)
    %Worst case: take the pressure at the bottom.
        h_s=coo(i,4)-coo(i,2); % tankheigth not good with vertical panels.
        p13=0.67*(10*h_p+delta_p);
        p14=rho*g_0*h_s+p_0;
        p(i,1)=max(p13,p14);
        sig(i)=130*f_1; %Longit. stiffened longitudenal girder within 0.4L
```

```
        if coo(i,1)==0 && coo(i,3)==0
        t_a(i,:)=[6 0.04]; %Max value: for center girder up to 2m above keel
        p(i,2)=4;           % Plating in double bottom, center girder
        else
        t_a(i,:)=[6 0.02];
        p(i,2)=5;           % Plating in double bottom, out of center
        end

        else %Bottom in sides- probably ballast
            p5=(10+0.5*av)*h_s;
            p7=0.67*(10*h_p+delta_p);
            p8=10*h_s+p_0;
            p(i,1)=max([p5 p7 p8]);
            sig(i)=140*f_1;
            p(i,2)=6; %inner bottom beside cargo
            t_a(i,:)=[6 0.03];      %otherwhere in hold if cieling is not fitted? xxx
        end
    %Section 7 Side Structures
    %Side plating, general (p1-p8)
    elseif strncmp(paneldef(i),'SIDE',4)==1
                if coo(i,2)<(T+4.6)           % Below summer waterline +4.6
                    t_a(i,:)=[5 0.04];
                elseif  (T+4.6+2.3)<coo(i,2)<=(T+4.6+2*2.3)
                    t_a(i,:)=[5 0.03];
                elseif (T+4.6+2*2.3)<coo(i,2)<=(T+4.6+3*2.3)
                    t_a(i,:)=[5 0.02];
                else
                    t_a(i,:)=[5 0.01];
                end
        sig(i)=140*f_1; %For longitudenal stiffened side plating (simpification) xxx
        %External
        if strcmp(paneldef(i),'SIDE_EXT')==1
            %Reduction because of service restruction
        p1=10*h_0+p_dp*(1-R);
        p2=max((p_dp-(4+0.2*ks)*h_0)*(1-R),6.250+0.025*L1);

            if coo(i,2)<T % Below summer waterline
                p(i,1)=p1;
                p(i,2)=4; %External side under water
            else
                p(i,1)=p2;
                p(i,2)=5; %External side over water
            end
        %internal
        else
            if coo(i,2)<T % ballast water line = T
            h_b=0.35*T-coo(i,2); %Distance from the load point to the minimum design↙
draugth. page 103
            else
                h_b=0;
            end
        p3=rho*(g_0+0.5*av)*h_s-10*h_b+p_0;
        p4=rho*g_0*h_s-10*h_b+p_0;
        p5=0.067*(rho*g_0*h_p+delta_p)-10*h_b;
```

```
        p(i,1)=max([p3 p4 p5]);
        p(i,2)=6; %Internal side, cargo depencence
        end

     %section 8 Deck structure
     %Strength deck plating (p1-p13)
    elseif strncmp(paneldef(i),'DECK',4)==1

        if strcmp(paneldef(i),'DECK_ST')==1 %STRENGTH DECK
        h_0=H_ship-T;
        p1=max(0.8*(p_dp-(4+.2*ks)*h_0)*(1-R),5);

             p(i,1)=p1;
             p(i,2)=7; %Strength deck, sea pressure
             sig(i)=120*f_1; %longitudenally stiffened
             t_a(i,:)=[5.5 0.02]; %Pick the max values

        else %Deck top or bottom
            p6=rho*(g_0+0.5*av)*h_s;
            p7=0.67*(rho*g_0*h_p+delta_p);
            p8=rho*g_0*h_s+p_0;

            p(i,1)=max([p6 p7 p8]);
            sig(i)=120*f_1; %longitudenally stiffened
            p(i,2)=12;
            t_a(i,:)=[5.5 0.00]; %Pick the max values
        end
    elseif strcmp(paneldef(i),'BULKHEAD')==1
            p5=rho*g_0*h_s+p_0;
            p3=rho*(g_0+0.5*av)*h_s;
            p4=0.67*(rho*g_0*h_p+delta_p);

            p(i,1)=max([p5 p3 p4]);
            sig(i)=160*f_1; %longitudenally stiffened
            p(i,2)=13;
            t_a(i,:)=[5 0.03]; %Longitudenal bulkheads

    else
        p(i,1)=0;
    end

    %If T0 and K given as input, these values override the program values:
    if T0~=0
        t_a(i,1)=T0;
    end

    if  K~=0
        t_a(i,2)=K;
    end

   %test(i,:)=[h_s h_p h_0];
end
end
```

```
function [p_dp] = dynpressure(coo_i,B,T,p_l)
%Sea pressure. DNV Rules for Ships, January 2013, Pt. Ch.1 Sec.4

%load point in bottom left corner
        %Horizontal distance from centerline.
        y=max(min(coo_i(3),coo_i(1)),B/4);
        %Vertival distance from baceline to load point
        z=min(min(coo_i(4),coo_i(2)),T);

        p_dp=p_l+135*(y/(B+75))-1.2*(T-z);

end
```

```
function [H_tank,B_tank]=tankheight(x1, y1, x2, y2, coo)
%Gives hight and breadt of tank it the bottom pannel is given. Finds the
%breadt of tween deck tank= higth of girder web but does not give heigth of
%inner side panels
npanel=length(coo(:,1));
panelnum1=zeros(1,npanel);
panelnum2=zeros(1,npanel);

inith=y1; % bottom corner of tank
initx=x1; % bottom corner of tank

H_tank=0.001;
B_tank=0; % if first coordinade not located in corner

while H_tank==0.001

tp=0;
tp2=0;

%Distinguish between panels that end and start in the end node of the
%choosen panel
for i=1:npanel
    %panels that starts in the panels end
    if x2==coo(i,1) & y2==coo(i,2)
        tp=tp+1;
        panelnum1(1,tp)=i;
       elseif (x2==coo(i,3) & y2==coo(i,4)) & (x1~=coo(i,1) || y1~=coo(i,2))
                tp2=tp2+1;
        panelnum2(1,tp2)=i;
    else
        %end of panel
        H_tank=y2-inith;
        B_tank=x2-initx;
    end
end

    panelnum1(panelnum1==0)=[]; %remove the unused columns
    panelnum2(panelnum2==0)=[]; %remove the unused columns

    if tp>= 1 && tp2==0 %panel up or to the next %
    %Multiple panels with same start value - pick the one with highest end
    %node, if only one, it switches to that one.

   [~, num]=max(coo(panelnum1,4)); %Find position of highest y value
   panelnum=panelnum1(num);
   x1=coo(panelnum,1); %Move start point
   y1=coo(panelnum,2); %Moce start point
   x2=coo(panelnum,3); %Move start point
   y2=coo(panelnum,4); %Moce start point

    elseif tp==0 && tp2==1  %Ship corner
              if y1<coo(panelnum2,4) %panel is vertical
                  H_tank=y2-inith;
                  B_tank=x2-initx;
                    if B_tank==0
```

```
                        x1=coo(panelnum2,1); %Move start point
                        y1=coo(panelnum2,2); %Moce start point
                        x2=coo(panelnum2,3); %Move start point
                        y2=coo(panelnum2,4); %Move start point

                    B_tank=initx-x1;
                    end
                else  %panel is horicontal
                    H_tank=0;
                    B_tank=x2-initx;
                end

    elseif tp>=1 && tp2==1  %Side or top position.
                            %Node where several panels meet.

        [~, num]=max(coo(panelnum1,4)); %Find position of highest y value
        panelnum1=panelnum1(num);

        %if the end panel is not horicontal, go up or to the best start panel
        if coo(panelnum2,2)~=coo(panelnum2,4)
            x1=coo(panelnum1,1); %Move start point
            y1=coo(panelnum1,2); %Moce start point
            x2=coo(panelnum1,3); %Move start point
            y2=coo(panelnum1,4); %Move start point
        else
            x1=coo(panelnum2,1); %Move start point
            y1=coo(panelnum2,2); %Moce start point
            x2=coo(panelnum2,3); %Move start point
            y2=coo(panelnum2,4); %Move start point
        H_tank=y2-inith;
        B_tank=x2-initx;
            if B_tank==0
            B_tank=initx-x1;
            end
        end

    else
      %nothing
    end
end
end
```

```
function [best_x]=StructOpt(n,Li,source,n_swarm,initfile, t_add, L1, f_1,pri)
%Created by Sören Ehlers

 %feasible values needed in the next run. Minimum the amount of variables.
swarm_size=n; %

%empty the files for old values
    fidOUT = fopen('generation_dev.txt','w');
    fclose(fidOUT);

% How to get initial population?
% 1 = random or 2 = read from file.
% PSO parameters.

 if source==1
  Algorithm = [n_swarm -1 1 1.4 0.8 3 2 pri];
 else
 Algorithm = [swarm_size -1 90 1.4 0.8 3 2 pri];
 end

% Upper and lower bound for continous variables tykkelse, stiver avstand
LowerB = [];
UpperB = [];

% PSO-parameter
% swarm_size = 5;         %minst så stor som variable mengden
% feasibles_initial_population = -1;              % Feasible designs in intial↙
population (-1 dont want to use).
% generations = 10;                               % Calculation rounds.
% inertia = 1.4;                                  % Intertia at start.
% beta = 0.8;                                     % Factor for dynamic inertia reduction
% beta_k = 3;                                     % Number of rounds when it should↙
improve, otherwise make inertia smaller
% penalty_factor = 2;                             % Penalty factor for violated↙
constraints
% print_results = 1;                              % 0 not printed, 1 is
% printed (results)


% Number of continous variables
    continous_variables = 0;

% Set of discrete variables
    nvariables=n;
    Feasible_Set=inf(nvariables,31);

    tn=1;
    for i=1:2:n
        tmin=max(ceil(t_add(tn,1)+t_add(tn,2)*L1/sqrt(f_1)),6);
        %tmin=7;
        tmax=27;
        tl=tmax-tmin+1;
    Feasible_Set(i,1:tl)=(tmin:1:tmax);

     Lf=round(Li(tn)*10)/10; %length of panel rounded.
```

```
     nl=ceil((min(1,Lf)-0.4)/0.05+1); %number of steps
     Feasible_Set(i+1,1:nl)=(0.4:0.05:min(1,Lf)); %steps of 0.5 m
     tn=tn+1;
    end

% Carry out the optimization.
if source == 1
    [best_f,best_x,best_g,history_f,history_x,history_g,iterations,...
        particle,particle_history,t_history] = PSO('particle_fun',...
        continous_variables,Feasible_Set,LowerB,UpperB,Algorithm,[]);
    %Get enough feasible values and store them in another file

elseif source == 2
    if isempty(initfile)==1
    [best_f,best_x,best_g,history_f,history_x,history_g,iterations,...
        particle,particle_history,t_history] = PSO('particle_fun',....
        continous_variables,Feasible_Set,LowerB,UpperB,Algorithm,...
        'initial_feasible_population_disc.txt');
    else
        [best_f,best_x,best_g,history_f,history_x,history_g,iterations,...
        particle,particle_history,t_history] = PSO('particle_fun',....
        continous_variables,Feasible_Set,LowerB,UpperB,Algorithm,...
        initfile);
    end
end


%Best feasible particletion
format compact;
disp(' ')
disp(' ')
disp('Results:')
best_f,best_x,best_g
disp(' ')
disp(' ')

end
```

```
function [best_f,best_x,best_g,history_f,history_x,history_g,iterations,...
          particle,particle_history,t_history] = PSO(fun,counter_number,Feasible_Set,↙
LowerB,...
                                                    UpperB,Algorithm,Init_Popu)
%Created by Sören Ehlers
% PSO-parameter.
swarm_size = Algorithm(1);
feasibles_initial_population = Algorithm(2);
generations = Algorithm(3);
inertia = Algorithm(4);
beta = Algorithm(5);
beta_k = Algorithm(6);
penalty_factor = Algorithm(7);
print_results = Algorithm(8);
c1 = 2;
c2 = 2;

% CPU-time at beginning.
t0 = cputime;

% Number of analysis.
particle = 0;

% No of discrete variables
if isempty(Feasible_Set)==1
    discrete_var_number = 0;
else
    discrete_var_number = length(Feasible_Set(:,1));
end

% Generate automatically the initial population randomly so that it
% contains desired amount of feasible particletion
if isempty(Init_Popu)==1
    [position,objective_function,constraint,particle,history_best_f,best_individual,...
        history_f,best_current_individual,history_g] = ...
        new_population(fun,swarm_size,(counter_number+discrete_var_number),↙
feasibles_initial_population,...
        Feasible_Set,penalty_factor,counter_number,LowerB,UpperB);

% Read init pop from file.
else
    [position,objective_function,constraint,particle,history_best_f,best_individual,...
        history_f,best_current_individual,history_g] = ...
        read_population(fun,Init_Popu,particle,swarm_size,...
        (counter_number+discrete_var_number),feasibles_initial_population,Feasible_Set,↙
penalty_factor);
end

% I pop objective and constraint values.
f = objective_function;
g = constraint;

% Best global position (feasible or infeasible) and objective function
% value
best_glob = best_individual;
```

```
best_glob_f = history_best_f;

% Best global position (feasible) and objective function
% value and constraints.
best_glob_current_f = history_f;
best_glob_current = best_current_individual;
best_glob_current_g = history_g;

% Each particals best position equals the intial position.
best_location = position;

% Calculate penaliest objective function values
penalized_f = lower_penalized(f,g,penalty_factor);

% Each particals best known position equals initial position
best_location_f = penalized_f;

% Calculate max speed
if counter_number > 0
    for i = 1:counter_number
        v_min(i) = LowerB(i);
        v_max(i) = UpperB(i);
        maximum_v(i) = 0.3*(v_max(i)-v_min(i));
    end
end
if discrete_var_number > 0
    for i = (counter_number+1):(counter_number+discrete_var_number)
        v_min(i) = Feasible_Set(i-counter_number,1);
        v_max(i) = max(Feasible_Set(i-counter_number,:));
        maximum_v(i) = rounding(0.3*(v_max(i)-v_min(i)),Feasible_Set(i-↙
counter_number,:));
    end
end

% Set 0 as init speed.
velocity = zeros((counter_number+discrete_var_number),swarm_size);

% Round calculator, size calculator and intertia calculator=1
round_counter = 1;
size_counter = 1;
inertia_counter = 1;


% Loop
while round_counter <= generations

    % Lets make intertia smaller, if best feasible objective function value has not↙
improved during certain amount of cycles
    if inertia_counter > beta_k
        inertia = beta*inertia;
        inertia_counter = 1;
    end

    % Best feasible particletion number of analysis and store the time
    history_f(round_counter) = best_glob_current_f;
```

```
    particle_history(round_counter) = particle;
    t_history(round_counter) = cputime-t0;


    % Print to the screen the progress of calculation
    disp(['Iter.:' sprintf('%4i',round_counter) ' /' sprintf('%4i',generations) ...
        '  Obj.: ' sprintf('%8.2f',best_glob_current_f) '  Analysis: ' sprintf('%4i',↙
particle) ...
        '  Inert.: ' sprintf('%2.4f',inertia)...
        '  Feasibles: ' sprintf('%4i',feasibles_initial_population) ' /' sprintf('%3i',↙
swarm_size)])


    % Speed and position for each particle
    for i=1:swarm_size
        % randomized r1 and r2.
        r1 = rand;
        r2 = rand;

        % calculate new speed per particle
        new_velocity(:,i) = inertia*velocity(:,i) + c1*r1*(best_location(:,i) - ...
            position(:,i)) + c2*r2*(best_glob_current - position(:,i));

        % Is the maximum speed exeeded?
        for j = 1:(counter_number+discrete_var_number)
            if new_velocity(j,i) > maximum_v(j)
                new_velocity(j,i) = maximum_v(j);
           elseif new_velocity(j,i) < -maximum_v(j)
                new_velocity(j,i) = -maximum_v(j);
            end
        end

        % Calculate new position for particle
        new_position(:,i) = position(:,i) + new_velocity(:,i);

        % Did we exceed upper and lower boundaries
        for j = 1:counter_number
            if new_position(j,i) < LowerB(j)
                new_position(j,i) = LowerB(j);
            end
            if new_position(j,i) > UpperB(j)
                new_position(j,i) = UpperB(j);
            end
        end

        % Standardisation of discrete variables
        for j = (counter_number+1):(counter_number+discrete_var_number)
            new_position(j,i) = rounding(new_position(j,i),Feasible_Set(j-↙
counter_number,:));
        end
    end

    % Update position and speed
    position = new_position;
    new_position = [];
```

```
velocity = new_velocity;
new_velocity = [];

% Calc objective and constraints for all particals in new positions
f = [];
g = [];
for i=1:swarm_size
    % Objective fct and constraints
    [new_f,new_g] = feval(fun,position(:,i));

    % Make number of analysis one higher
    particle = particle+1;

    f(i) = new_f;
    g(:,i) = new_g;
end

% Calc penalyzed objective fct
penalized_f = lower_penalized(f,g,penalty_factor);

% Set no of feasible particles to 0
feasibles_initial_population = 0;

% Update the best position and value of each particle in memory
% global best position and value and best feasible position and value
% plus constraint values
for i=1:swarm_size
    % check if best known value is improved
    if penalized_f(i) < best_location_f(i)
        best_location_f(i) = penalized_f(i);
        best_location(:,i) = position(:,i);
    end

    % check if best global value is improved
    if penalized_f(i) < best_glob_f
        best_glob_f = penalized_f(i);
        best_glob = position(:,i);
    end

    % check if best global feasible value is improved
    if penalized_f(i) < best_glob_current_f & all(g(:,i)<0)==1
        best_glob_current_f = penalized_f(i);

        best_glob_current = position(:,i);
        best_glob_current_g = g(:,i);
    end

    % record the number of feasible particles.
    if all(g(:,i)<0)==1
        feasibles_initial_population = feasibles_initial_population+1;
    end
end

% safe the best known and feasible value
if isempty(best_glob_current_g) == 1
```

```
        history_x = [];
        history_g = [];
    else
        history_x(:,round_counter) = best_glob_current;
        history_g(:,round_counter) = best_glob_current_g;
    end
    history_best_f(round_counter) = best_glob_f;


    % check if size and inertia calculators should equal to 1
    if round_counter>2 & history_f(round_counter)<history_f(round_counter-1)
         inertia_counter = 1;
         size_counter = 1;
    else
         inertia_counter = inertia_counter+1;
         size_counter = size_counter+1;
    end

    % increase round calc by 1
    round_counter = round_counter+1;

end

% no of calculation rounds
iterations = round_counter-1;


% return best feasible particletion
best_f = history_f(iterations);
best_x = best_glob_current;
best_g = best_glob_current_g;


% print graphs if requested
if print_results==1
    figure;
    plot([1:round_counter-1],history_best_f,[1:round_counter-1],history_f)
    legend('Best known','Best feasible')
    grid on;
    ylabel('Objective function')
    xlabel('No of Iterations')

    figure;
    plot(particle_history,history_best_f,particle_history,history_f)
    legend('Best known','best feasible')
    grid on;
    ylabel('Objective function')
    xlabel('Analysis')

    figure;
    hold on;
    for i=1:length(history_g(:,1))
        plot([1:round_counter-1],history_g(i,:));
    end
    title('Development of Constraints.');
```

```
    xlabel('Iteration');
end
```

```
function [population,objective_function,constraint,particle_counter,best_f,best_x,...
    best_current_f,best_current_x,best_current_g] = read_population(fun,tiedosto,...
    particle_counter,swarm_size,counter_number,feasibles_initial_population,...
    Feasible_Set,penalty_factor)

pid=fopen(tiedosto);
if pid==-1
    error(' ');
end
i=1;
while 1
   line=fgetl(pid);
   if isempty(line)==1 | line==-1, break, end
   read_alternative(i,:)=str2num(line);
   i=i+1;
end
population=read_alternative';
fclose(pid);
if length(population(1,:))<swarm_size
    error(' ');
end
if length(population(1,:))>swarm_size
    error(' ');
end
if length(population(:,1))<counter_number
    error(' ');
end
if length(population(:,1))>counter_number
    error(' ');
end
feasible_solutions=0;
infeasible_solutions=0;
best_f=inf;
best_current_f=inf;
for i=1:swarm_size
    [f,g]=feval(fun,population(:,i));
    particle_counter=particle_counter+1;
    if all(g<0)==1
        feasible_solutions=feasible_solutions+1;
        current_objective_function(feasible_solutions)=f;
        current_x(:,feasible_solutions)=population(:,i);
        current_g(:,feasible_solutions)=g;
    end
    objective_function(i)=f;
    constraint(:,i)=g;
end
if feasible_solutions<feasibles_initial_population & feasibles_initial_population~=-1
    error(' ')
end
if feasible_solutions>feasibles_initial_population & feasibles_initial_population~=-1
    error(' ')
end
if feasible_solutions~=feasibles_initial_population & feasibles_initial_population~=-1
    error(' ')
end
```

```
penalized_f=lower_penalized(objective_function,constraint,penalty_factor);
[best_current_f,index]=min(current_objective_function);
best_current_x=current_x(:,index);
best_current_g=current_g(:,index);
[best_f,index]=min(penalized_f);
best_x=population(:,index);
```

```
function [population,target_f,constraint,particle_counter,history_best_f,...
    best_individual,history_f,best_current_individual,history_g]=...
    new_population(fun,entire_population,counting_number,feasibles_initial_population,↙
Feasible_Set,penalty_factor,...
    counter_number,LowerB,UpperB)

feasible_solutions=0;
infeasible_solutions=0;

feasibles=[];
current_objective_function=[];
current_constraints=[];

infeasible_current_solution=[];
infeasible_objective_function=[];
infeasible_constraints=[];

counter=1;

best_f=inf;
best_current_f=inf;

particle_counter=0;

if feasibles_initial_population==-1
    for i=1:entire_population
        for j=1:counter_number
            population(j,i) = LowerB(j) + (UpperB(j)-LowerB(j))*rand;
        end
        for j=counter_number+1:counting_number
            profile_number=length(find(Feasible_Set(j-counter_number,:)~=inf));
            profile=Feasible_Set(j-counter_number,(1:profile_number));
            population(j,i)=item_value(profile);
        end
    end
    for i=1:entire_population
        [f,g]=feval(fun,population(:,i));
        particle_counter=particle_counter+1;
        if all(g<0)==1
            feasible_solutions=feasible_solutions+1;
            feasibles(:,feasible_solutions)=population(:,i);
            current_objective_function(feasible_solutions)=f;
            current_constraints(:,feasible_solutions)=g;
        else
            infeasible_solutions=infeasible_solutions+1;
            infeasible_current_solution(:,infeasible_solutions)=population(:,i);
            infeasible_objective_function(infeasible_solutions)=f;
            infeasible_constraints(:,infeasible_solutions)=g;
        end
    end
else
    while feasible_solutions < feasibles_initial_population
        population=[];
        for i=1:entire_population
            for j=1:counter_number
```

```
                population(j,i) = LowerB(j) + (UpperB(j)-LowerB(j))*rand;
            end
            for j=counter_number+1:counting_number
                profile_number=length(find(Feasible_Set(j-counter_number,:)~=inf));
                profile=Feasible_Set(j-counter_number,(1:profile_number));
                population(j,i)=item_value(profile);
            end
        end
        for i=1:entire_population
            [f,g]=feval(fun,population(:,i));
            particle_counter=particle_counter+1;
            if all(g<0)==1
                feasible_solutions=feasible_solutions+1;
                feasibles(:,feasible_solutions)=population(:,i);
                current_objective_function(feasible_solutions)=f;
                current_constraints(:,feasible_solutions)=g;
                disp(' ');
                disp('Amount of feasible solutions:')
                 if feasible_solutions==feasibles_initial_population
                    break;
                end
            elseif infeasible_solutions<entire_population
                infeasible_solutions=infeasible_solutions+1;
                infeasible_current_solution(:,infeasible_solutions)=population(:,i);
                infeasible_objective_function(infeasible_solutions)=f;
                infeasible_constraints(:,infeasible_solutions)=g;

            end
        end
        counter=counter+1;
    end
   if infeasible_solutions < (entire_population-feasibles_initial_population)
        while infeasible_solutions < (entire_population-feasibles_initial_population)
            for j=1:counter_number
                uusi_individual(1,j) = LowerB(j) + (UpperB(j)-LowerB(j))*rand;
            end
            for j=counter_number+1:counting_number
                profile_number=length(find(Feasible_Set(j-counter_number,:)~=inf));
                profile=Feasible_Set(j-counter_number,(1:profile_number));
                uusi_individual(1,j)=item_value(profile);
            end
            [f,g]=feval(fun,uusi_individual);
            particle_counter=particle_counter+1;
            if all(g<0)==0
                infeasible_solutions=infeasible_solutions+1;
                infeasible_current_solution(:,infeasible_solutions)=uusi_individual';
                infeasible_objective_function(infeasible_solutions)=f;
                infeasible_constraints(:,infeasible_solutions)=g;
            end
        end
    end
end
population=infeasible_current_solution;
objective_function=infeasible_objective_function;
constraints=infeasible_constraints;
```

```
for i=(entire_population-feasible_solutions):entire_population-1
    population(:,i+1)=feasibles(:,i-(entire_population-feasible_solutions)+1);
    objective_function(i+1)=current_objective_function(i-(entire_population-↙
feasible_solutions)+1);
    constraints(:,i+1)=current_constraints(:,i-(entire_population-feasible_solutions)↙
+1);
end
penalized_f=lower_penalized(objective_function,constraints,penalty_factor);
target_f=penalized_f;
constraint=constraints;
if length(current_objective_function)~=0
    [history_f,index]=min(current_objective_function);
    index=index+(entire_population-feasible_solutions);
    history_g(:,1)=constraints(:,index);
    best_current_individual=population(:,index);
else
    history_f=inf;
    history_g=[];
    best_current_individual=[];
end
[history_best_f,index]=min(penalized_f);
best_individual=population(:,index);
```

```
function [Z,tp,Z_b,Z_dnv_b,Z_s,Z_dnv_s, S,t_w,w_h] = constrfun(si,ti,coo,L1,...
                                  Li,Pi,sig,f_1,l_s,mirror,t_add,paneldef,D,Fa)
%The function finds the restrictions and the actual values to be used in
%the constrain test.
%Every number in [m]
high=D;
n=length(coo(:,1));
ta=zeros(n,1);

Av  =zeros(n,1);         %Empty vector
A   =zeros(n,1);         %Empty vector
I_x =zeros(n,1);         %Empty vector
Sb =zeros(n,1);         %Empty vector
I_xt =zeros(n,1);         %Empty vector
zi  =zeros(n,1);         %Empty vector
%sig =zeros(n,1);         %Empty vector
for i=1:n

if si(i)==0
ta(i)=0;
else
  %Area of panel+ smeared stiffener
 ta(i)=Fa(i)/si(i);      %[m] thickhness added from smeared stiffener.
end

A(i)=(ti(i)+ta(i))*Li(i);   %[m^2]

if coo(i,4)== coo(i,2) %horisontal plate
    zi(i)=coo(i,2)+(ti(i)+ta(i))/2;
elseif coo(i,1)==coo(i,3)  %vertival plate
    zi(i)=coo(i,2)+Li(i)/2;
else
    %sloped panel
    zi(i)=coo(i,2)+(coo(i,4)-coo(i,2))/2;  %[m]
end


%Weigthed total Area
Av(i)=A(i)*zi(i);                        %[m^3]

%Area moment of inertia about the x - axis
%Mechanics of Materials Ansel C. Ugural
    if coo(i,4)== coo(i,2) %horicontal
        I_x(i) = (Li(i)*(ti(i)+ta(i))^3)/12; %[m^4]
    elseif coo(i,1)== coo(i,3) %vertical
        I_x(i) = ((ti(i)+ta(i))*Li(i)^3)/12; %[m^4]
    else %sloped
        Ix = (Li(i)*(ti(i)+ta(i))^3)/12;
        Iy = (Li(i)^3*(ti(i)+ta(i)))/12;
        tet= atan((coo(i,4)-coo(i,2))/(coo(i,3)-coo(i,1))); %rad
        I_x(i)=((Ix+Iy)/2+(Ix-Iy)/2)*cos(2*tet);
    end

end
```

```
%Nautral axis
zn=sum(Av)/(sum(A));                %[m]

%Moment of inertia z axis
for i=1:n
    Sb(i)=(A(i)* (zi(i)-zn)^2);     %[m^4]
    I_xt(i) = I_x(i) + Sb(i) ;      %[m^4]
end

%Transverce gider requirement
[Z_b, Z_dnv_b,Z_s, Z_dnv_s,S,t_w,w_h]=Trans_girder(coo,Li,L1,Pi,paneldef,l_s,ti,ta,↙
f_1);

%Thickness requrement
ka=max(0.72,min((1.1-0.25*si/l_s).^2,1)); %Begrense?
tp=15.8*ka.*si.*sqrt(Pi)./(sqrt(sig*f_1)*1000); %[m]
%tp2=(t_add(:,1)+t_add(:,2)*L1/sqrt(f_1))/1000; %[m] Used to set the
%feasible range

%tp=max(tp1,tp2);
Z = 10^6*mirror*sum(I_xt/(max(zn,(high-zn)))); % [cm^3]
```

```
function [f,g_t] = particle_fun(x)
%Function checking the constraint and storing the trial

% objective and constraint function value
%f=objective;        %Objective
%g=constraint;       %Constraint feasible -1 to 0 an infeasible is from >0 to 1

fid = fopen('move_val.txt');
C = textscan(fid, '%s %s %f %f %f %f %f %f');
inn1=char(C{1});
inn2=char(C{2});
yinit=C{3};
Cinit=C{4};
excel=C{5};
C_w=C{6};
Zg=C{7};
alfa=C{8};

[Ls, D, T, B, v, C_B, f_1, R_, ~, rho_s, rho_c, H_air, mirror, l_s,...
    coo,Li, T0, K, T_C, paneldef,~] = inputvalues(inn1,inn2,excel);
n=length(Li);
L1=min(Ls,300);
[P,sig,t_add] = platepressure(coo, Ls, T, B, v, rho_c,  H_air,C_B, ...
                f_1, C_w, R_, T0, K,paneldef,D,mirror);
Pi=P(:,1);

t=zeros(n,1);
s=zeros(n,1);
k=1;
for j=1:2:length(x)-1
    t(k)=x(j)/1000;     %[m]
    s(k)=x(j+1);        %[m]
    k=k+1;
end
[~,~, Fa,t_hp,~,HPm]= stipar(s,Li,Pi,l_s,f_1);
[Z,tp,Z_b, Z_dnv_b,Z_s, Z_dnv_s,S,t_w,w_h] = constrfun(s,t, coo,L1, Li,...
    Pi,sig,f_1,l_s,mirror,t_add,paneldef,D,Fa);
f=paramfun(s,t,rho_s,Li,l_s,alfa,mirror,T_C,S,t_w,w_h,Fa,t_hp,Cinit,yinit);
% Less than zero is feasible
g_t=[(Zg-Z)./(Zg+Z); (tp-t)./(tp+t);...
    (Z_dnv_b-Z_b)./(Z_dnv_b+Z_b);(Z_dnv_s-Z_s)./(Z_dnv_s+Z_s);(Fa-(HPm+1/100^2))./(Fa+↙
(HPm+1/100^2))]; %max -> en ut
g_t=max(g_t);
%Records all the values
   fidOUT = fopen('generation_dev.txt','a');
   if max(g_t) <0
        fprintf(fidOUT,'% 9.3f ',[1 max(g_t) f]);
   else
        fprintf(fidOUT,'% 9.3f ',[0 max(g_t) f]);
   end
   fprintf(fidOUT,'% 9.3f ',x');
   fprintf(fidOUT,'\r\n');
   fclose('all');
```

```
function f = paramfun(s,t,rho_s,Li,l_s,alfa,mirror,T_C,S,t_w,w_h,Fa,t_hp,Cinit,yinit)
%Normilising the object function. Variables: stiffener distances in meter and pannel
%plate thicknes. Parameters: Length and dimentioning pressure vary between
%the different panels

%Weigths function (only of input, not mirrored)

[y]= weight(s,t, Li,rho_s,Fa,T_C,S,t_w,w_h,l_s);
%Cost estimate

[C_s1, C_s2, C_w, C_l, Frame]= costsimple(s,t,T_C, rho_s,Li,Fa,S,t_w,l_s,w_h,t_hp);
C=sum(C_s1+C_s2)+sum(C_w)+sum(C_l)+sum(Frame);

f=(mirror*C/Cinit)*alfa+(mirror*y/yinit)*(1-alfa); % Use mirror to ...
                                                   %account for whole cross section
```

```
function[Z_b, Z_dnv_b,Z_s, Z_dnv_s, S, t_w,w_h]=Trans_girder(coo, Li, ...
                                                L1, Pi, paneldef,l_s, ti, ta,f_1)
% Code calculation the demand for the transverse beames. Both DNV demand
% and the actual section modulus on the beam by using the highest pressure
% on the beam and the smallest flange thicknesses.
Panel=girder_flange(coo,paneldef);
GP      =zeros(2,2);
w_h     =zeros(1,2);
t_gir   =zeros(length(Panel),4);

S=[0 0]; %length of beams, one in side and one in bottom
for i=1:length(Panel(:,1))
    pn=Panel(i,2); %bottom
    pi=Panel(i,3); %top
    if Panel(i,1)==1 %Bottom
        %find max P and min t for topp and bottom flande
        %Max pressure
        if Pi(pn)>GP(1,1)
            GP(1,1)=Pi(pn);
        end
        if Pi(pi)>GP(1,2)
            GP(1,2)=Pi(pi);
        end

       if coo(pn,2)==coo(pn,4);
        S(1)=S(1)+Li(pn);
       else
        S(1)=S(1)+(coo(pn,3)-coo(pn,1));
       end

        w_h(1)=coo(pi,2)-coo(pn,2);%y-coordinate of top flange
        t_w=6+0.02*L1/sqrt(f_1); %floor min t
        t_gir(i,:)=[1 ti(pn)+ta(pn) ti(pi)+ta(pi) t_w/1000];

    else %pane(i,1)==2 Side

        %Max pressure
        if Pi(pn)>GP(2,1)
            GP(2,1)=Pi(pn);
        end
        if Pi(pi)>GP(2,2)
            GP(2,2)=Pi(pi);
        end
        S(2)=S(2)+Li(pn);
        w_h(2)=coo(pn,1)-coo(pi,1);
        t_w=5+0.02*L1/sqrt(f_1);
        t_gir(i,:)=[2 ti(pn)+ta(pn) ti(pi)+ta(pi) t_w/1000]; %m
    end
end

t_girb=t_gir(find(t_gir(:,1)==1),:);
t_girs=t_gir(find(t_gir(:,1)==2),:);

if length(t_girb(:,1))>1
[minValb ~] = min(t_girb(:,2:4));
```

```
else
    minValb=t_girb(1,2:4);
end
[minVals ~] = min(t_girs(:,2:4));
tminb=minValb;
tmins=minVals;
t_w=[tminb(3) tmins(3)];

Z_b=sec_modH(tminb(1),l_s,tminb(2),l_s,w_h(1),tminb(3),S(1))*100^3; %cm^3
Z_dnv_b=100*S(1)^2*l_s*max(GP(1,:))/160; %cm^3 -simple beam analyse
Z_s=sec_modH(tmins(1),l_s,tmins(2),l_s,w_h(2),tmins(3),S(2))*100^3; %cm^3
Z_dnv_s=100*S(2)^2*l_s*max(GP(2,:))/(160*f_1); %cm^3 -DNV
```

```
function [PN]=girder_flange(coo, paneldef)
%Find the oposite panel for input to H-beam: thickness of flange, heigth of
%web, thickness of web


t=1; %counter
PN=zeros(length(paneldef),3);
for i=1:length(paneldef)
    if   strcmp(paneldef(i),'BOTTOM_OUT')==1
        if coo(i,4)==coo(i,2) %horizontal panel
      for h=1:length(paneldef) %
       for j=1:length(paneldef)
           if (coo(i,3)==coo(h,1) && coo(i,4)==coo(h,2)) && ...
                   (coo(j,3)==coo(h,3) && coo(j,4)==coo(h,4)) && h~=j
               PN(t,:)=[1 i j];
               t=t+1;
           end
       end
      end
        else %sloped panel
                for j=1:length(paneldef)
                if  i~=j && (coo(i,3)==coo(j,3) && coo(i,4)==coo(j,4)) ...
                        && coo(j,2)==coo(j,4)
                    PN(t,:)=[1 i j];
                    t=t+1;
                end
                end
        end
    end
end

%find the horizontal panel over the keel conected with the center girder
for i=1:length(paneldef)
   if strcmp(paneldef(i),'KEEL')==1

       for h=1:length(paneldef)
                for j=1:length(paneldef)
                if (coo(h,1)==coo(i,1) && coo(h,2)==coo(i,2)) && i~=h && ...
                   (coo(h,3)==coo(j,1) && coo(h,4)==coo(j,4))  && coo(j,2)==coo(j,4)
                    PN(t,:)=[1 i j];
                    H_ib=coo(j,2);
                    t=t+1;
                end
                end
       end
   end
end



for i=1:length(paneldef)
if strcmp(paneldef(i),'SIDE_EXT')==1 && coo(i,2)>=H_ib %indre bunn kode
   for h=1:length(paneldef)
       for j=1:length(paneldef)
           for k=1:length(paneldef)
```

```
            if (coo(i,3)==coo(h,3) && coo(i,4)==coo(h,4)) && ...
                    (coo(k,3)==coo(h,1) && coo(k,4)==coo(h,2)) && ...
                    (coo(i,1)==coo(j,3) && coo(i,2)==coo(j,4)) && ...
                    (coo(k,1)==coo(j,1) && coo(k,2)==coo(j,2)) && k~=j
                PN(t,:)=[2 i k];
                t=t+1;
            end
            end
        end
    end
end
end
PN(PN(:,1)==0,:)=[]; %remove the unused columns
for i=1:length(PN(:,1))
    for j=1:length(PN(:,1))
    if PN(i,2)==PN(j,2) && j~=i && (PN(j,1)~=0 && PN(i,1)~=0)
        if coo(PN(i,3),3)-coo(i,3) < coo(PN(j,3),3)-coo(i,3) %delta x
          PN(j,1)=0;
        else
        PN(i,1)=0;
        end
    end
    end
end

PN(PN(:,1)==0,:)=[]; %remove the empty rows
```

```
function[Z]=sec_modH(tb,lb,tt,lt,h_w,t_w,S)
%Calculates the section modulus of a H-beam. All input must be in m.

ab1=0.6*S/lb; %length flange bottom
ab2=0.6*S/lt; %length flange top
%formula for effective bredt based on DNV table C2, Pt.3 Ch.1 Sec.3
b_ecb=lb*(-0.0525*ab1^2+0.4385*ab1-0.0015); %Effective plate flange
b_ect=lt*(-0.0525*ab2^2+0.4385*ab2-0.0015);
%girder in ship y diraction
%Center of parts
%bottom flange
z_bf=tb/2;          %1 x
z_w=h_w/2+tb;       %2 x
z_tf=tt/2+h_w+tb;   %3 x

A1=tb*b_ecb; % x^2
Av1=A1*z_bf;   % x^3
A2=h_w*t_w;
Av2=A2*z_w;
A3=tt*b_ect;
Av3=A3*z_tf;

zn=(Av1+Av2+Av3)/(A1+A2+A3);

Ix1=b_ecb*tb^3/12; % x^4
Ix2=t_w*h_w^3/12;
Ix3=b_ect*tt^3/12;

A=[A1,A2,A3];
z=[z_bf, z_w,z_tf];
Ix=[Ix1,Ix2,Ix3];

%Moment of inertia z axis
Sb=[0,0,0];
I_xt=[0,0,0];
for i=1:3
    Sb(i)=(A(i)* (z(i)-zn)^2);
    I_xt(i) = Ix(i) + Sb(i) ;
end
hight=(tb+tt+h_w);
Z = sum(I_xt)/(max(zn,(hight-zn))); %x^3
```

```
function [nst,Fai, Fa,t_hp,h_hp,HPm]= stipar(s,Li,P,l_s,f_1)
%Calculating the section modulus demand of a stiffener based on stiffener
%distance and pressure on the panel. Empirical formula.

n=length(Li);
%In vector form:
nst =zeros(n,1);        %Empty vector
Z  =zeros(n,1);        %Empty vector
Zm  =zeros(n,1);        %Empty vector
Fai  =zeros(n,1);        %Empty vector
for i=1:length(Li)
nst(i)=round(Li(i)/s(i));               %Number of stiffeners on the plate.
                                        %Rounded to nearest integer.
Z(i)=(83*P(i)*s(i)*l_s^2)/((160*f_1));      %[cm^3]
Fai(i) = (-7*10^-5*Z(i)^2 + 0.1084*Z(i) + 6.0776); %[cm^2] Area function for stiffener
Zm(i)=(Z(i)*100^3);      %[m^3]
end
%Matrix of available stiffeners
[Fa,HP,HPn,HPm]=profiles(Fai/100^2); %[m^2]
t_hp=HP(HPn,3)/1000; %[m]
h_hp=HP(HPn,2);
```

```
function [Fa,HP,HPn,HPm]=profiles(Fai)
%Set stiffener to closest acceptable. Input area in in m^2
n=length(Fai);
% USER DEFINED:
 HP=[ 7.74 100 6; 9.74 100 8; 9.31 120 6; 11.7 120 8; 13.8 140 8; ...
     15.2 140 9; 16.2 160 8; 17.8 160 9; 20.7 180 9; 23.6 200 9; 29 220 10; 32.4 240↙
10];
 HP=sortrows(HP);
 [hn ~]=size(HP);
 HPm=HP(hn,1)/100^2; %[m^2]
 HPn=zeros(n,1);
 Fa=zeros(n,1);

 for i=1:n
 for j=1:length(HP(:,1))
    if Fai(i)<=HP(j,1)/100^2;
        Fa(i)=HP(j,1)/100^2;
        HPn(i)=j;
        break
    end
    if j==length(HP(:,1)) && HPn(i)==0
        Fa(i)=HPm+2/100^2; % can't go higher. Need to reduce stiffener distance
        HPn(i)=j;
    end
 end

 end
```

```
function[yt]= weight(si,ti, Li,rho_s,Fa,T_C,S,t_w,w_h, l_s)
%Calculates the weigth of the section between two frames including one
%frame in form of the transvers girders.

n   =length(Li);       %Panel number. 2 variables per panel
y1   =zeros(1,n);        %Empty vector
y2   =zeros(1,n);        %Empty vector

%Weight of individual parts
for i=1:n
    %weitht section
y1(i)=rho_s*Li(i)*(ti(i)+T_C(i)/1000); %[kg/m]
    if si(i)==0
    y2(i)=0;
    else
    y2(i)=rho_s*Li(i)*Fa(i)/si(i); %[kg/m]
    end
end
Frame=S.*t_w.*w_h*rho_s/l_s; %[kg/m]
yt=sum(y1+y2)+sum(Frame);
```

```
function [C_s1, C_s2, C_w, C_l, Frame]= costsimple(s,t,t_c, rho_s,Li,Fa,S,t_w,l_s,w_h,↙
t_hp)
%Cost function
npanel=length(Li);
C_s1=zeros(npanel,1);    %steel cost plate
C_s2=zeros(npanel,1);    %steel cost stiffener
C_w=zeros(npanel,1);    %Weld cost
C_l=zeros(npanel,1);    %Labour cost
Frame=zeros(npanel,1);    %Labour cost
%C_weld=zeros(n,1);  %Weld number of meters

t_refp=0.01;
t_refhp=0.01;
c_pl=0.8;  %euro/kg
c_st=1.6;   %euro/kg
K=80; %kg steel/hour
P4_0=2.5; %workload per meter welding og long stiffeners
dP4=0.02;
P10_0=0.15; %workload to prepare 1 m^2 plate
dP10=0.04;
C8x=2;
dc8x=0.05;

%total steel weigth cost, different parameters for plates and stiffeners
%C_pl=(t_p+t_c)*Li*rho_s*c_pl;  %m^2*t/m^3=t/m

for i=1:npanel
   ns=Li(i)/s(i);
   P10=P10_0*(1+((t(i)+t_c(i)/1000)-t_refp)*10^3*dP10);
    %cost of the steel
   C_s1(i)=(t(i)+t_c(i)/1000)*Li(i)*c_pl*rho_s; %Euro/m
   if s(i)==0
       C_s2(i)=0;
       C_w(i)=0;
       C_l(i)=K*c_st*Li(i)*P10;
   else
   C_s2(i)=Fa(i)*ns*c_st*rho_s;
   %consumables by weld
   C_w(i)=Li(i)*(C8x*(1+dc8x*(t_hp(i)-t_refhp)*10^3)/s(i));
   %Labour
   P4=P4_0*(1+(t_hp(i)-t_refhp)*10^3*dP4);
   C_l(i)=K*c_pl*Li(i)*(P4/s(i)+P10);
   end
end
Frame(1:2,1)=S.*t_w.*w_h*rho_s*c_pl/l_s; %euro/m
```