Jib Rest Design for Crane Barge Type CB6324

A New Design Approach

Master Thesis
by
Rian Oudewortel

Student Number:	1353004	
Document Version:	Public	
	December 17, 2015	
Thesis committee:	Dr. A. Romeijn	TUDelft
	Dr.ir. S.A. Miedema	TUDelft
	Dr.ir. J.H. den Besten	TUDelft
	A. van Boheemen MSc	Damen

Abstract

The Damen Crane Barge type CB6324 is a transhipment barge, designed to operate in harbours, inshore and in near coastal waters worldwide, mainly for the on- and offloading of bulk carriers. In the second generation of this crane barge, optimization of the equipment and overall design is sought, increasing usability of the barge and reducing overall costs. The same is sought in the design of a jib rest, as it is found to be heavy and cost reduction can be realized by reducing weight. Additionally, it is suspected that the jib rest can be hit by the crane grab during operation.

The new optimized design is restrained by the need to be able to withstand the load of the crane jib in travel conditions overseas and during its complete operational lifetime. As the barge needs to be able to operate worldwide, identification of Ultimate Limit Loads is key. Lloyds Register provides a conservative method to determine the loads acting on the floating crane barge and its equipment, based on extreme weather conditions. As specific data on the CB6324 is available, a different calculation method is proposed, using motion responses, sea state data and design criteria as an input. The method is based on probability of encountering a sea state, the probability of non-exceedance of a wave in such a sea state, and the operability of the barge, to predict the maximum probable accelerations encountered.

Finally, analysis of two first generation crane barges built, show some fatigue crack forming on the jib rests during transport to the operational locations. This signifies an interest of a fatigue analysis and estimation, adding up to three design objectives:

A) Finding load cases using modeled motion responses and sea state probability

B) Finding a new jib rest design that complies to the criteria set by Damen and with the load cases found with design objective A

C) Finding an estimation on the fatigue lifetime on both the first and new generation jib rest

For the determination of the accelerations, a tool is developed, which automates the calculations for the maximum expected loads in worldwide near shore operation and transport. The barge motion responses in regular waves are determined and used as the primary input. Secondary input is composed of wave data, direction and spreading. Wind speed, roll, pitch, flooding angles and (bending) stresses in the crane pedestal are used as additional criteria. As an output, it gives an overview of operability over the world, along with a selection of the maximum expected accelerations in the critical directions.

Parallel to the load calculations and development of the tool, the design of the new jib rest is made. Starting with a concept and building up detail as more information is brought to light in every calculation step. The resulting design is composed of a 15 ton, hinged, steel A-frame with a weight reduction of approx. 5 ton compared to the first generation. Fatigue prediction shows a significant improvement on the fatigue lifetime, as a transport load case estimates the new design spending approx. 10% of its fatigue lifetime in comparison to approx. 200% in the old design.

More weight reduction could be achieved by further research on whether the f_r is a reasonable restriction for this barge. The stiffness of the jib rest is increased significantly to comply to this criterium in dropped down position. This weight reduction can only be realised if a less conservative load case calculation method is used, like the one proposed in this thesis.

Afstuderen is net watertrappelen

Preface

Workspace Coupling and Python

If there is one thing I have adapted from programmers, is the drive to automate everything that is automatable, especially calculations. Some have asked me if it is really beneficial to spend time on automating a calculation, as it might be used just a couple of times and doing it by hand would suffice. A good argument. But. Engineering relies on software. From communication to numerical modelling. It uses software tools to facilitate in the design, often acquired in large and expensive packages. In my oppinion, (structural) engineering is still done in a somewhat oldfashioned way and is falling behind on the rapid development strategies of software design. There is a lot that can be learned from this other discipline. If we are relying so much on this software, we should at least have an idea how the software is structured and see how calculations are automated. This way, we might find a faster way of improving the engineering design cycle.

Making an attempt to revolutionize the engineering design strategies might be a little far fetched to do next to a design thesis. During this thesis, a reasonably smaller step towards automation is done. A lot of data about this thesis' subject - the crane barge - is scrambled over different drawings and manuals. And there are several software tools used in the process, all using the same parameters and information. Therefore, next to a tool described in this thesis, a small linking scipt is built, to keep all parameters at one location. It handles all parameters, variables, switches, inputs and outputs, coupling the entire workspace. Programs and programming languages like Excel, Latex, Eclipse, Tikz, PRECAL$_R$, Multiframe and Autocad are all linked together, minimizing the cases of redoing. For me, this has fulfilled the need for automation.

Instead of using a large and paid package to do make the code in, I have chosen to use and learn an upcoming and open source programming language; Python. In hindsight, learning a new programming language at the beginning of a thesis while programming a model, designing a jib rest and mainly gaining more knowledge about structural engineering and sea keeping, might not have come at a perfect time. Nevertheless, I am glad I have taken this opertunity to do so.

Acknowledgements

There are quite some people who helped me complete this design thesis, by giving guidance, support and ideas. I would like to thank them, and some in particular:

Arno van Boheemen, my daily supervisor at Damen, for giving me the oppertunity to contribute to one of the products of Pontoons and Barges, allowing me to learn more about Damen and for giving me responsibility for the design of the jib rest.

Hugo Hoekstra and collegues, for sharing their knowledge and ideas, keeping me on track of getting to a realizable jib rest design.

Maximiliano Roth for sharing his experience and insights, and for his involvement in the final phase of the thesis. Especially your patience and undivided attention has been a great help.

My thesis committee members representing TUDelft, for their input on my method and findings. Especially **Arie Romeijn**, for his guidance and giving me confidence. You always seem to be able to make time for a student in need of advice.

Thomas Sneep, a fellow graduate and sparring partner. I have enjoyed our discussions on about everything.

Charlotte de Jong, for her support in these busy times. Now we can close a big chapter and start with new adventures!

Contents

List of Figures

List of Tables

Glossary

A_{grab} 18

A_{total} total added mass matrix 32

A_{tot} crossectional area of the barge 13

$A_{x,z}^{jib}$ Horizontal area jib side 46

A_z response amplitude 32

B_{total} total added damping matrix 32

CI CI 10

CR CR 10

C_{total} total added restoring matrix 32

F_{pre} pretension force according to Liebherr (10% SWL) 46

F_w^{YH} force due to wind pressure on jib 46

F_x^{YH} force due to gravity in x direction for load case YH 46

F_y^{YH} force due to gravity in y direction for load case YH 46

F_z^{YH} force due to gravity in z direction for load case YH 46

$H_s[m]$ significant wave height 44

H_s significant wave height 35, 37

H_{ba} height of the barge 13

I_{xx} moment of inertia over x-axis 12, 13

J_{xx} torsional moment of inertia over x-axis 13

L_{ba} length of the barge 14

OP operability 39, 41

PA parkability 39, 41

$R(H_s, T_z, \omega)$ response spectrum 37

$S_\zeta(H_s, T_z, \omega, \mu)$ wave spectrum 37

T_z zero up crossing period 35, 37

$V(p, w)$ weighed criteria score 23, 27, 29, 31

W_{ba} width of the barge 13, 14

$Wtot$ Wtot 10

X surge 32

Y sway 32

Z heave 32

Φ roll 32, 44

Ψ yaw 32

Θ pitch 32, 44

$\bar{F}$ total excitation vector 32

$\bar{x}$ motion response vector 32

ϵ response phase 33

γ_σ acceleration - stress ratio 53

$\gamma_{f,E,a}$ environmental load factor in load combination a 46

$\gamma_{f,E,b}$ environmental load factor in load combination b 46

$\gamma_{f,G,a}$ permanent load factor in load combination a 46

$\gamma_{f,G,b}$ permanent load factor in load combination b 46

γ_m material load factor 46

γ non-dimensional peak shape parameter 36

κ_x^{jib} roll radius of gyration 33

κ_y^{jib} pitch radius of gyration 33

κ_z^{jib} yaw radius of gyration 33

κ radii of gyration 32

λ_b buckling ratio ix, 51

λ_b lower boundary of allowable buckling ratio 11

$\mu_{p,j}$ ratio between loads transferred to jib rest and crane pedestal 46

μ wave direction 33, 37

ω_{slew} Slewing speed 18

ω_{slew} 18

ω wave frequency 32, 33, 36, 37

ϕ^{grab} maximum admissable heel for operation (Grab) 11

ϕ_{flood} maximum heel allowable heel 10

ϕ 14

ρ_{air} density of air at 0 deg C 18, 46

σ_s spectral width parameter 36

σ_{fat} rough allowable fatigue stress estimation 10

θ^{grab} maximum admissable pitch for operation (Grab) 11

θ_w wave phase 32

ζ^* maximum consistancy error 10

$a_{RP0,y,beam}$ Y-direction set as primary direction in RP:0, while under load of beam waves, defined as load case YB 43

$a_{RP0,y,head}$ Y-direction set as primary direction in RP:0, while under load of head waves, defined as load case YH 43

$a_{RP0,z,beam}$ Z-direction set as primary direction in RP:0, while under load of beam waves, defined as load case ZB 43

$a_{RP0,z,head}$ Z-direction set as primary direction in RP:0, while under load of head waves, defined as load case ZH 43

a_x^{YH} acceleration in x direction for load case YH 46

a_y^{YH} acceleration in y direction for load case YH 46

a_z^{YH} acceleration in z direction for load case YH 46

b_a section width 13

c_{beam} within the beam of the barge 9, 10, 23, 27, 29, 31

c_{compl} complexity of the structure ix, 9, 10, 23, 27, 29, 31

c_{deck} fatigue or structural impact on deck 10, 23, 27, 29, 31

c_{fabr} fabrication time and costs 9, 10, 23, 27, 29, 31

c_{fat} fatigue resistance 9, 10, 23, 27, 29, 31

c_{flex} time and effort needed to move from resting- to working position 9, 10, 23, 27, 29, 31

c_{grab} 18

c_{insusc} insusceptible to cargo/bulk spillage 9, 10, 23, 27, 29, 31

c_j drag coeffecient of the jib 46

c_{maint} maintenace needed including inspection intervals 9, 10, 23, 27, 29, 31

c_{mass} weight of the structure 9, 10, 23, 27, 29, 31

c_{mat} availability of the material 9, 10, 23, 27, 29, 31

c_{mot} effect on motions of the barge in different sea states 9, 10, 23, 27, 29, 31

c_{platf} provide inspection platform 9, 10, 23, 27, 29, 31

c_{size} overall size of the structure 10, 23, 27, 29, 31

c_{sturd} sturdiness of the structure 9, 10, 23, 27, 29, 31

c_{trav} travel conditions 9, 10, 23, 27, 29, 31

c_{winch} obstruction of the winches 10, 23, 27, 29, 31

c_{zone} out of working zone ix, 9, 10, 23, 27, 29, 31

d_{grab} width grab 18

f_r lower boundary of allowable modal frequency i, 51, 52, 60

$m_0(H_s, T_z)$ spectral moment 37

m_{BW} ballasted mass to get
glscog in the barge centre 14

m_{CT} mass crane top 46

m_{JR} approx. mass jib rest (6324JRv9) 85

m_{grab} mass of grabber 18

m_{jib} mass jib 33, 46

m_{ped} mass pedestal 46

$p_{N.E.}$ probability of non exceedance vii, 37, 42, 43, 56

p_w^{YH} wind pressure for load case YH 46

p unweighed criteria score rated from 0 to 1 23, 27, 29, 31

$r_{1/3}$ significant response 37, 39

r_m maximum response 37, 39

t_a section thickness 13

t_{eff} effective thickness 12, 13

u_w wind speed 44, 46

v_{wind}^{max} maximum wind speed at parking condition 11

v_{work}^{wind} maximum wind speed at working condition 11, 18

w criteria weight 23, 27, 29, 31

$x_{C.O.G.}^{jib}$ coordinates of jibs C.O.G. 33

x_{mass} 14

$y_{C.O.G.}^{jib}$ 33

$z_{C.O.G.}^{jib}$ 33

ACK aft-centerline-keel 32

BWM best-worst method ix, 9, 10, 23, 27, 29, 31

CB6324 Crane Barge type 6324 vii, 2, 4, 5, 7, 9, 12, 20, 61

CFD cumulative fatigue damage theory 53

Class rules and regulations set by classification bureaus, e.g. Lloyd's Register (LR) or Det Norske Veritas and Germanischer Lloyd (DNV GL) 4–6, 35, 39, 45, 59

COG centre of gravity vii, 8, 32, 33, 47

DNV GL Det Norske Veritas and Germanischer Lloyd 4, 5, 10, 35, 46

FAT fatigue class 53

FCP fatigue crack propagation theory 53

FEM finite element modelling 7, 47, 57

FoS factor of safety 7, 45, 46

IIW International Institue of Welding 53

joint distribution 3-parameter Weibull probability density function and lognormal distribution vii, 35, 36

JR jib rest vii, 2–7, 9, 12, 18, 20, 21, 23, 24, 45, 47, 49, 51–53, 55, 59–61, 74, 85, 86

LH Liebherr 2, 10, 12

LR Lloyd's Register 4, 5, 10, 39, 43, 45, 47, 59, 61

MCDM multi-criteria decision-making 9

MDEM Marine Design Engineering Mykolayiv 57

MF Multiframe 49, 57

NX Siemens NX 57

P&B Damen Pontoons and Barges 2–5, 9, 10, 12, 18, 51, 59

PRECAL_R Marins PRECAL software to determine ship behaviour in waves 7, 32, 33

Python Python 3.0, a numerical programming language 35

RAO reponse amplitude operator vii, 6, 7, 32–35, 37, 39, 65

RHS rectangular hollow sections 12, 20

RP reference point vii, 32–34, 37, 38, 43, 47, 85

S-N curve stress cycle curve 53

SCF stress concentration factor 53

ULS ultimate limit state ix, 5, 43–47, 53, 57, 59

YH load case with y as primary direction and barge under beam waves 46

1 Introduction

Figure 1.1: Current crane barge design

The Damen CB6324 is a transhipment barge, designed to operate in harbours, inshore and near coastal waters worldwide, mainly for the on- and offloading of bulk carriers. It provides a flexible solution for the transfer of goods in areas which may prove difficult to reach by roads, or where waterdepth in harbours cannot dock larger carriers.

Damen is an international shipyard group, focusing on the market of ship building, repair and conversion. Their design philosophy is standardization; reducing development costs and delivery time, while maintaining quality. This results in a wide range of standardized products, from pontoons to tugs, platform supply vessels and high speed crafts.

Transhipment is considered a new niche market and the CB6324 is, with its second generation, at an early stage of standardization. It evolved from a standard pontoon with a Liebherr (LH) CBG 350 floating cargo crane, along with some accomodation and a set of (optional) equipment, under the care of the product group Damen Pontoons and Barges (P&B). The crane is relatively high with a total height of 31 meters and its jib at 20m, which needs to be supported by a JR in stowed condition during its down time and transport.

1.1 Problem Description

The first generation of the JR design is a stiff K-frame shaped support, positioned on the fore part of the barge. The jib itself is lengthened by an extension where it rests on the JR, moving the JR a couple of meters away from the working range of the crane. In the second generation, opitimization of the equipment arrangement and overall design is sought, increasing the usability of the barge and reducing the overall costs. The same is sought in the design of the JR, as - despite of the jib rest extension - it is suspected that the JR can be hit during operation.

Additionally, the complete construction of the JR is found to be heavy, and cost reduction can be realised by reducing weight. An optimised design might allow this reduction, under the restraint that it will be able to withstand the load of the jib in travel conditions and during its lifetime. As the barge needs to be able to operate worldwide, identification of these loads is key.

Finally, two of the first generation crane barges built, show fatigue crack growth in the JR before starting its operation.

1.2 Design Objectives

Applying to most equipment on top of floating structures, the loads acting on the jib rest are governed by the motions of the barge at sea. And with additional criteria described by P&B as shown in section 3 and with the design methodology description in section 2, the following three design objectives are found:

A) Finding the load cases using modelled motion responses and sea state probability

B) Finding a new jib rest design complying with criteria listed in section 3 and based on the load cases found in

C) Finding an estimation on the fatigue lifetime of both the old and new JR design

1.3 Scope

Although analysing the complete barge will be beneficial to any insights about the its purpose and its functionality, the focus of this thesis will lay on the design of solely the JR. Any results from calculations done might bring up recommendations on enhancements for a third generation crane barge, but for this generation, no alterations will be done to the barge hull, structure or crane.

2 Methodology and Design Objectives

This section describes a summary of steps that will be made to get to a JR design. The criteria provided by P&B give a starting point (section 2.1), and through a review of rules and regulations set by classification bureaus, e.g. LR or DNV GL (DNV GL), an alternative, more specific calculation method for load cases is proposed (section 2.2). This method, the design and an additional fatiuge analyses results into the design objectives and a complete overview of the design process (section 2.3).

2.1 Criteria

The need of a new JR design is initiated by P&B, providing the following main criteria:

1. **Lightweight**; total construction should be lighter than the current K-frame design
2. **Out of working zone**; retractable, moveable or in any way out of the working zone of the crane
3. **Load Capacity**; capable of withstanding loads and damage because of the heavy work environment
4. **Flexible**; Quickly be placed in operating position or travelling position, if the design allows transition between these positions. On/offloading operation time is in the range of 12h to 24h, therefore the position transition time should not take over 1h
5. **Worldwide Operable**; in a wide range of environmental and sea conditions at harbours, shipyards and coastal areas
6. **Insusceptible to cargo on deck**; any bulk spilled on deck should not effect use of the jip rest
7. **Provide a inspection platform**; a platform should be located in the top of the jip rest, for inspection and maintenance of the jip rest and crane boom connection, and should be accessible from main deck
8. **Within beam**; placement of the jib rest in any condition should not be extended over the sides of the CB6324

These criteria are very understandable and do not need a lot of explanation as to why they are considered to be the boundary conditions for a JR, or even any offshore structure. However, this leaves a lot of room for interpretation and a need of definition. As satisfying the criteria *lightweight* or *sturdy* could have opposing effects on eachother for example. Their importance is quantified along with the other main and secondary criteria in section 3 to have a measureable choice of concepts, but their conformity in definition can best be approached by determining loads on the structure and using safety factors on the resulting stresses. Looking for an optimum between the criteria requires a clear view on the expected loads and well defined geometry of the structure.

2.2 Load Identification

These loads are directly related to the most unfavourable motions and environmental conditions the barge would encounter during its operation or transport to its final location, or any forseeable hazardous events. The motions are dependent on many factors and varies significantly between ship types, geometry and location [4, 5, 27]. Class have made an effort to set up guidelines for these motions, with DNV GL emphasizing on finding the unique motion beheaviour per individual case and LR stating representative motions that envelope the motions of a large range of ship types. The guidelines or rules set up by LR stated in a couple of paragraphs state [29]:

> " 2.11.3: In the stowed condition, the crane, its stowage arrangements and the structure in way are to be designed to withstand forces resulting from the following two design combinations:
>
> a Acceleration normal to deck of $\pm 1,0g$
> Acceleration parallel to deck in fore and aft direction of $\pm 0,5g$
> Static heel of 30 deg
> Wind of $63m/s$ acting in fore and aft direction.

b Acceleration normal to deck of $\pm 1,0g$
Acceleration parallel to deck in transverse direction of $\pm 0,5g$
Static heel of 30 deg
Wind of $63m/s$ acting in a transverse direction.

2.11.4 Alternatively, where the crane is to be fitted to a conventional ship and the ships characteristics are known, the forces may be calculated using accelerations obtained from consideration of the ships motions given in Table 4.2.2, together with the force due to a wind speed of $63m/s$ acting in the most unfavourable direction.

2.11.7 Proposals to use other values are to be substantiated by calculations and will be subject to special consideration."

And DNV GL [5]:

" 2.6.1 The wave load analysis may be carried out for operating conditions with specific wave environments at the considered site, and may also be carried out for transit conditions as alternative to the requirements given in the DNV Rules for Classification of Ships Pt.3 Ch.1 Sec.5."

The envelope of application in LR's first guideline uses very conservative values for wind and wave loadings. They consider a maximum roll angle of = 30.00 deg and a wind speed of $v_{wind}^{max} = 63.00$ m/s , plus accelerations which might be on the high side for a crane barge. A roll angle of this magnitude alone would cause flooding, without even considering the wind loads acting on the barge and the relatively high crane on deck. And it could be contemplated whether a near shore crane barge would or could need to survive a hurricane at open sea. When the barge would find itself in such a hazardous environmental condition, the crane and its pedestal would withstand these loads from a 'allowable-stress' point of view, but would also endure high load cycles which would result in shortening of lifetime due to fatigue. In which either the crane or the pedestal would fail. Designing a JR for these conditions will result in an overdimensioned construction, and concequently more use of material.

Keeping the weight in mind, a step towards a more detailed sea motion analysis is made and the resulting loads provide a ULS condition for the JR design. This ULS is expected to be less conservative then proposed by Class and is used for the dimensioning of the JR. When a final design is reached that complies to this ULS and criteria, load cases derived from Class will be applied and the JR dimensions are scaled up if necessary. This thesis will provide a different approach to the calculation of load cases, but when push comes to shove, the JR will have to comply with Class rules.

The sea motion analysis brings in the need of specific sea state data and from the design and sales point of view, one of the criteria set by P&B for the CB6324 is that the barge can operate near shore, all over the world, and is backed up by the demand expectations [14]. Stating a criterium such as this brings in a lot of different sea states to take in to account. Plus, as reviewed in section 5, the transport tow brings in another number of these states. The different sea states and probability of occurance are highly dependant on geological location and time of year [10, 15]. Figure 2.1 shows a map of all nautical areas of which past sea states are recorded and compiled into scatter diagrams [4]. Additionally, this figure shows possible transport routes and the crane barge demand per area. One could discuss whether to use a global or averaged scatter instead of a large number of area data. This would simplify the calculations, but would not give any insights whether *all over the world* is a reasonable criteria or not, and since we are looking for a *maximum* response, averaging would seem unsuitable. Specific operating site scatter data would be the most preferable, but zooming in at these areas is considered to be a sufficient level of detail for now.

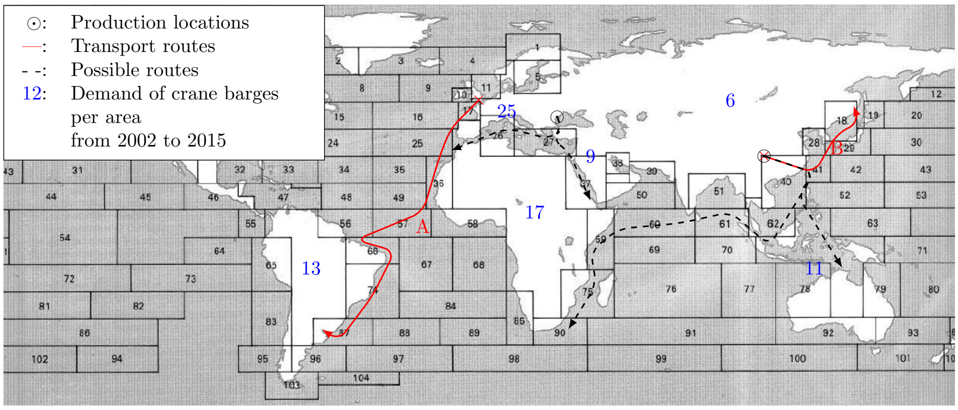


Figure 2.1: Nautical areas and possible transport routes

Worldwide operability is a bold statement for a near shore transhipment barge and introduces a call for quantification of this operability. Class provides rules and guidelines on when the crane is still allowed to operate and when it should be stowed. If the crane has to be stowed a large amount of the time, it might not even be worthwile to position the barge at that location. And if a certain sea state would cause flooding or damage to the equipment on board, that sea state should be avoided at all times. Consequently, its probability can be disregarded from the motion analyses.

To get a more realistic estimation on the loads endured by the JR, a set of criteria, Class rules, sea states and several layers of probability have to be taken into account. This results in a large number of dependent variables and a calculation method that needs to be transparent, expandable and adaptable. As more insight about the expected loads will develop when all responses are found in a particular sea state and can be compared with the criteria.

Handling different calculation methods and dependent variables, while storing a wide range of intermittent data which can be altered and reused at a later stage, asks for a custom calculation tool. A tool which is object oriented and can be changed on the fly. RAOs, scatters, spectra and responses can be initialised as objects, allowing all calculations and manipulations to be done locally in the object itself. Because there are a large number of these scatters, spectra and responses to be analysed only one object per type needs to be checked and tested. Detailed information on the programming structure and code can be found in appendix E.

An additional outcome of the operability is the expected time in stowed position. Overlapping this with the probability a sea state occurs and its wave spectrum provides an estimation on the load cycles the structure would encounter. And thus a possibility to estimate the fatigue lifetime of the JR. A review of the old JR design shows some fatigue crack growth (section 5) and fatigue is considered as a field of interest for this design thesis.

Parallel to the load calculations and development of a tool, a design of a new JR is made, using the information and insights found in every calculation step. Starting with a concept and by adding detail in both the geometry and the evaluation of the design. Finishing with a simplification of the barge, crane and jib rest combination and the loads as load cases.

2.3 Design Objectives

The load and fatigue estimation using motions and probability, and the requirement of a new JR, brings this report to the following design objectives:

A) Finding the load cases using modelled motion responses and sea state probability

B) Finding a new jib rest design complying with criteria listed in section 3 and based on the load cases found in

C) Finding an estimation on the fatigue lifetime of both the old and new JR design

To reach these objectives, the complete design cycle can be represented as a flow diagram as seen in fig. 2.2. Starting with preliminary calculations on any expected imposed forces because of deformations of the barge itself and defining the position of the JR where it could be hit (section 4). Modelling the CB6324 as a simple rectangular barge, a set of RAOs are found (section 7). These are used along with criteria (section 8.3), sea states per nautical area (section 8.1), and factor of safety (FoS) (section 9) as an input for the solver. This solver returns a set of load cases. The complete workflow, including the solver is described in section 8. With these load cases the dimensioning of the basic shape of JR can be made, and checked for nominal stress, buckling, natural frequencies and any supplementary criteria (section 10). As a final step in the design, a fatigue assessment is made along with structural detailing parallel to the engineering and a finite element modelling (FEM) analysis (sections 11 and 12).

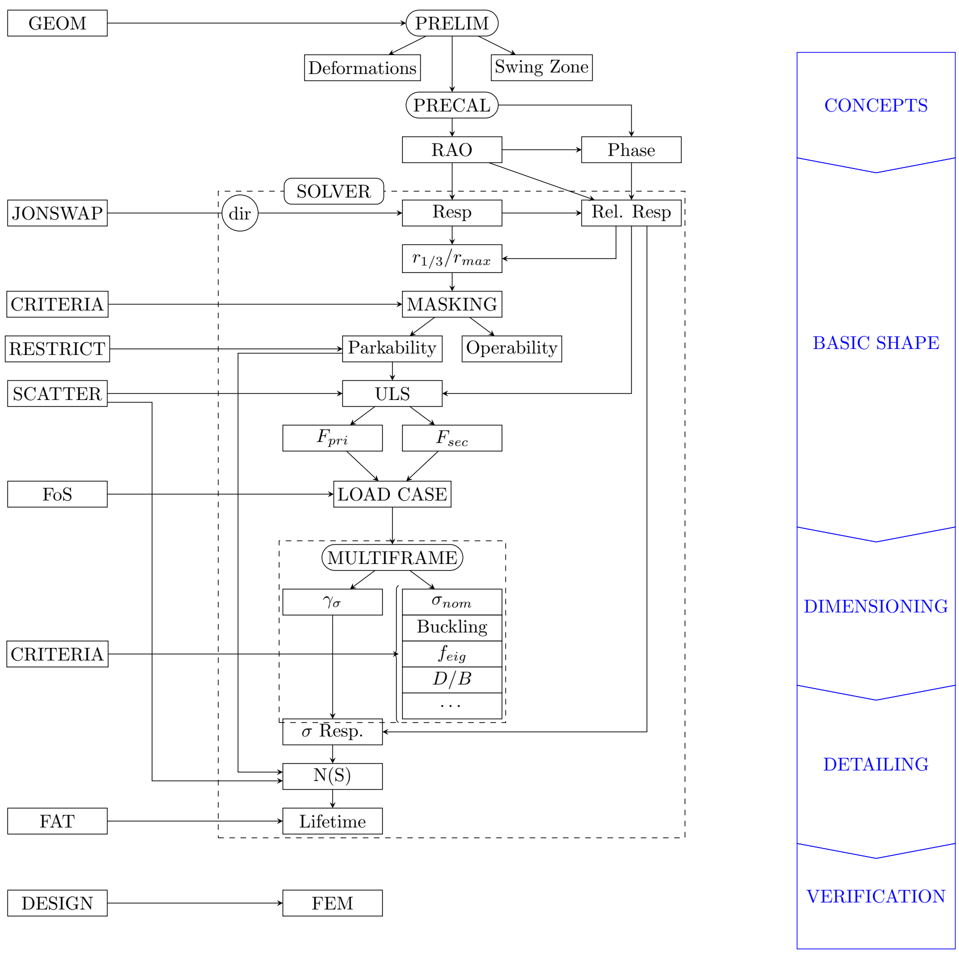


Figure 2.2: Model flowchart

2.4 Assumptions

The design and analysis of the JR rely on a set of simplifications and assumptions in the methodology and model. The load identification, predicting the motions of the barge and fatigue analysis are affected by a broad range of subjects. And before trying to open Pandora's box and investigating all its contents, this section will describe the assumptions made. However, the concept of the model and the programming of the tool is developed with the ability to add functionality where it is required, without having to redo a complete remodelling. The assumptions are as follows:

1. In the sea motion analysis, the barge is simplified as a rectangular barge, disregarding the sloped bow or stern. Mainly because the tool Marins PRECAL software to determine ship behaviour in waves (PRECAL_R) used for the motion beheaviour is validated for this hull shape. Since the sloped parts of the hull is small in respect to the length and width of the barge, disregarding these parts is assumed to have little effect to the motion responses. Additionally, the overall motions are the focus in this analysis and not the drag or speed characteristics, what would bring in more interest in the hull shape.

2. For the determination of the motion responses, the barge is assumed levelled and without any load on deck, as this would be the case during any transport overseas. This is done by finding the COG of the barge, the pedestal, crane top and jib, and ballasting the forward ballast tanks until the total COG is located centrally on the x and y axes.

3. Worldwide operability as a design criteria suggests a usage of an average scatter diagram in response probability analysis. But since the scatter diagrams differ significantly per nautical area - or any location for that matter - and the barge will operate in near coastal areas all over the world, a list of these areas is taken into account. The specific shape of these scatters are adapted from [4].

4. In addition to the coastal areas as described above, areas that are located in probable transport routes are included in the list of scatters.

5. The long term wave statistics described in the scatters per area are assumed to be a sufficient level of detail. And any responses found from these scatters are regarded to have sufficient accuracy for load determination.

6. It is assumed that all sea states can be described by a JONSWAP wave spectrum [4].

CONCEPT PHASE

3 Quantification of Design Criteria

Quantification of the design criteria set by P&B gives the means for a more substantiated choice of concepts. This section describes a method to rate and weigh the criteria. And next to the design aspect, there are some more quantified criteria to which the new JR and crane will have to comply.

In correspondence with P&B, the different design aspects of the JR are categorized as the following design criteria:

1. **Lightweight**; total construction should be lighter than the current K-frame design
2. **Out of working zone**; retractable, moveable or in any way out of the working zone of the crane
3. **Load Capacity**; capable of withstanding loads and damage because of the heavy work environment
4. **Flexible**; Quickly be placed in operating position or travelling position, if the design allows transition between these positions. On/offloading operation time is in the range of 12h to 24h, therefore the position transition time should not take over 1h
5. **Worldwide Operable**; in a wide range of environmental and sea conditions at harbours, shipyards and coastal areas
6. **Insusceptible to cargo on deck**; any bulk spilled on deck should not effect use of the jib rest
7. **Provide a inspection platform**; a platform should be located in the top of the jip rest, for inspection and maintenance of the jip rest and crane boom connection, and should be accessible from main deck
8. **Within beam**; placement of the jib rest in any condition should not be extended over the sides of the CB6324

Design criteria are used as a quantification of all aspects involved, to compare different concepts and measure their design restrictions. To be able to make a trade-off between the concepts in a later stage, the design aspects are projected as criteria in a multi-criteria decision-making (MCDM) problem. To determine the weigh factors of these criteria, the BWM is used.

The following criteria are derived from the design specifications, with additional secondary design criteria:

- weight of the structure (c_{mass})
- out of working zone (c_{zone})
- sturdiness of the structure (c_{sturd})
- time and effort needed to move from resting- to working position (c_{flex})
- travel conditions (c_{trav})
- insusceptible to cargo/bulk spillage (c_{insusc})
- provide inspection platform (c_{platf})
- within the beam of the barge (c_{beam})
- maintenace needed including inspection intervals (c_{maint})
- fatigue resistance (c_{fat})
- effect on motions of the barge in different sea states (c_{mot})
- availability of the material (c_{mat})
- complexity of the structure (c_{compl})
- fabrication time and costs (c_{fabr})

- overall size of the structure (c_{size})
- obstruction of the winches (c_{winch})
- fatigue or structural impact on deck (c_{deck})

Initially, c_{zone} and c_{compl} are selected as the most and least important criteria - respectfully, which the importance of the remaining criteria are compared to, on a scale of one to five. In **??** the criteria are denoted with their scores: 'Most/c' describes how much more important the most important criteria is compared to criteria j. In the same way 'c/Least' compares the importance of criteria j to the least.
With these values, the weigh factor or weight of the criteria can be calculated. This weight is calculated by using the Excel Solver plugin, by keeping the maximum error as low as possible, and keeping the total of the weight one. The maximum consistancy error (ζ^*), CI (CI), total of the weigh factors, and CR (CR) can be found in table 3.1, with c_{zone} as *best* and c_{compl} as *least preferable.*

Table 3.1: Best-worst method

Criteria	Most/c	c/Least	Weight	$\zeta(Most)$	$\zeta(Least)$	$\epsilon(Most)$	$\epsilon(Most)$	description
c_{mass}	1	5	0.11	0.04	0.36	0.36	0.36	weight of the structure
c_{zone}	1	5	0.11	0.00	0.58	0.58	0.14	out of working zone
c_{sturd}	2	4	0.09	0.70	0.30	0.70	0.01	sturdiness of the structure
c_{flex}	2	4	0.09	0.68	0.23	0.68	0.04	time and effort needed to move from resting- to working position
c_{trav}	2	4	0.09	0.72	0.35	0.72	0.00	travel conditions
c_{insusc}	3	3	0.05	0.69	0.59	0.69	0.03	insusceptible to cargo/bulk spillage
c_{platf}	2	4	0.09	0.68	0.22	0.68	0.04	provide inspection platform
c_{beam}	3	3	0.05	0.56	0.71	0.71	0.00	within the beam of the barge
c_{maint}	2	3	0.07	0.35	0.39	0.39	0.33	maintenace needed including inspection intervals
c_{fat}	3	3	0.05	0.58	0.70	0.70	0.02	fatigue resistance
c_{mot}	4	2	0.03	0.43	0.44	0.44	0.28	effect on motions of the barge in different sea states
c_{mat}	5	1	0.03	0.71	0.30	0.71	0.01	availability of the material
c_{compl}	5	1	0.02	0.58	0.00	0.58	0.14	complexity of the structure
c_{fabr}	4	2	0.03	0.58	0.37	0.58	0.14	fabrication time and costs
c_{size}	4	2	0.03	0.55	0.38	0.55	0.17	overall size of the structure
c_{winch}	3	3	0.05	0.56	0.72	0.72	0.00	obstruction of the winches
c_{deck}	4	2	0.03	0.22	0.53	0.53	0.19	fatigue or structural impact on deck

These weigh factors are found with a minimum CR as seen in table 3.2, and therefore are accepted as the weigh factors used.

Table 3.2: Consistancy variables for BWM

Consistancy	Value
ζ^*	0.72
CI	2.30
Wtot ($Wtot$)	1.00
CR	0.31

Apart from the more weighed criteria factors a list of relevant restictions and criteria are defined by P&B, LH, LR and DNV GL. These restrictions are used in the model as described in section 8.3:

- $\phi_{flood} = 14.99\,\text{deg}$; maximum heel allowable heel
- $\sigma_{fat} = 100.00\,\text{MPa}$; rough allowable fatigue stress estimation

- $\lambda_b = 3.00$; lower boundary of allowable buckling ratio
- $v_{work}^{wind} = 20.00\,\mathrm{m/s}$; maximum wind speed at working condition
- $v_{wind}^{max} = 63.00\,\mathrm{m/s}$; maximum wind speed at parking condition
- $\theta^{grab} = 3.00\,\mathrm{deg}$; maximum admissable pitch for operation (Grab)
- $\phi^{grab} = 3.00\,\mathrm{deg}$; maximum admissable heel for operation (Grab)

4 Preliminary Analysis

During the conceptualising of any JR designs, a set of analyses of the barge itself are required. The design philosophy of Damen is standardising; decreasing engineering and development costs and speeding up fabrication and delivery times. In the same way the CB6324 is developed, using a standard pontoon from the product range, and adding a LH crane to it, with any additional required equipment and structures like accomodation and whinches. Only minor changes are made to the construction itself, keeping the main dimensions as set in the pontoon template. This template enables a wide range of applications and its hull and deck construction is designed to withstand any high loads. With a tough hull structure, the pontoon could be considered stiff, but will always have to allow some deformation, depending on the waves it encounters, especially during a tow or operation near shore. Making deformation one of the analyses (section 4.1). Another interesting analysis comes from one of the design criteria for the JR described in section 3: Out of the working zone. The criterium coming from P&B might be substantiated by Murphys law, where having a risk of hitting the JR is enough to develop a new design which minimises this risk. Therefore, an analysis is made to identify the zone on deck in which the crane operates in section 4.2.

4.1 Deformation

Added structures or modifications to the hull have a direct effect on the stiffness of the complete barge itself. Again, the pontoon template might be considered stiff, but a large construction like a stowed crane spanning over a wide distance on deck could function as a frame. This frame could add stiffness to the complete construction and result in large unwanted stresses in the crane, its pedestal or JR. The added stiffness can be subsided by choice of connections between the hull structure and its substructures as described in section 6, but any deformations of the barge during a tow or in operating conditions could lead to forced displacements of the crane and JR.

Barge as Beam Modelling Modelling the CB6324 as a beam provides an effective approach to determine the preliminary deformations of the barge in towing conditions. For this approach, the bending and torsion stiffness is taken into account, as a small deformation of the barge over x and y axis can propagate to a relative large deflection of the jib rest and crane pedestal at a height of 18.08 m .

To determine the stiffness of the barge, the barge sections are simplified as a rectangular hollow sections (RHS) with the dimensions seen in fig. 4.1. For the moment of inertia over x-axis (I_{xx}) aneffective thickness (t_{eff}) is used as the hull thickness, since deck, bottom and walls of the barge are composed of stiffened plates.

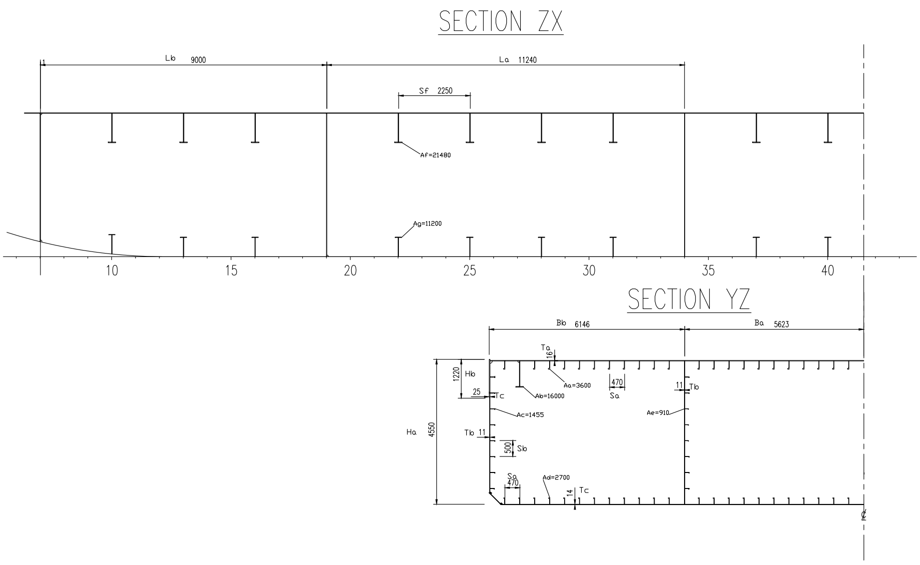


Figure 4.1: Simplified front and side view of the barge sections, with dimensions

With these main dimensions, the following torsional moment of inertia over x-axis (J_{xx}) is determined according [30]:

$$J_{xx} = \frac{4 \cdot A_{tot}}{\oint ds/t}$$

$$J_{xx} = \frac{4 \cdot W_{ba} \cdot H_{ba}}{\sum b_a/t_a} = 8.28\,\mathrm{m}^4$$

And the I_{xx}:

$$I_{xx} = \sum \frac{1}{12} \cdot {t_{eff}}^3 \cdot W_{ba} + \frac{H_{ba} - t_a}{2} \cdot W_{ba} * t_{eff} = 8.69\,\mathrm{m}^4$$

Using these stiffnesses, the forces and moments acting on the barge can be modelled according:

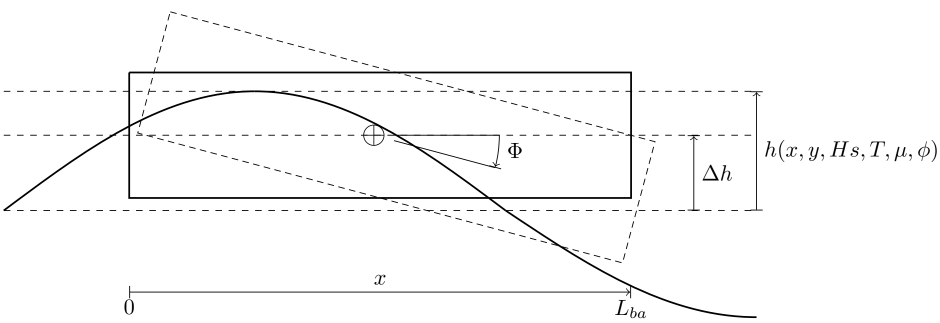


Figure 4.2: Sketch of translation and rotation due to wave height and barge mass

$$\sum F(x) = \int_{-0.5 \cdot W_{ba}}^{0.5 \cdot W_{ba}} \int_0^{L_{ba}} \rho \cdot g \cdot h^*(x, y, Hs, T, \mu, \phi) \delta x \delta y - m_t \cdot g = 0$$

$$\sum M(x) = \int_{-0.5 \cdot W_{ba}}^{0.5 \cdot W_{ba}} \int_0^{L_{ba}} \rho \cdot g \cdot h^*(x, y, Hs, T, \mu, \phi) \cdot x \cdot \delta x \delta y - m_t \cdot x_{mass} \cdot g = 0$$

with [15]:

$$\begin{aligned} h(x, y, Hs, T, \mu, \phi) &= a \cdot \cos\{x \cdot b + y \cdot c - \phi\} \\ a &= \frac{Hs}{2} \\ b &= k \cdot \cos(\mu) \\ c &= k \cdot \sin(\mu) \\ k &= (2\pi/Ts)^2 / g \\ h^*(x, y, Hs, T, \mu, \phi) &= h(x, y, Hs, T, \mu, \phi) - \sin(\theta \cdot x) - \Delta h \end{aligned}$$

Integrating over $x : (0, L_{ba})$ and $y : (-0.5 \cdot W_{ba}, 0.5 \cdot W_{ba})$, and writing this in matrix form gives:

$$\begin{bmatrix} L_{ba} \cdot W_{ba} & L_{ba} \cdot (1/2 L_{ba}^2 - x_{mass} \cdot x \cdot L_{ba}) \\ 1/2 L_{ba}^2 \cdot W_{ba} & W_{ba} \cdot 1/3 \cdot L_{ba}^3 - 1/2 \cdot L_{ba}^2 \cdot x_{mass} \end{bmatrix}$$

$$\begin{vmatrix} -a \cdot (2 \cdot \sin(c \cdot 1/2 \cdot W_{ba})) \\ -a \cdot \left(\frac{2 \cdot \sin(c \cdot 1/2 \cdot W_{ba}) \cdot (b \cdot L_{ba} \cdot \sin(b \cdot L_{ba} + \phi)) + \cos(b \cdot L_{ba} + \phi) - \cos(\phi)}{c \cdot b^2} + m_{BW}/\rho \right) \end{vmatrix}$$

In this preliminary stage, $H_{tow}^{max} = 4.30\,\text{m}$ is assumed to be the largest wave height the barge would encounter and using the barge length to find a wave period of $T_{\lambda=l} = 6.35\,\text{s}$, a regular wave is used to solve the matrix. Figures 4.3 to 4.8 show the deformations, rotations and stresses found in the hull structure for a head wave and different wave phases. In the 3D graphs, the top mesh represents the barge and bottom mesh the wave.

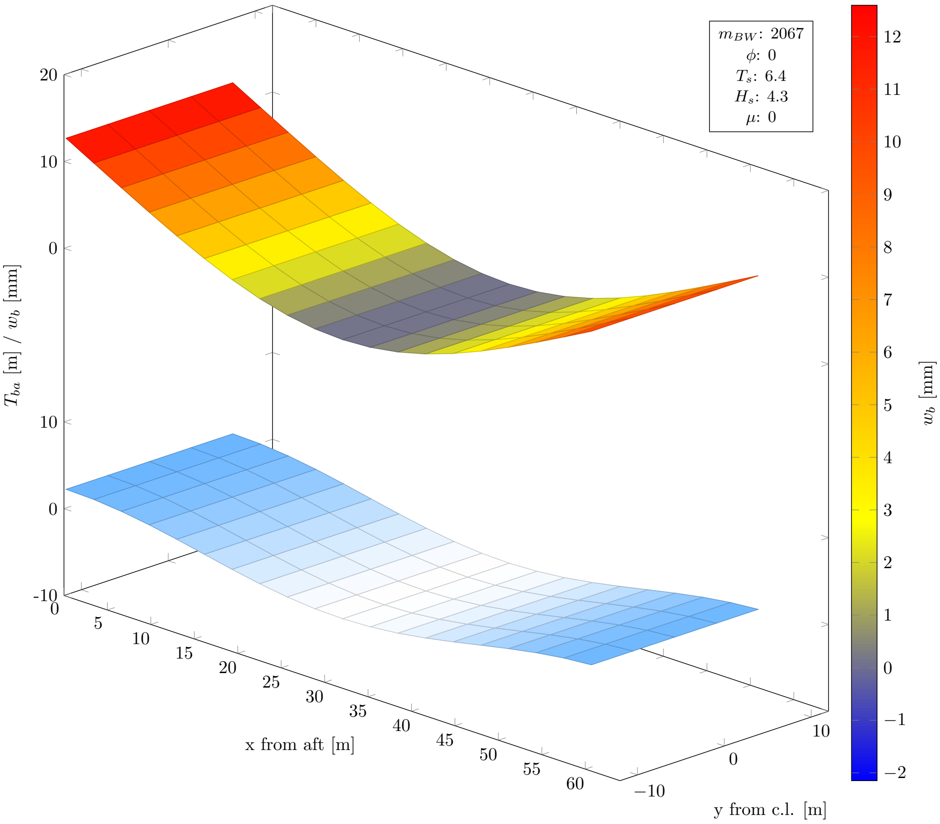


Figure 4.3: Deformations of the barge with respect to z-axis

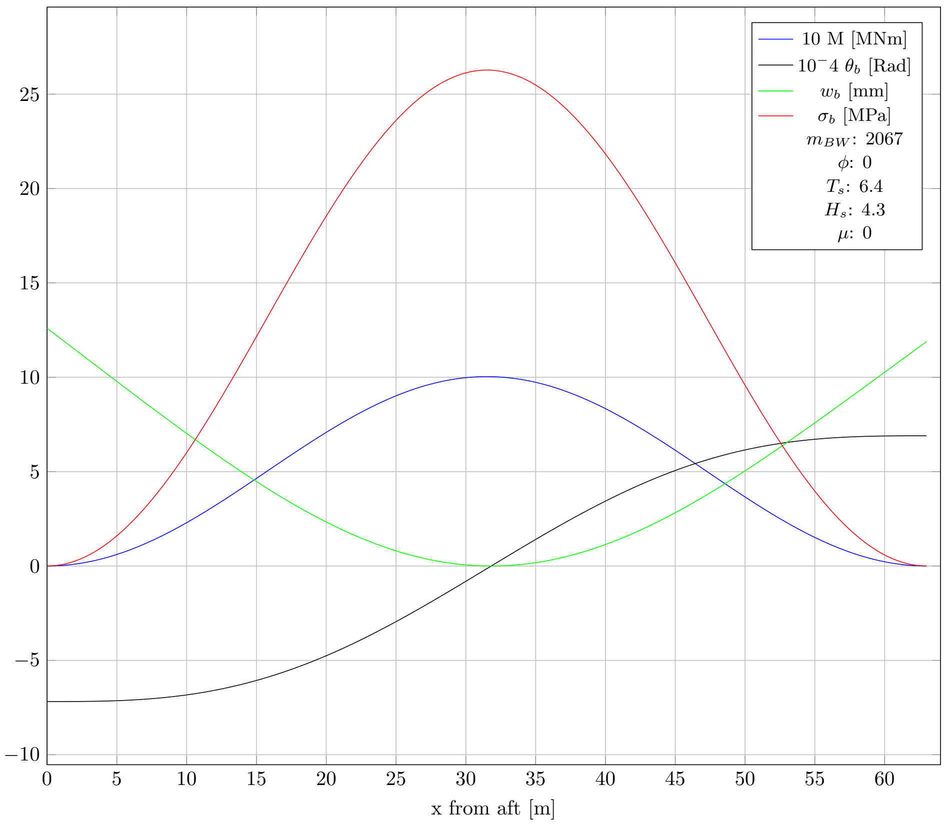


Figure 4.4: Deformations and bending stresses of the barge

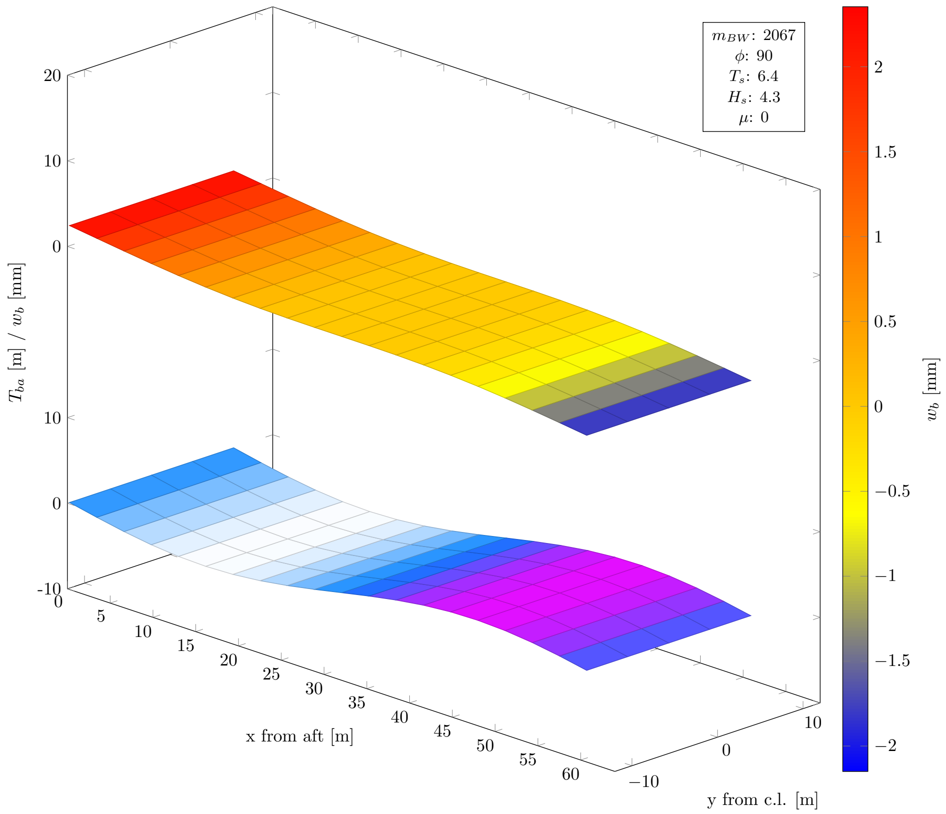


Figure 4.5: Deformations of the barge with respect to z-axis

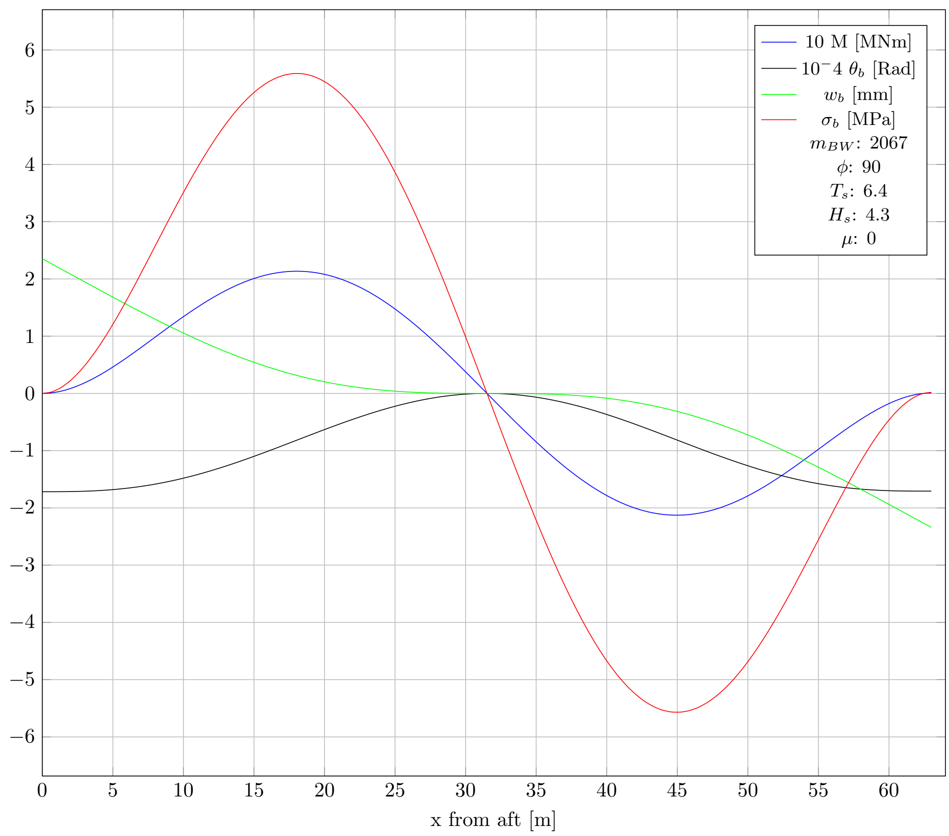


Figure 4.6: Deformations and bending stresses of the barge

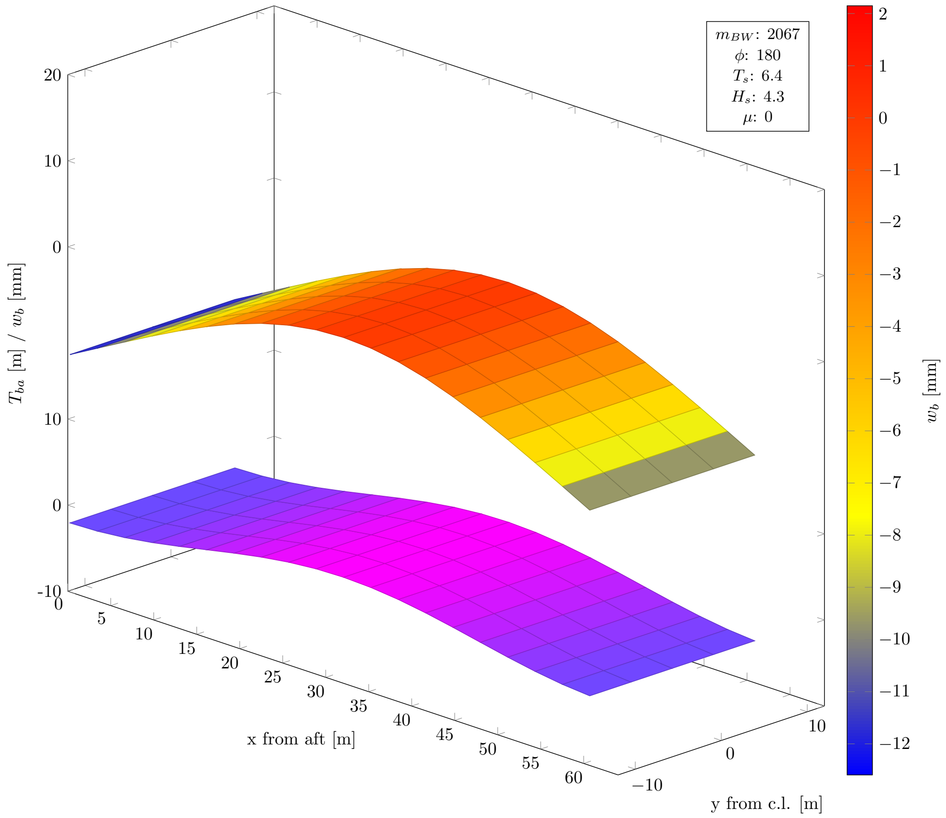


Figure 4.7: Deformations of the barge with respect to z-axis

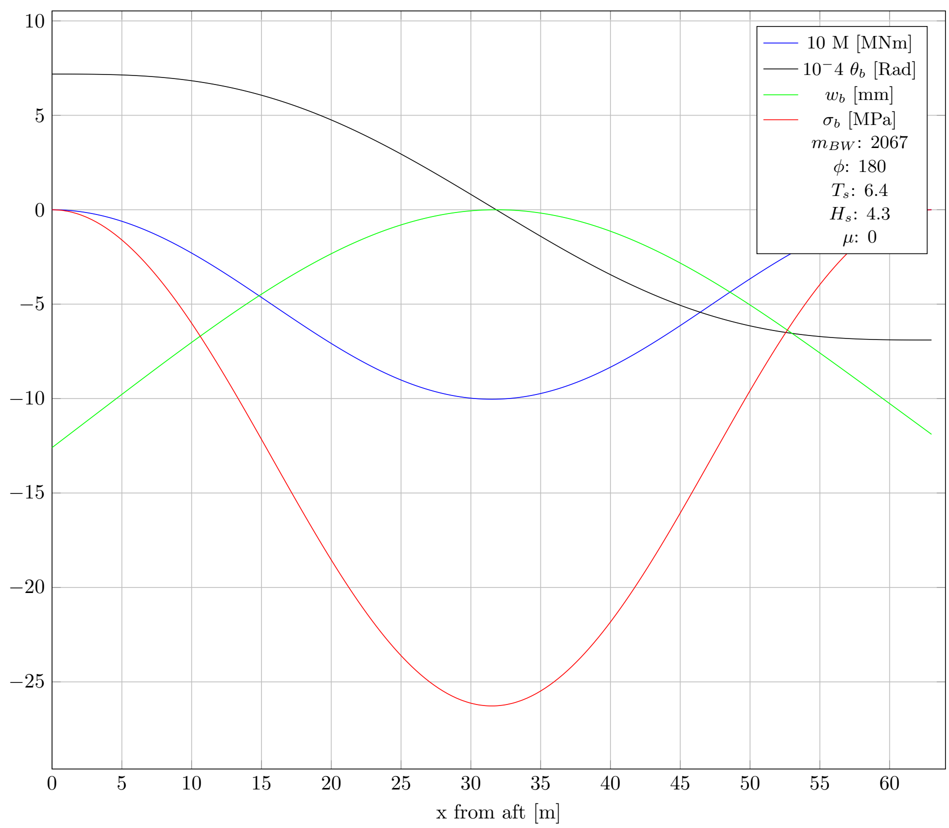


Figure 4.8: Deformations and bending stresses of the barge

This analysis shows relatively low deformations and stresses for a regular wave of $H_{tow}^{max} = 4.30\,\mathrm{m}$, what indicates a stiff barge. With a side note that the total mass is assumed evenly distributed over the complete length and width of the barge, and without taking the deformation of the crane, the pedestal or JR into account. Whether further analysis of the hull structure should be done by including different wave directions and torsion, is up to P&B. But in the writers oppinion, the barge itself is considered as a stiff plane in all further calculations.

4.2 Swing Area

The JR being out of the working zone on deck is mainly dominated by the outreach of the crane, equipment on deck and any space reserved for cargo. The outreach range is enlarged by the rotational speed of the crane and by wind acting on the grab or load, defined as the swing area. A part of this swing out is compensated by extending the crane jib and increasing the distance between the pedestal and JR, but to quantify this distance, the swing out is calculated using the crane specifications.

The Slewing speed (ω_{slew}) is used to determine the swing out of the grab, keeping the distance from deck constant at a varying boom angle. The total force directed outward is considered as the maximum wind acting on the grab plus the rotational force, resulting in the following equations to be solved (eq. (1)) and a schematic in fig. 4.9.

$$
\begin{aligned}
0.5 \cdot c_{grab} \cdot \rho_{air} \cdot A_{grab} \cdot v_{work}^{wind2} / m_{grab} + \omega_{slew}^2 \cdot (R_{tot} + r_{out}) - \tan(\arcsin(r_{out}/l)) \cdot 9.81 = & \; 0 \quad (1) \\
R = & \; R_{tot} + r_{out} + 0.5 \cdot d_{grab}
\end{aligned}
$$

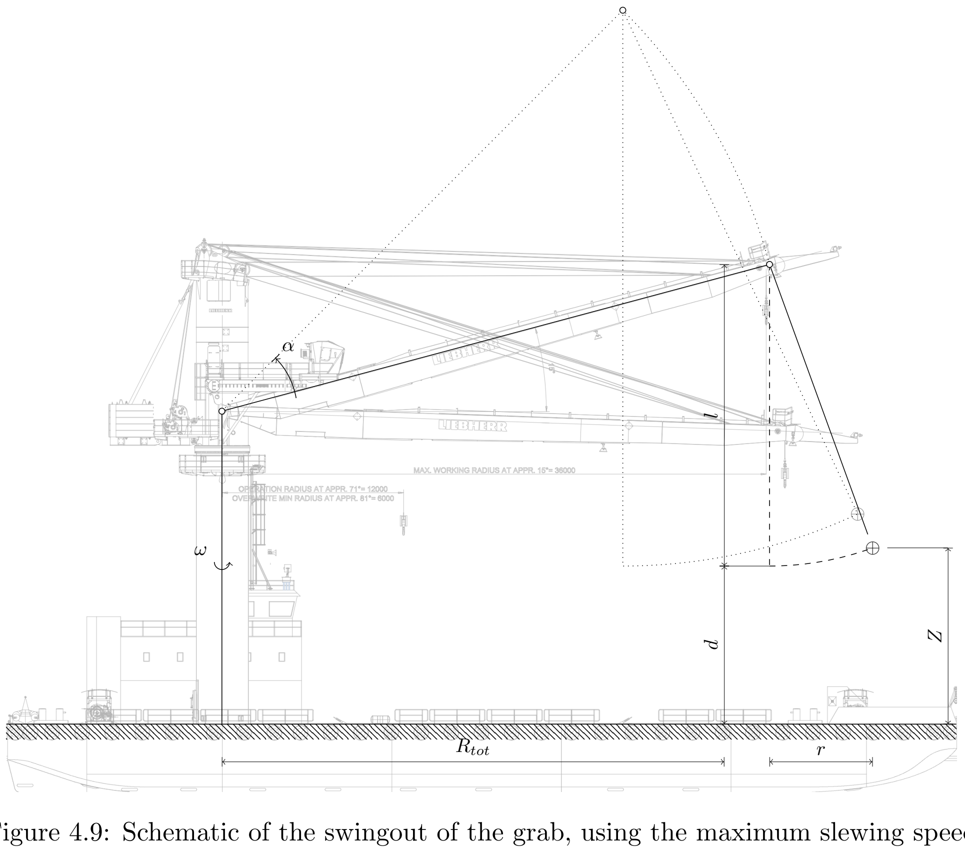


Figure 4.9: Schematic of the swingout of the grab, using the maximum slewing speed

Varying the boom angle and grab or hook distance from deck, the swing out is found for three different configurations: Grab operation without wind, grab operation with wind and (slow) container placement operation (figs. 4.10 and 4.11). This shows that the complete JR would either need to be removed completely, or dropped to an angle of approx. 30 deg relative to deck to ensure no collision can take place.

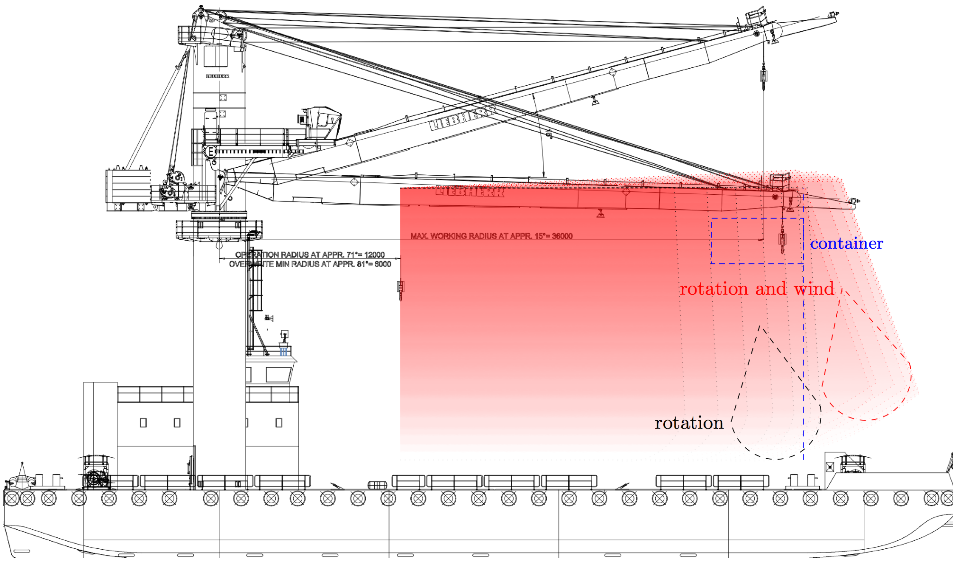


Figure 4.10: Calculated swing out of the grab, and the maximum outreach of the container

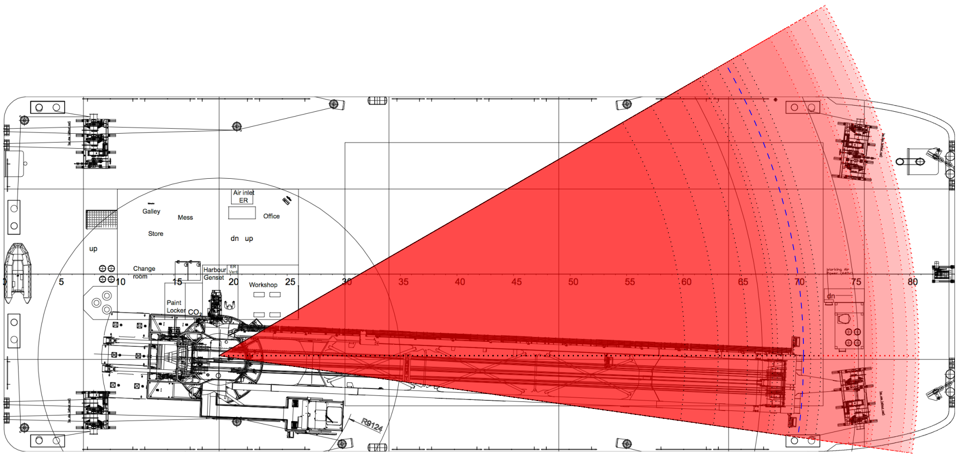


Figure 4.11: Calculated swing out of the grab, and the maximum outreach of the container, top view

5 Current Design Analysis

Two CB6324 are built and both showed issues even before operation. As any product in such an early stage of its life, any issues are to be resolved by the manufacturer and in this case, initiated a design overhaul of the JR. In this section, the old JR design is described, its issues are analysed and set as a benchmark for criteria score.

5.1 Description

The current JR is a K-framed structure of welded RHS members, providing the elevation needed for parking the jib. It is located in front of the crane in longitudinal direction, approx. a meter further than the maximum unloaded outreach of the crane, to reduce the chance of collission with the grab during transshipment operation. See fig. 5.1 for the location and fig. 5.2 for the construction details. This elevation of the JR is bound by the maximum angle in which the boom can be lowered. Lowering it more would either damage the hinges at the crane pedestal or would result in unwanted forces in the crane jib and crane itself. Optionally, the whole JR can be tilted, to provide even more distance between the grab and the rest itself, by use of a cilinder and a hinge on the bottom of the JR as seen in fig. 5.3.

5.2 Fatigue Failure

Two of the CB6324 are operating in Russia and Uruguay and were transported from China and Rotterdam respectively. During either of these transports the barge is towed and the jib is fixed onto the JR by straps with a pretension descibed by Liebherr [21]. Towing reports show that the tow is continued as long the maximum wave height predictions stay below 4.30 m [26, 3].

During the towages both jib rests were damaged and were showing fatigue cracks like seen in fig. 5.4.

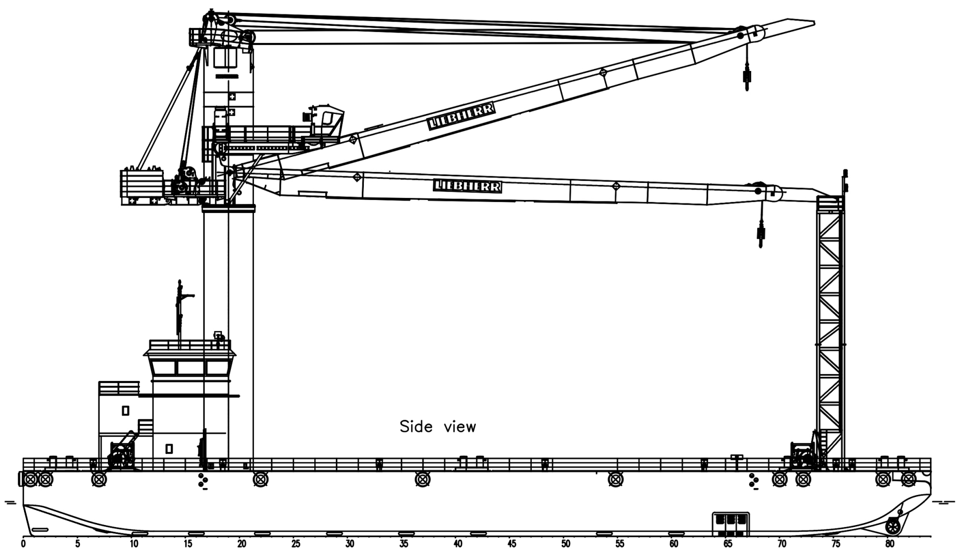


Figure 5.1: Side view of the current CB6324 design

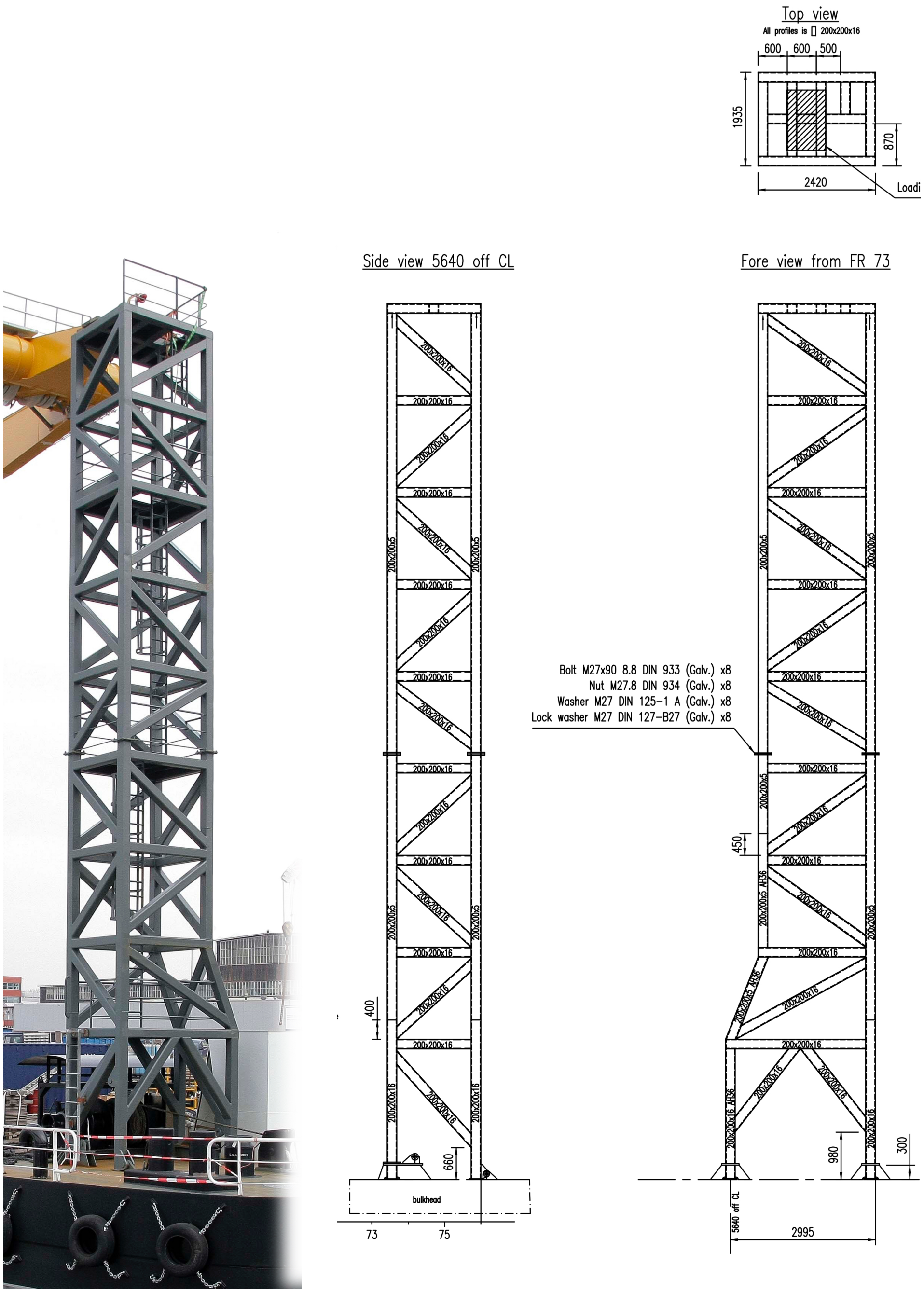


Figure 5.2: Construction details of the current JR design

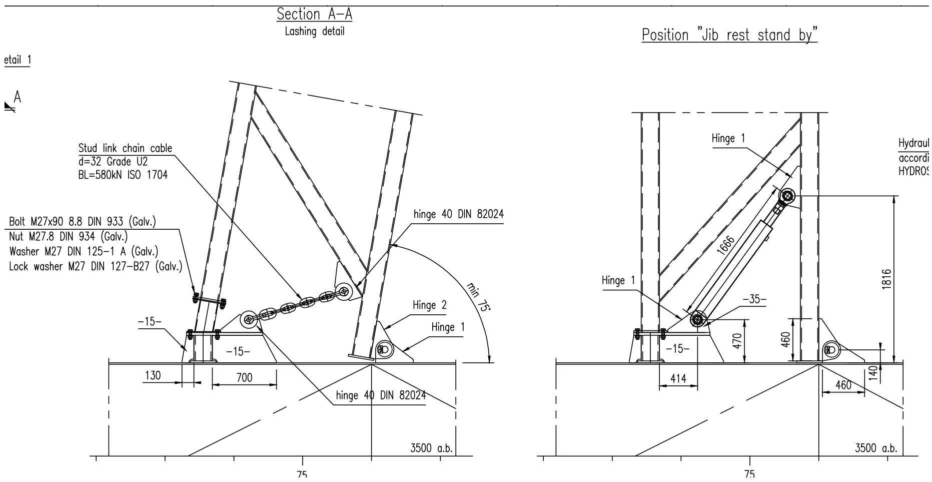


Figure 5.3: Hinge detail

Figure 5.4: Fatigue failure

5.3 Design Benchmark

Looking at the design parameters described in section 3, a benchmark can be set by grading the current JR design on the criteria, which results in the scorecard as seen in table 5.1. In this table the criteria are shown with their scores, weights, weighted scores and its motivation, and summing it up to an overall benchmarked total score of $S_{tot} = 0.53$.

Table 5.1: Current design BWM score

Criteria	p	w	$V(p,w)$	motivation
c_{mass}	0.00	0.11	**0.00**	Benchmarked as Heavy
c_{zone}	0.00	0.11	**0.00**	Benchmarked as 'in working zone'
c_{sturd}	1.00	0.09	**0.09**	A stiff K-frame with 200x200x16 profiles
c_{flex}	0.25	0.09	**0.02**	Hinged with cilinder; chain and bolted removable skids (dwg:523907-540-003); time consuming
c_{trav}	1.00	0.09	**0.09**	Stiff frame unaffected by weather conditions
c_{insusc}	1.00	0.05	**0.05**	Hinges welded on deck and easy acces
c_{platf}	1.00	0.09	**0.09**	Top frame used as platform
c_{beam}	1.00	0.05	**0.05**	Within beam
c_{maint}	0.25	0.07	**0.02**	A lot of welds to be checked; painting of nooks and crannies
c_{fat}	0.00	0.05	**0.00**	Stiff frame; large number of welds; weld failing during first transport
c_{mot}	0.50	0.03	**0.02**	centre of gravity at half of crane boom height
c_{mat}	1.00	0.03	**0.03**	Steel Grade A; standard material used in marine applications
c_{compl}	0.50	0.02	**0.01**	Simple welding/production procedures; but a lot of welds and different sizes
c_{fabr}	0.50	0.03	**0.02**	Large number of welds
c_{size}	0.50	0.03	**0.02**	Overall size slightly on the bigger size
c_{winch}	1.00	0.05	**0.05**	K-frame provides room for winches
c_{deck}	0.25	0.03	**0.01**	4-point welded connections on deck
			0.53	

6 Concepts

Before providing a solution for a new JR design, the complete barge with its drawings is analysed, looking for any information what could affect the design and the predifined criteria. Categorizing sub challenges and providing them with sub solutions gives a schematic overview of the more 'complex' overall system. Several viable combinations of the sub solutions are evaluated and scored according the weigh factors found in section 3, of which the three best concepts rise. Further analysis, during the development of the solver and definition of the basic shape, some of the sub solutions of two concepts are merged, as it provides a better design.

6.1 Morphologic Analysis

Figure 6.1 represents the categorisation of sub solutions, where *frame type* denotes the frame of the JR itself, *transition* the if and how it is removed or relocated. How this relocation is realised is described as *actuation*, connection between the jib and jib rest as the *jib connection*, *jib connection location* the actual location beneath or near the jib and the *unused location* as the place the JR is stored during operation.

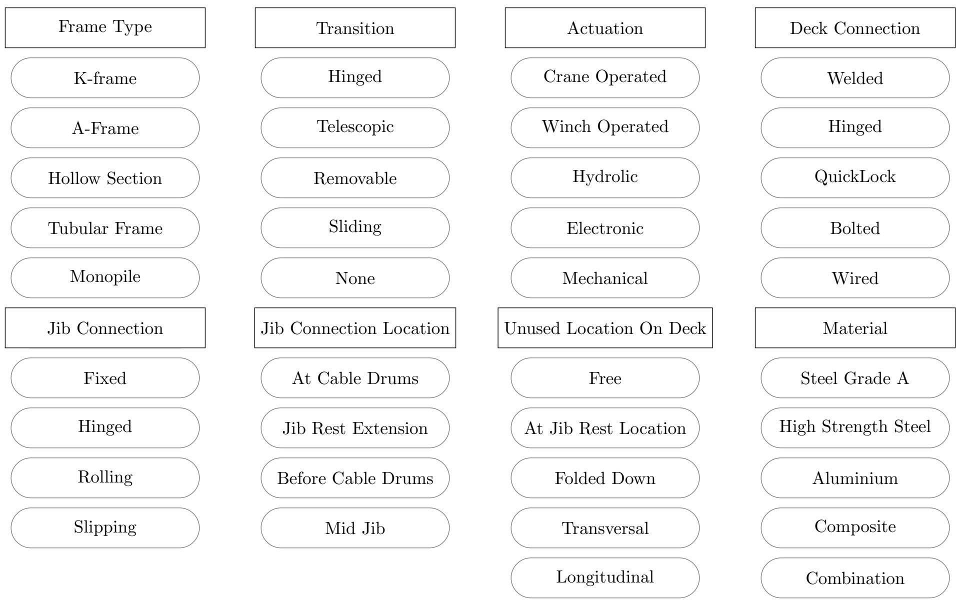


Figure 6.1: Morphologic Overview

6.2 Concept I

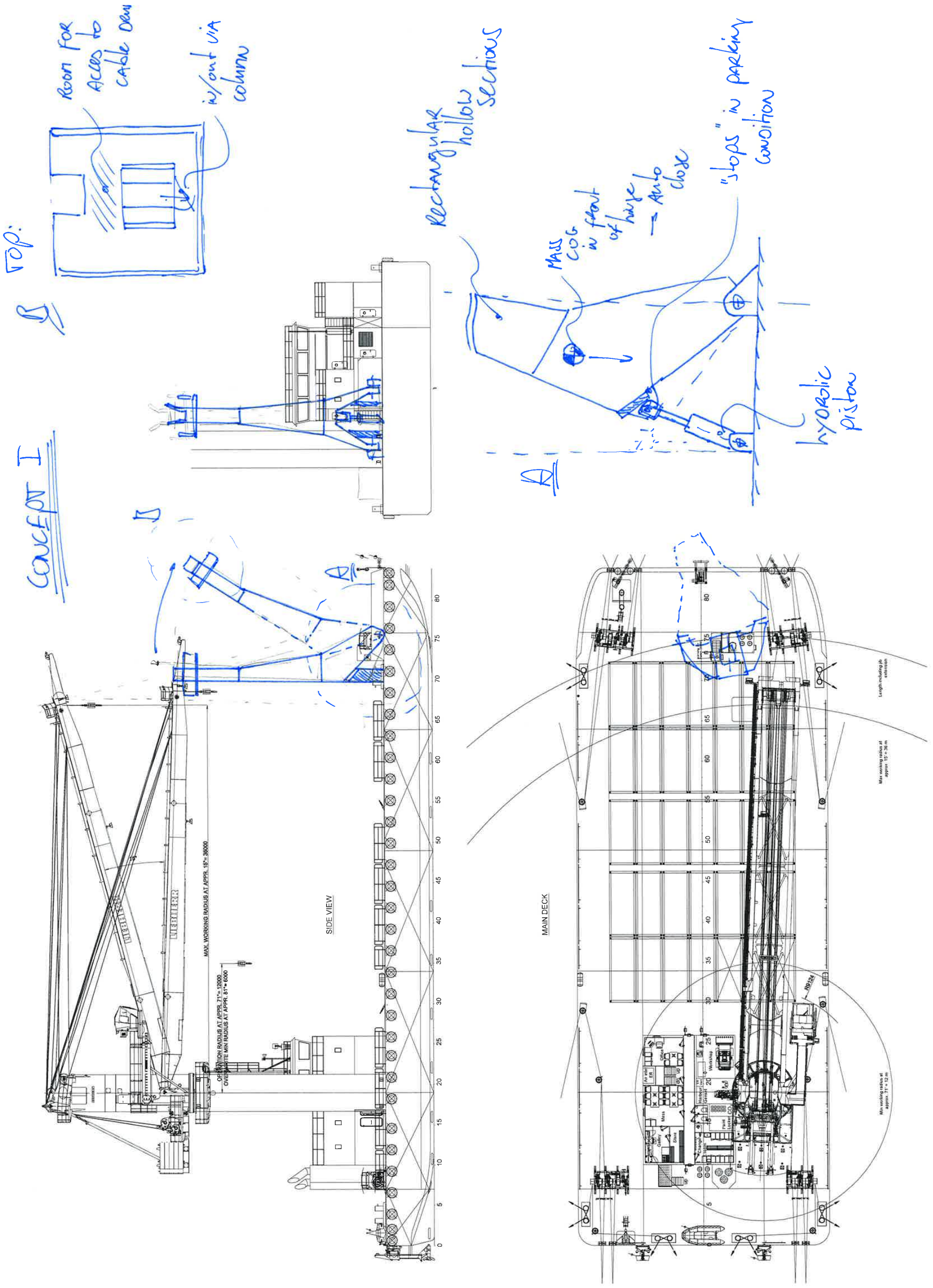


Figure 6.2: Sketch of Concept I

This first concept is based on a hydraulically operated, bottom hinged rectangular hollow section functioning as a boom rest. The hollow section mimmics the shape of a crane boom, where steel plates are welded to each other to create the section, allowing automated welding techniques.

The width of the structure would increase from top to deck to prevent having large bending stresses on deck or in the construction itself, without having a lot of impact on the deck area. In fig. 6.3 the used configuration elements are highlighted.

Figure 6.3: Morphologic overview concept I

Using the criteria set up in section 3, the concept scores can be found in table 6.1, with an overall score of 0.88 .

Table 6.1: BWM score of concept I

Criteria	p	w	$V(p,w)$	motivation
c_{mass}	1.00	0.11	**0.11**	From the first estimations of size; the structure can be slender and thin walled
c_{zone}	0.75	0.11	**0.08**	Hinge and cylinder will make the jib rest be able to move out of the way; but if the structure should be self returning; the rest could still be in the way
c_{sturd}	1.00	0.09	**0.09**	Even though it is thin walled; the hollow rectangular structure should be sturdy; and can be increased by increasing the thickness of the plating
c_{flex}	1.00	0.09	**0.09**	Using the hydraulics; it should be very simple to change from parking to working position
c_{trav}	0.75	0.09	**0.07**	Cylinder must be able to withstand forces due to wind and wave forces when it is returning to its parked condition
c_{insusc}	0.75	0.05	**0.04**	Hinges and cylinder are exposed to bulk spillage; if no protective measures are taken
c_{platf}	1.00	0.09	**0.09**	Structure is very capable to contain a service platform
c_{beam}	1.00	0.05	**0.05**	If placed in x direction; there is no overhang to either PS or SB
c_{maint}	0.75	0.07	**0.05**	Minimum number of welds and rectangular hollow sections with a large area would allow for minimum maintenance. However; bearings and the cylinder would require scheduled maintenance.
c_{fat}	1.00	0.05	**0.05**	The low number of welds and low number of tension concentrations minimises potential crack initiations. Only the base and top of the structure - two point hinges; one point cylinder; and the jib cradle are the significant hotspots
c_{mot}	0.75	0.03	**0.02**	Lowered mass and low centre of gravity would affect the barge motions as less
c_{mat}	1.00	0.03	**0.03**	Grade A steel will be available at all production locations
c_{compl}	0.75	0.02	**0.02**	Other than shape of the to be welded plating; no complex parts are in the structure
c_{fabr}	1.00	0.03	**0.03**	Simple material; welds and drawings
c_{size}	1.00	0.03	**0.03**	
c_{winch}	1.00	0.05	**0.05**	Placement out of the way of and next to the whinches
c_{deck}	0.25	0.03	**0.01**	three point connection to the deck result in high concentrations of forces
			0.88	

6.3 Concept II

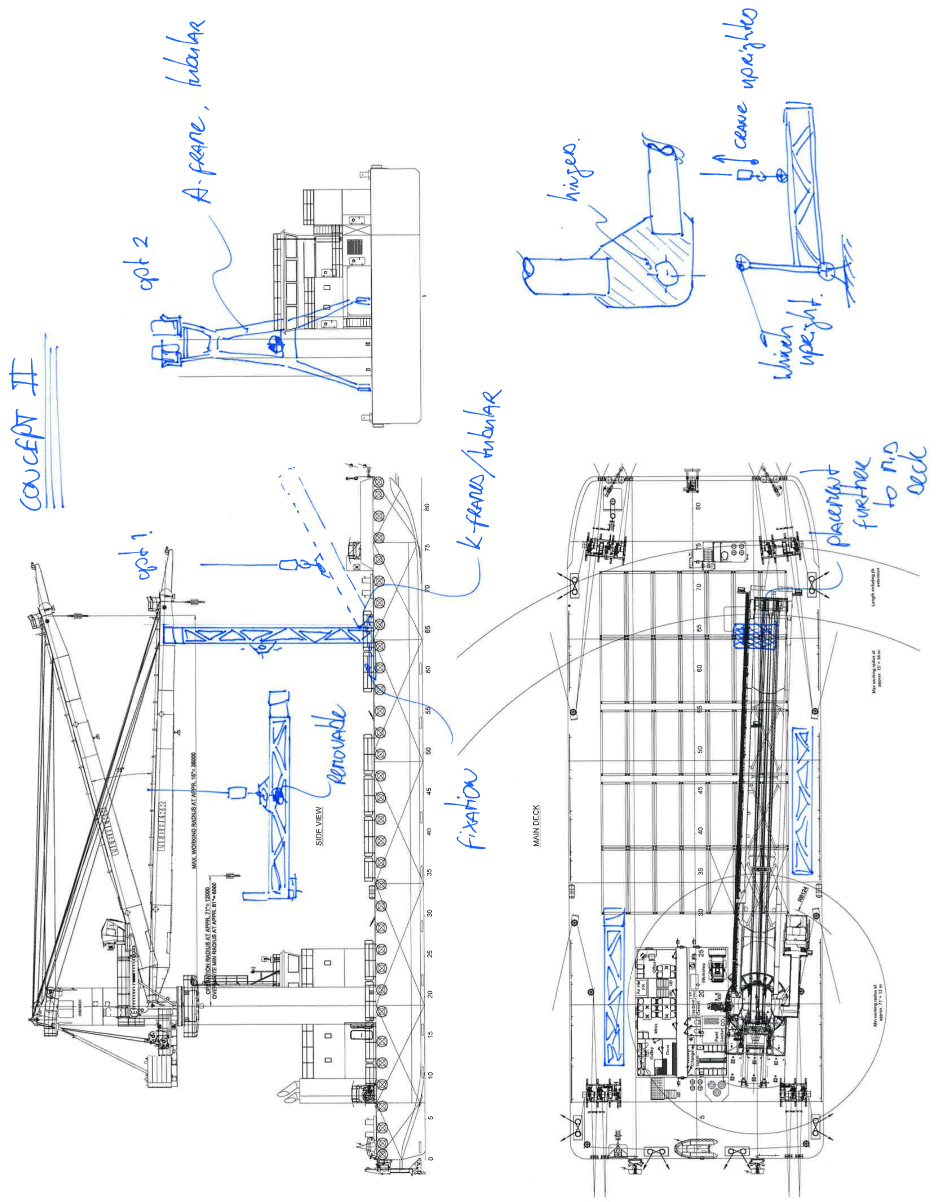


Figure 6.4: Sketch of Concept II

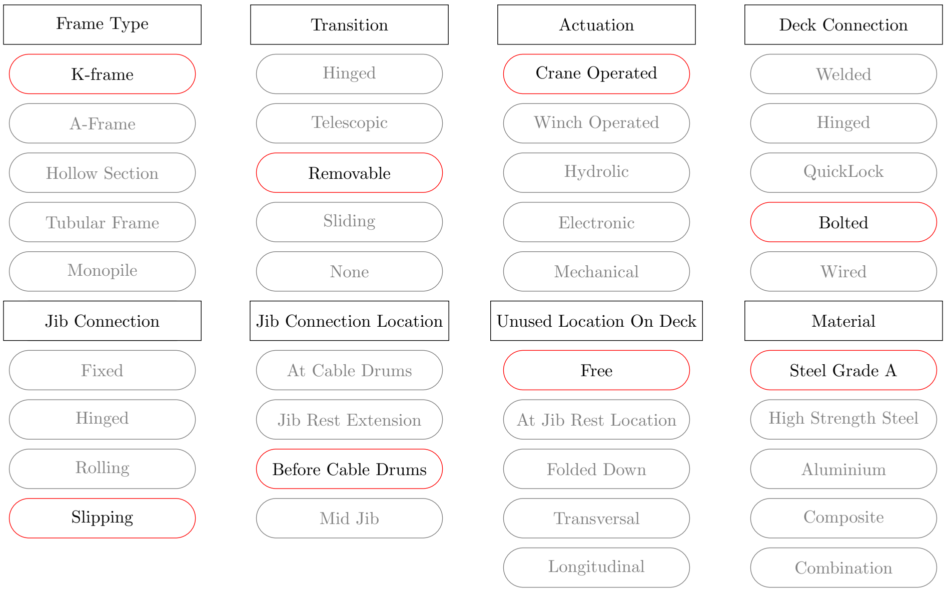


Figure 6.5: Morphologic overview concept II

Table 6.2: BWM score of concept II

Criteria	p	w	$V(p,w)$	motivation
c_{mass}	0.75	0.11	**0.08**	Designed shape results in smaller frame parts in the top
c_{zone}	1.00	0.11	**0.11**	complete removal of the jib rest
c_{sturd}	1.00	0.09	**0.09**	Tubular members
c_{flex}	0.50	0.09	**0.04**	Placement / removal is time consuming
c_{trav}	0.50	0.09	**0.04**	Placement in harsh weather conditions might prove difficult
c_{insusc}	1.00	0.05	**0.05**	Other than the deck connection; the structure is insusceptable to bulk damage (depending on storing position)
c_{platf}	1.00	0.09	**0.09**	Structure is very capable to contain a service platform
c_{beam}	1.00	0.05	**0.05**	
c_{maint}	0.50	0.07	**0.03**	A lot of welds to be checked; painting of nooks and crannies
c_{fat}	0.50	0.05	**0.02**	large number of welds
c_{mot}	0.75	0.03	**0.02**	Lowered mass and low centre of gravity would affect the barge motions as less
c_{mat}	1.00	0.03	**0.03**	Grade A steel will be available at all production locations
c_{compl}	0.50	0.02	**0.01**	large number of welds
c_{fabr}	0.50	0.03	**0.02**	large number of welds
c_{size}	1.00	0.03	**0.03**	
c_{winch}	1.00	0.05	**0.05**	Placement out of the way of and next to the whinches
c_{deck}	1.00	0.03	**0.03**	three point connection to the deck result in high concentrations of forces
			0.79	

6.4 Concept III

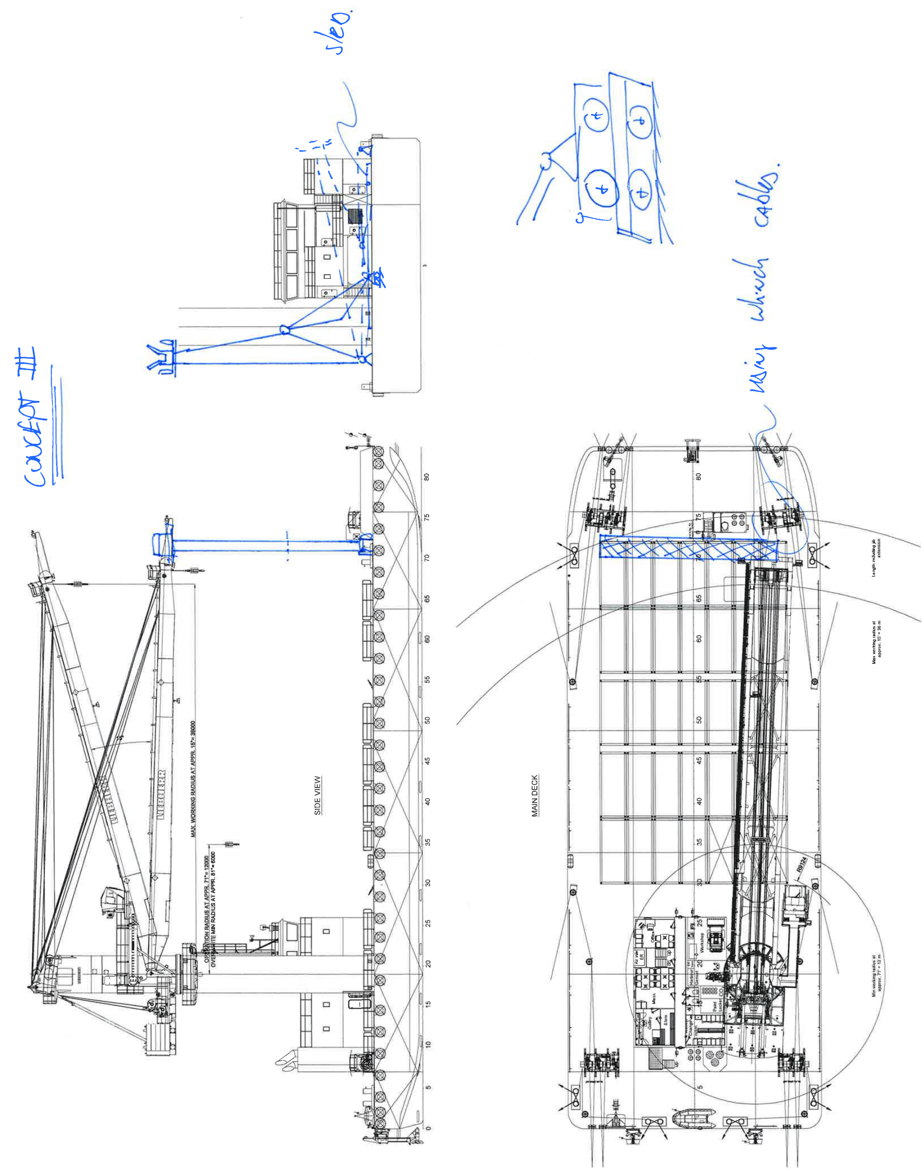


Figure 6.6: Sketch of Concept III

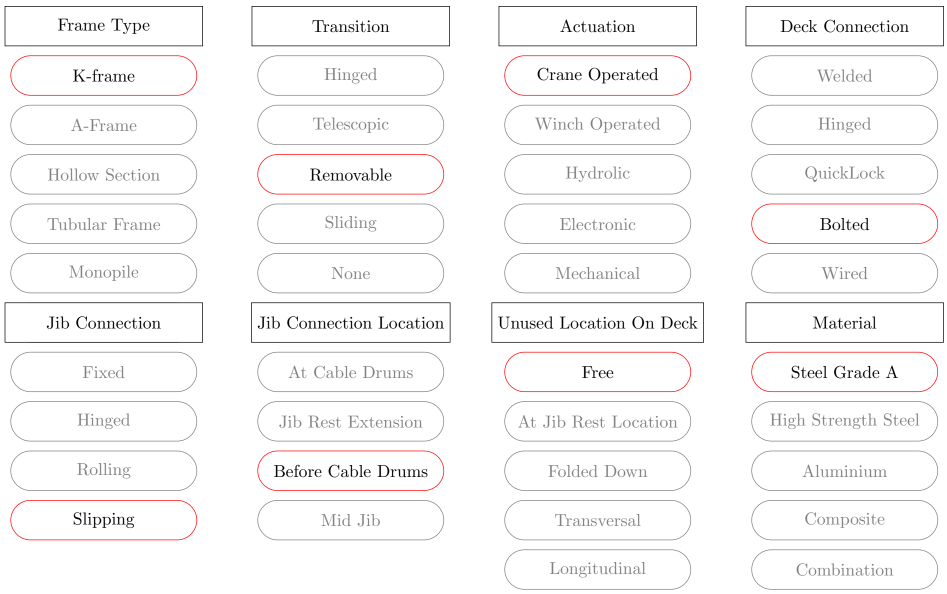


Figure 6.7: Morphologic overview concept III

Table 6.3: BWM score of concept III

Criteria	p	w	$V(p,w)$	motivation
c_{mass}	0.50	0.11	**0.05**	hinging construction would be large
c_{zone}	1.00	0.11	**0.11**	dropped down would be out of the swing zone
c_{sturd}	0.50	0.09	**0.04**	multiple hinges has impact on the overall sturdyness
c_{flex}	1.00	0.09	**0.09**	using either hydraulics or winch cables
c_{trav}	0.50	0.09	**0.04**	
c_{insusc}	0.50	0.05	**0.02**	large area which could be affected by cargo spill
c_{platf}	1.00	0.09	**0.09**	
c_{beam}	1.00	0.05	**0.05**	
c_{maint}	0.25	0.07	**0.02**	a lot of sliding or hinging parts
c_{fat}	0.75	0.05	**0.04**	
c_{mot}	1.00	0.03	**0.03**	
c_{mat}	1.00	0.03	**0.03**	
c_{compl}	0.25	0.02	**0.01**	
c_{fabr}	0.50	0.03	**0.02**	
c_{size}	0.25	0.03	**0.01**	
c_{winch}	0.50	0.05	**0.02**	
c_{deck}	0.25	0.03	**0.01**	
			0.67	

7 Response Amplitude Operators

After conceptualizing and having part of the preliminary results done, a step towards encountered motions is taken by calculating the barge responses. RAOs are the base of predicting any acceleration, and is used as the main input of the solver, which makes this one of the most critical subjects in the model. Deticated validated software is used to calculate the responses, as integration into the tool would be possible and interesting, but out of the scope of this thesis.

The responses of the barge are found by solving the equation of motion (eq. (2)) [19, 8] in any point, with respect to the and are represented by a RAO and a phase for every motion reponse.

$$\overrightarrow{F} = (-\omega^2(M + A_{total}) - i\omega B_{total} + C_{total})\overrightarrow{x} \tag{2}$$

Where:

$A_{total} =$ total added mass matrix
$B_{total} =$ total added damping matrix
$C_{total} =$ total added restoring matrix
$\bar{F} =$ total excitation vector
$\bar{x} =$ motion response vector
$\omega =$ wave frequency

PRECAL_R is a tool which handles ship geometry and mass to calculate the motion responses. In a nutshell, PRECAL_R needs dampening coefficients, masses, draft and the barge hull geometry as a panel model to calculate the rigid body motions, while passing a regular 1m high wave through the centre of the barge at different frequencies. Resulting are the rigid body motions at the COG, namely; surge (X), sway (Y), heave (Z), yaw (Ψ), pitch (Θ) and roll (Φ) in the form of [19, 15, 8]:

$$X = A_x \cos(\omega t + \epsilon_x) \tag{3}$$

Where:

$A_z =$ response amplitude
$\theta_w =$ wave phase

From which the RAOs follow. Combining the main motions with location information, results in relative motions in any point from the COG. And the needed accelerations can be found by integrating these relative motions twice, which results for RAOs for accelerations.

The barge itself is simplyfied as a rectangular hull, without any sloping or bow shape. The pedestal, crane boom and crane top are modelled as mass elements with their radii of gyration (κ) and the relative distance from aft-centerline-keel (ACK) to their local COGs introduced as RPs, as shown in fig. 7.1. Additionally some locations near the jib rest are also added as RPs. Presumably for this design stage, the heavyest environmental loads would occur during transport to its first operating location. During this transport, the barge is unmanned, presumed with an empty deck and levelled by ballasting. Arriving at the following values as input for the PRECAL_R calculations []:

Crane Jib

m_{jib}	39.20 t
$x^{jib}_{C.O.G.}$	32.40 m
$y^{jib}_{C.O.G.}$	−5.40 m
$z^{jib}_{C.O.G.}$	24.31 m
κ^{jib}_{x}	1.48 m
κ^{jib}_{y}	10.96 m
κ^{jib}_{z}	12.51 m
RP	0

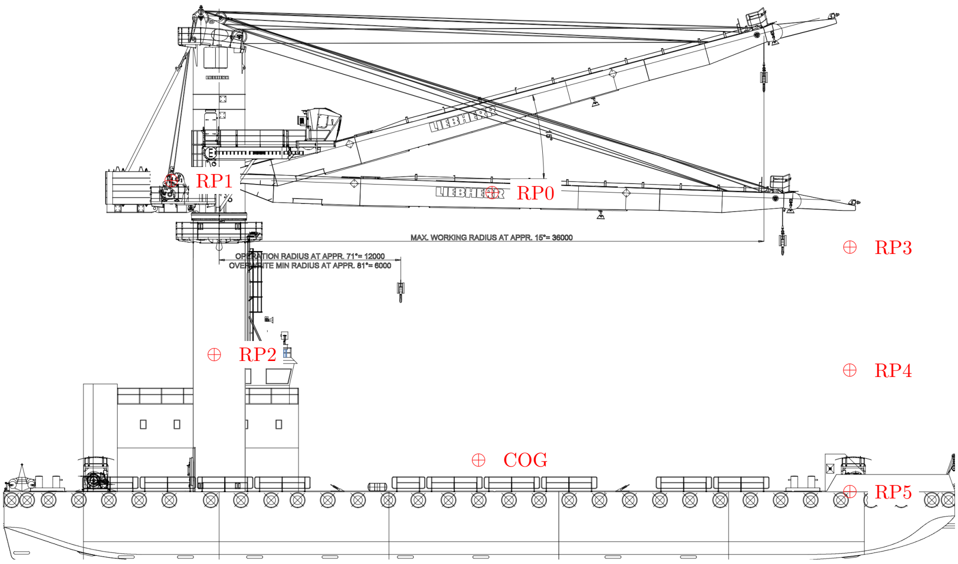


Figure 7.1: Local COGs as RP

Varying the wave frequency (ω) and wave direction (μ), PRECAL_R gives an output in the form of RAO and response phase (ϵ) per ω per motion or RP, as plotted in fig. 7.2, fig. 7.3 and **??**.

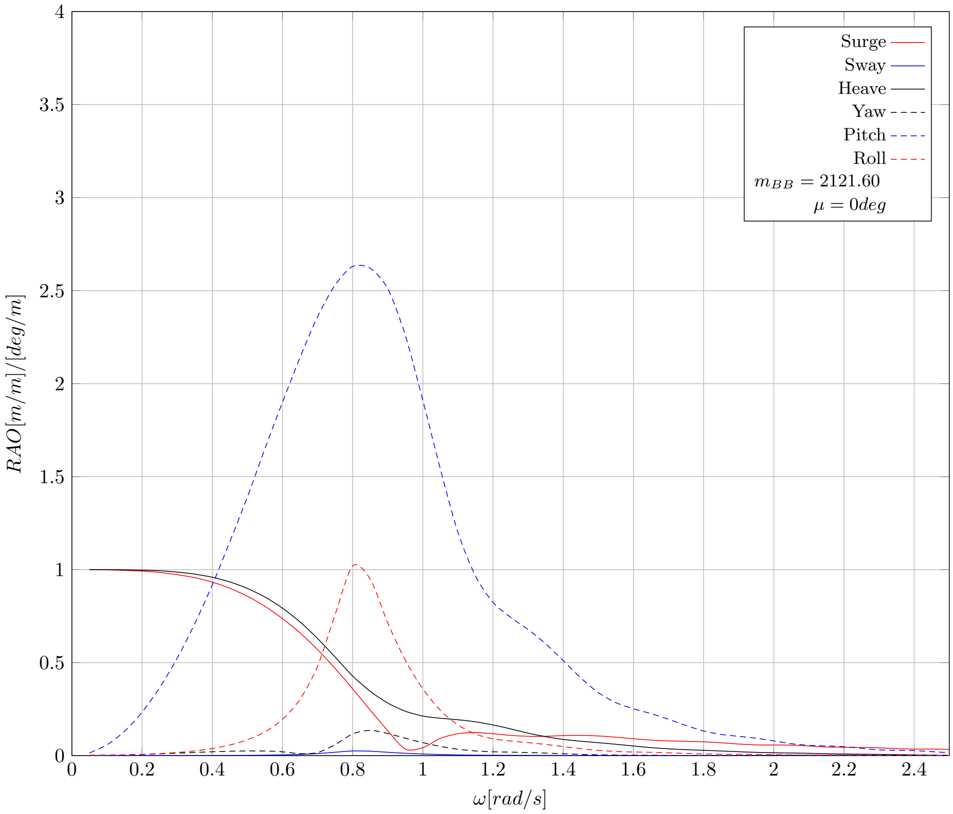


Figure 7.2: RAOs for head waves

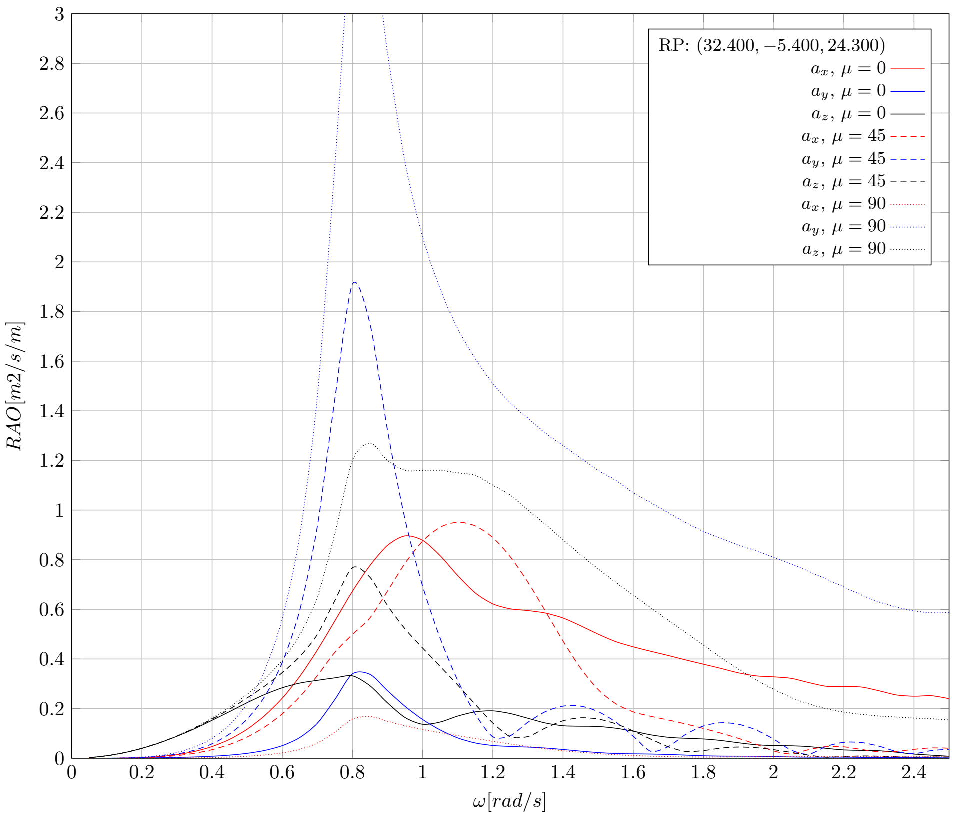


Figure 7.3: Acceleration RAOs for RP:0

BASIC SHAPE PHASE

8 Solver

The solver built in Python and its contents described in this chapter, combines the acceleration RAOs with a range of representations of sea states and digests it into a map of significant or maximum responses that the barge would encounter in those sea states. These responses are compared with all criteria and any structural or Class imposed restrictions to provide the operability of the barge. From this operability, or rather its parkability - explained in section 8.4 - and the probability of occuring seastates, the maximum accelerations and concequently the maximum loads are found. Keeping the structure of the model as seen in fig. 2.2 in mind.

8.1 Sea Modelling

The significant wave height (H_s) and zero up crossing period (T_z) that are measured during a sea state and its occurance are recorded in the scatters and could be seen as stochastic distributions [31, 17, 22]. DNV GL provides parameters for all these joint distribution scatters formulated as a joint distribution [23, 4] (eqs. (4) and (5)). fig. 8.1 shows a graph of the joint distribution.

$$p(H) = \frac{\beta_{H_s}}{\alpha_{H_s}} \left\{ \frac{H - \gamma_{H_s}}{\alpha_{H_s}} \right\}^{\beta_{H_s}-1} \cdot \exp - \left(\frac{H - \gamma_{H_s}}{\alpha_{H_s}} \right)^{\beta_{H_s}} \tag{4}$$

$$p_(Tz|H) = \frac{1}{\sigma t\sqrt{2\pi}} \cdot \exp \left\{ -\frac{(\ln t - \mu)^2}{2\sigma^2} \right\} \tag{5}$$

$$= p(H)p(Tz|H) \tag{6}$$

With:

$$\mu = 0.70 + a_1 H^{a_2}$$
$$\sigma = 0.07 + b_1 \exp(b_2 H)$$

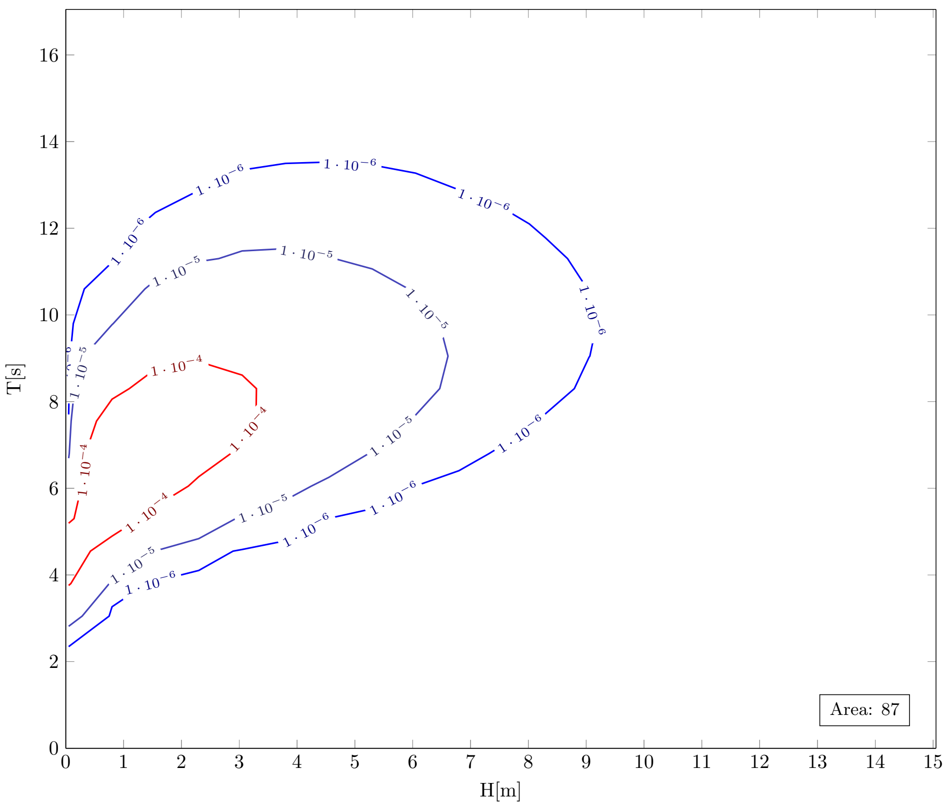


Figure 8.1: Joint distribution of area 87

This concludes the modelling of when and if the sea states will occur, but these joint distributions do not contain all information about the distribution of wave heights and periods during this sea state. This is where a irragular wave description in the form of a wave spectrum is introduced [16, 19]. The wave surface elevation can be described by a superpositioning of an infinite number of waves with varying amplitudes and wave frequencies. Its information of occuring or its (physical) amplitude per ω are stored in variance or energy density spectra, respectively. Common models are the Bretschneider-, Pierson-Moskowitz or the JONSWAP-spectra [16, 4]. The latter is considered to be the most representative in this design and is used for every sea state analysed. The equation describing the JONSWAP can be found in eq. (8) [4, 19].

$$S_J(\omega) = \quad A_\gamma \cdot \frac{5}{16} \cdot H_s^2 \cdot \omega_p^4 \cdot \omega^{-5} \cdot \exp\left(-\frac{5}{4}\left(\frac{\omega}{\omega_p}\right)^{-4}\right) \cdot \gamma^{\exp\left(-0.5\left(\frac{\omega-\omega_p}{\sigma\omega_p}\right)\right)} \tag{7}$$

Where:

$$\omega_p = \quad 2\pi/(T_z \cdot (1.30301 - 0.01698 \cdot \gamma + 0.12102/\gamma)) \tag{8}$$

$$\gamma = \quad \text{non-dimensional peak shape parameter} \qquad = 3.3$$

$$\sigma_s = \quad \text{spectral width parameter}$$

$$\sigma_s = \quad \sigma_a \quad \text{for} \quad \omega \leq \omega_p \qquad = 0.07$$

$$\sigma_s = \quad \sigma_b \quad \text{for} \quad \omega > \omega_p \qquad = 0.09$$

$$A_\gamma = \quad 1 - 0.287\ln(\gamma) \quad \text{normalizing factor} \tag{9}$$

Directional Spreading Since the barge responses and therefore the accelerations encountered are highly dependent on the wave direction, a distinction is made between head and beam waves. During transport and non-operating conditions while moored by one anchor it would encounter mainly beam waves, while as it is positioned next to a bulk carrier or using multiple point mooring, waves can hit the barge from all directions. As the barge has a simple rectangular shape, no distinction is made between head and stern. And while the barge would seemingly encounter waves from one direction, the total wave energy described in the wave spectrum will have a directional spreading [4, 9], with a new spectrum modelled as:

$$S_{zeta}(\omega) = \cos^2(\mu) \cdot S_J(\omega) \tag{10}$$

Wind Coupling Apart from the accelerations of the barge, wind has a significant role in the total load endured by any structure on deck. Realistically, in any storm or weather condition, wind will pick up and will generate more waves adding on to the waves already develloped at a (nearby) location. When the wind dies out, waves still contain a lot of energy and will die out long after the wind has passed. But for simplifying the interlinked environmental conditions and its probability of occurance, the wind speed is set to the maximum speed encountered during such a sea state, according to the Beaufort scale [28, 32]. With a speed considered maximum when the $p_{N.E.}$ is 99%.

8.2 Responses

With a decent description of all sea states and calculated RAOs, the response spectrum ($R(H_s, T_z, \omega)$) per motion or acceleration in any RP can be found according to eq. (11), finding a response spectrum per sea state, directly from the sea state spectrum.

$$R(H_s, T_z, \omega) = \int |RAO(\mu)|^2 \cdot S_\zeta(H_s, T_z, \omega, \mu) d\mu \tag{11}$$

From all the spectra calculated with varying H_s and T_z, the significant response ($r_{1/3}$) can be calculated by using the spectral moment ($m_0(H_s, T_z)$) as seen in eqs. (12) and (13). And accordingly, with a $p_{N.E.}$ of 0.95, the maximum response (r_m) is found by use of eq. (14). [19, 4].

$$\begin{aligned} m_0(H_s, T_z) &= \int R(H_s, T_z, \omega) d\omega \\ &= \cdot \int \int |RAO(\mu)|^2 \cdot S_\zeta(H_s, T_z, \omega, \mu) d\mu d\omega \end{aligned} \tag{12}$$

$$r_{1/3} = 2\sqrt{m_0(H_s, T_z)} \tag{13}$$

$$\begin{aligned} r_m &= \sqrt{-\ln p_{N.E.} \cdot 2 \cdot m_0(T_z) \cdot H_s^2} \\ &= H_s \cdot \sqrt{-\ln(p_{N.E.}) \cdot 2 \cdot m_0(T_z)} \end{aligned} \tag{14}$$

As a result from these calculations, both the $r_{1/3}$ and r_m for any response can be plotted as done with the roll motion and acceleration in RP:0 in figs. 8.2 and 8.3, where the solid line represents a contour line of the significant response and the dashed line the maximum response, both with waves in head direction. Due to the usage of one shape parameter for the all JONSWAP spectra for every seastate, this response scales linearly with H_s, and would not need to be plotted in a contour map. However, the use of such a map simplyfies the use of scatter diagrams, as these can be merged numerically and makes future change in wave spectra modelling possible. All remaining response plots per motion of RP can be found in appendix A

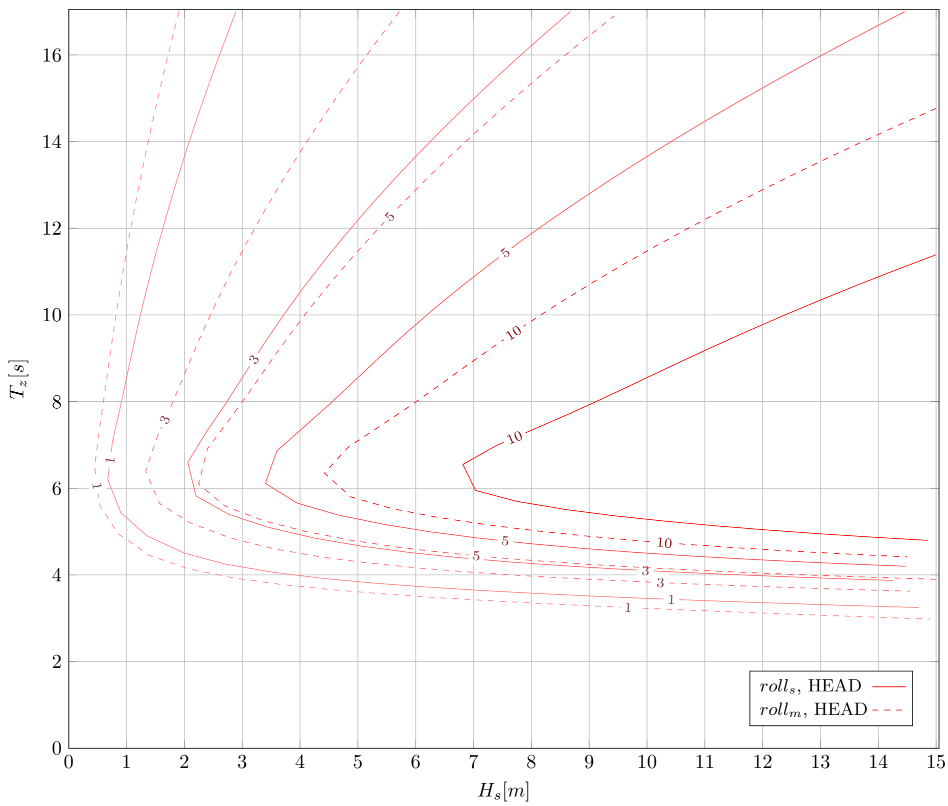


Figure 8.2: Roll responses for head waves

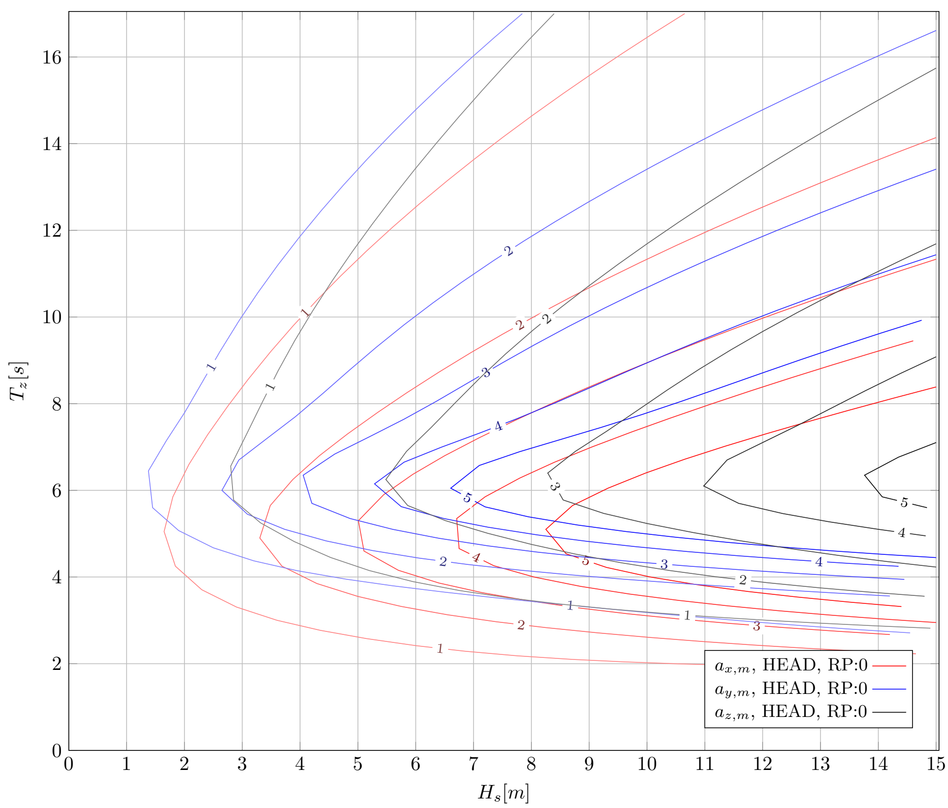


Figure 8.3: Maximum acceleration responses for head waves in RP:0

8.3 Criteria and Restrictions

With all responses calculated and mapped, design criteria, Class rules and structural restrictions regarding the crane and barge can be considered. Most of these restrictions or criteria are directly adapted from LR [21, 29]:

- $\phi^{grab} = 3.00\,\mathrm{deg}$
- $\theta^{grab} = 3.00\,\mathrm{deg}$
- $v_{work}^{wind} = 20.00\,\mathrm{m/s}$
- $v_{wind}^{max} = 63.00\,\mathrm{m/s}$
- $g_{max} = 4.91\,\mathrm{m/s^2}$

Missing a restriction of an acceleration, when the crane driver would endure unpleasant accelerations in the crane driving position [2]:

- $\sqrt{m_0} = 288.00\,\mathrm{mm/s^2}$

And for a stability criterium, the following roll angle is introduced:

- $14.99\,\mathrm{deg}$

The RAOs are calculated with a regular wave with a height of 1 m. When the deck edge submerges, the responses of the barge would change significantly as green water flows on deck.

Lastly, a restriction of maximum stress in the pedestal is added. If the pedestal would fail, the jib rest is allowed to fail as well. The pedestal itself does comply to class regulations, but failure is expected to occur when the barge is in a rough seastate, resulting in high stress cycles in the base of the pedestal. This stress in the pedestal base is calculated as seen in appendix C and a rough estimation of allowable significant stress cycle is set to:

- $\sigma_{fat,ped} = 70.00\,\mathrm{Mpa}$

Comparing these criteria to either the $r_{1/3}$ or r_m maps, a new map can be created, showing a contour line of when this criteria would be exceeded and in what sea state. Resulting in the criteria map for beam waves in fig. 8.4. The most interesting lines here are *motion* - representing the unpleasant accelerations - and *pedestal stress*, because they define the lines between operability (OP), parkability (PA) and removal of the barge.

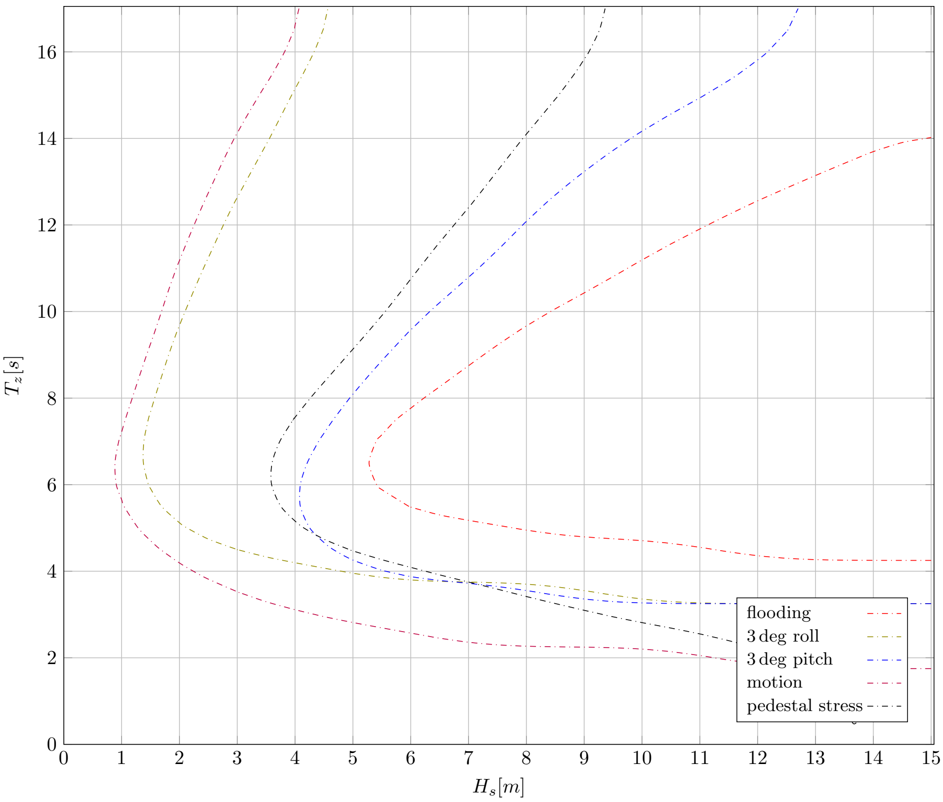


Figure 8.4: Criteria map

8.4 Masking and Operability

The criteria maps, defining the border between OP and PA, can be overlapped by the scatter diagrams of any coastal area the barge is supposed to operate. This creates a operability plot, based on long term wave statistics as seen in fig. 8.5. Here, the hashed area stands for the OP, which is bound by *motion* and the dotted area for PA and bound by *pedestal stress*. The area below these hashes can be summed up to get a total percentage of OP and PA. The legend shows this OP and the total parkability minus the operability as PA.

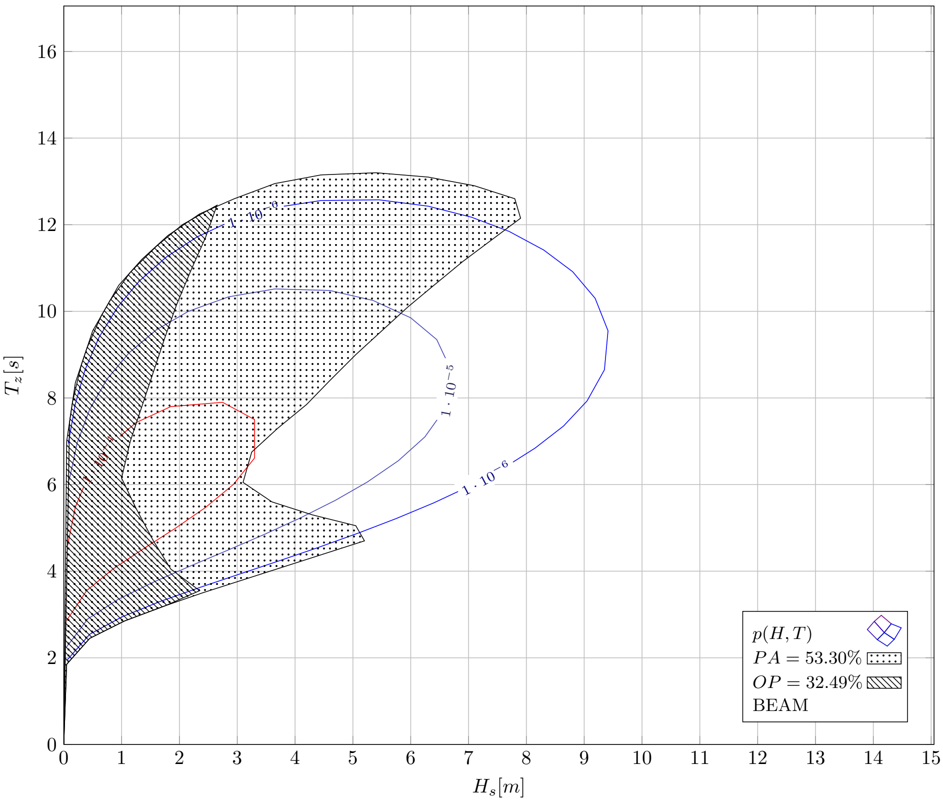


Figure 8.5: Operability map with beam waves

This overlapping of criteria, or masking of boundaries onto the scatter, is done for all nautical areas. The resulting OP (green) and PA (yellow) is shown in pie charts per area in figs. 8.6 and 8.7. Making a difference between head and beam main wave directions. These figures are visually very strong, requiring a recollection of the operability: The operability is calculated using the resposes of the barge itself at open sea, without interaction with external equipment, shore or ships. And with the assumption that no shoaling or sheltering takes place and the crane will not be moving or actually operating, what would affect the heel and trim.

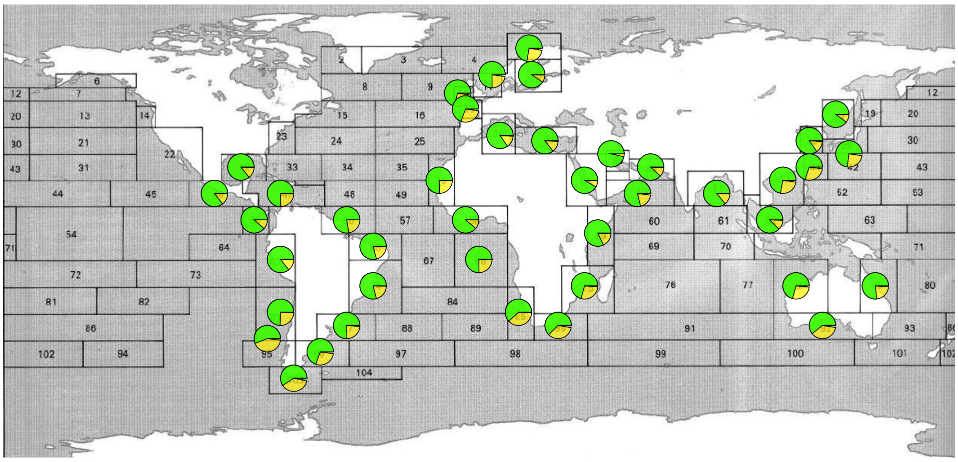


Figure 8.6: Operability map in head waves

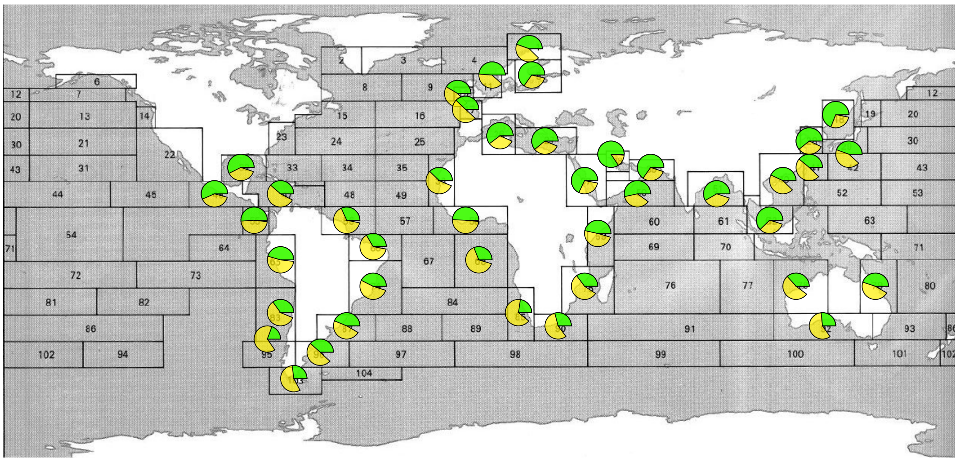


Figure 8.7: Operability map in beam waves

8.5 Probability Encountering Accelerations

Masking can also be done with the scatters and the acceleration response maps. This reveals the part of the scatter that is to be used in finding the maximimum responses. The probability of encountering a maximum response is a combination of the $p_{N.E.}$ of the response in a seastate and the probability of occurance of that seastate []. Simply put, the same accelerations can occur in different seastates, so the product of the probability of a seastate and the response in it can be summed to get a total $p_{N.E.}$. This relation is formulated in eq. (15).

$$p(a_r) = \int\int p(H_s, T_z) \cdot \exp\{\frac{-r^2}{2 \cdot m_0\,(H_s, T_z)}\} dH_s dT_z \qquad (15)$$

This equation is solved by increasing the a_r and checking the sum until the probability drops below the assumed threshold, visualised in fig. 8.8 as different hashed areas.

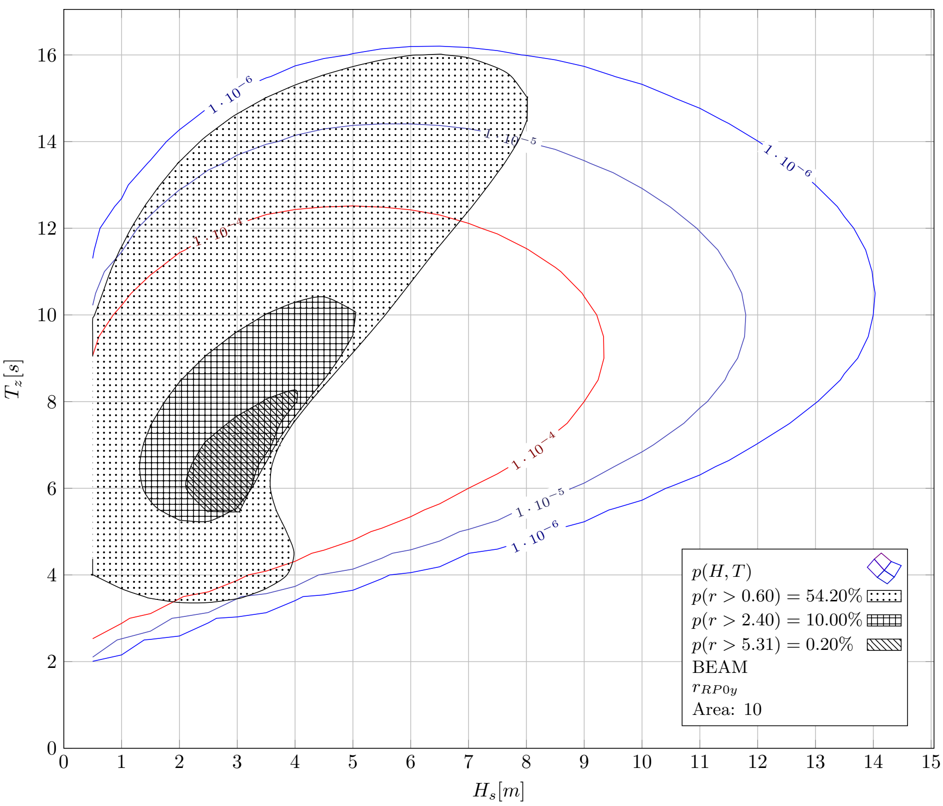


Figure 8.8: Combined $p_{N.E.}$ for area 10, RP:0 and acceleration in y direction

Mainly the maximum accelerations in y and z directions are of interest, and will not occur simultaneously since their phases are different in regard to the phase of the wave. Therefore, a primary direction is selected and calculated according eq. (15) at RP:0, the other directions and other RPs are set as secundary directions calculated with eq. (14), by using the mask from fig. 8.8 and taking the maximum out of that mask. The primary direction probability takes the chance of occuring seastate into account, while for the secundary directions it is presumed that the seastate is already reached. In this case the secundary accelerations are on the conservative side.

To get to a ULS, four states are calculated for every nautical area, from which the highest acceleration is selected as the ULS:

- Y-direction set as primary direction in RP:0, while under load of head waves, defined as load case YH ($a_{RP0,y,head}$)
- Z-direction set as primary direction in RP:0, while under load of head waves, defined as load case ZH ($a_{RP0,z,head}$)
- Y-direction set as primary direction in RP:0, while under load of beam waves, defined as load case YB ($a_{RP0,y,beam}$)
- Z-direction set as primary direction in RP:0, while under load of beam waves, defined as load case ZB ($a_{RP0,z,beam}$)

The solver itself, its programming structure, how it actually handles inputs and provides the output can be found in **??**. Resulting in a set of the four cases, with highest probable acceleration in every RP, the roll and pitch motions, wave heights and wind speeds as seen in tables 8.1 to 8.3. And showing accelerations in x- and y-direction well below 0.50 g (4.91 m/s2) and in z-direction of well below 1 g (9.81 m/s2), as well as roll and pitch motion below 30 deg as LR states as motions to be used to calculate the ULS.

Table 8.1: ULS acceleration results from solver in m/s^2 (part 1)

	$a_{x,RP:0}$	$a_{y,RP:0}$	$a_{z,RP:0}$	$a_{x,RP:1}$	$a_{y,RP:1}$	$a_{z,RP:1}$	$a_{x,RP:2}$	$a_{y,RP:2}$	$a_{z,RP:2}$
YH	1.78	2.46	1.30	1.84	2.78	2.19	0.96	1.74	2.01
ZH	1.54	2.23	1.43	1.60	2.30	2.15	0.82	1.45	2.01
YB	0.95	3.62	1.53	0.98	3.35	1.78	0.46	2.05	1.71
ZB	1.20	3.54	1.66	1.25	3.35	2.02	0.58	2.05	1.93

Table 8.2: ULS acceleration results from solver in m/s^2 (part 2)

	$a_{x,RP:3}$	$a_{y,RP:3}$	$a_{z,RP:3}$	$a_{x,RP:4}$	$a_{y,RP:4}$	$a_{z,RP:4}$	$a_{x,RP:5}$	$a_{y,RP:5}$	$a_{z,RP:5}$
YH	1.50	2.40	2.12	0.88	1.68	2.12	0.32	0.99	2.12
ZH	1.30	1.99	2.02	0.75	1.39	2.02	0.26	0.81	2.02
YB	0.78	3.47	1.93	0.42	2.52	1.93	0.17	1.60	1.93
ZB	0.99	3.47	2.29	0.52	2.52	2.29	0.21	1.60	2.29

Table 8.3: ULS motion and sea state results from solver (part 3)

	Φ [deg]	Θ [deg]	$H_s[m]$	u_w $[m/s]$
YH	8.11	6.22	5	19.90
ZH	7.88	7.12	7	24.50
YB	10.50	2.97	3.50	15.96
ZB	10.50	3.29	3.50	15.96

DIMENSIONING PHASE

9 ULS Load Cases

One way to solely examine the loads and stresses on JR itself, the barge and crane combination can be simplified significantly. And by making use of the accelerations, motions and wind speeds found in section 8, all forces acting on the pedestal, crane and jib can also be simplified and superpositioned upon the JR. Making a distinction between the type of loads and by applying FoSs accordingly, a set of final load cases are defined and a complete ULS analysis can be done. But since the barge will have to comply with Class regulations, the first method defined by LR in section 2 is used to find a different set of loadcases. These load cases are used partially as a reference, but mainly as an ensurance of the useability of the design.

9.1 Model Load Cases

The whole barge, crane and JR configuration can be simplified as a rectangular frame as seen in fig. 9.1 and with only the design of the JR considered as the scope, the system can be simplified even further as in fig. 9.2. In this simplification, the crane masses and construction are distilled to a spring, and the forces acting on it in x direction are transferred to the JR top. To prevent the stowed crane acting as a stiff frame on deck, the JR top and bottom are presumed hinged.

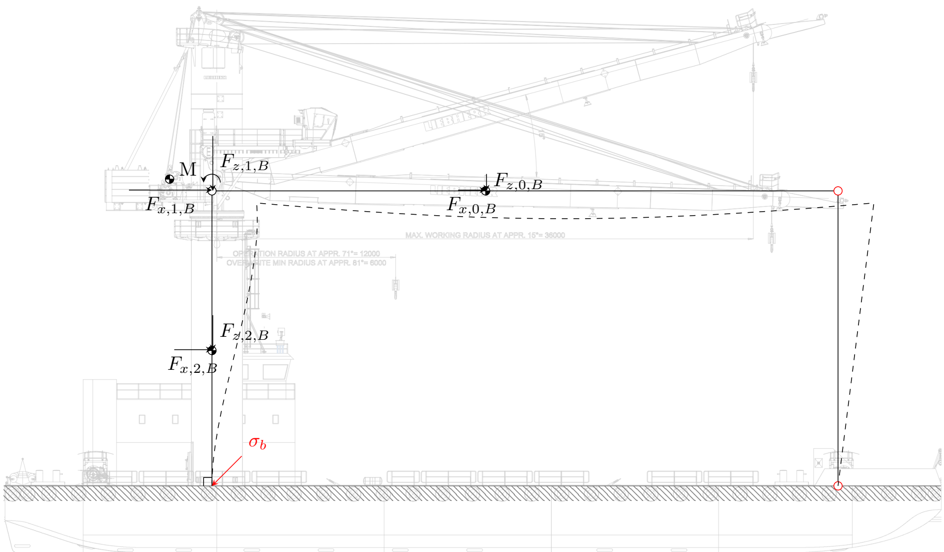


Figure 9.1: Crane barge simplified as a rectangular frame

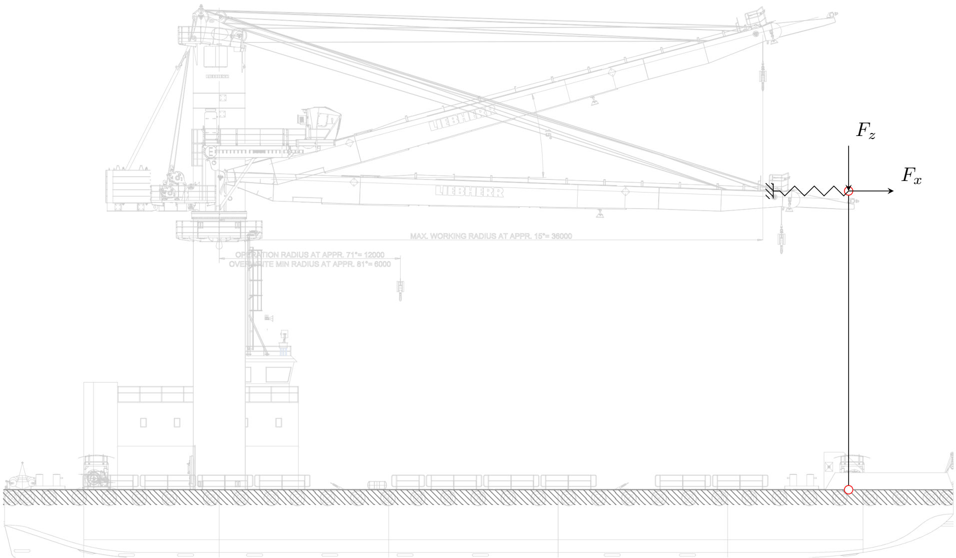


Figure 9.2: Further simplification of the system

With the crane modelled as a spring, the total force acting on the top of the jib with the accelerations found in section 8.5, the forces in x, y and z direction and wind pressures are calculated according:

$$F_x^{YH} = m_{jib} \cdot a_{x,RP0}^{YH} + m_{CT} \cdot a_{x,RP2}^{YH} + 1/2 \cdot m_{ped} \cdot a_{x,RP1}^{YH} = 490.69\,\text{kN} \tag{16}$$

$$F_y^{YH} = m_{jib} \cdot a_{y,RP0}^{YH} \cdot \mu_{p,j} = 42.15\,\text{kN} \tag{17}$$

$$F_z^{YH} = m_{jib} \cdot (a_{z,RP0}^{YH} + g) \cdot \mu_{p,j} + F_{pre} = 235.37\,\text{kN} \tag{18}$$

$$a_x^{YH} = a_{x,RP4}^{YH} = 0.88\,\text{m/s}^2 \tag{19}$$

$$a_y^{YH} = a_{y,RP4}^{YH} = 1.68\,\text{m/s}^2 \tag{20}$$

$$a_z^{YH} = a_{z,RP4}^{YH} = 11.93\,\text{m/s}^2 \tag{21}$$

$$F_w^{YH} = 1/2 \cdot c_j \cdot \rho_{air} \cdot {u_w}^2 \cdot A_{x,z}^{jib} \cdot \mu_{p,j} = 10.21\,\text{kN} \tag{22}$$

$$p_w^{YH} = 1/2 \cdot c_j \cdot \rho_{air} \cdot {u_w}^2 = 0.41\,\text{kN/m}^2 \tag{23}$$

Where:

YH : load case with y as primary direction and barge under beam waves

$\mu_{p,j}$: ratio between loads transferred to jib rest and crane pedestal

F_{pre} : pretension force according to Liebherr (10% SWL) [24]

DNV GL states a ULS analysis should take two sets of design load combinations into account, with different load factors. This FoS is split in a combination where functional and permanent loads are the main focus (a), and one where environmental have the focus (b), resulting in the following factors [6]:

- $\gamma_{f,G,a} = 1.20$: permanent load factor in load combination a
- $\gamma_{f,E,a} = 0.70$: environmental load factor in load combination a
- $\gamma_{f,G,b} = 1.00$: permanent load factor in load combination b
- $\gamma_{f,E,b} = 1.15$: environmental load factor in load combination b
- $\gamma_m = 1.15$: material load factor

And by combining these, the following overall factors are found:

- $FoS_{E,a} = 0.80$
- $FoS_{G,a} = 1.38$
- $FoS_{E,b} = 1.32$
- $FoS_{G,b} = 1.15$

Assuming the forces due to motion and gravity are functional and wind is environmental, the combination of different wave directions, primary acceleration directions and load factors lead to two ULS load cases (table 9.1 and ??).

Table 9.1: Load Case: 6 (YHa)

Fx @JRT	677.15	kN
Fy @JRT	66.38	kN
Fz @JRT	324.81	kN
py	0.41	kN/m^2
ax @JRM	1.21	m/s^2
ay @JRM	2.32	m/s^2
az @JRM	−16.46	m/s^2
FoS	1.00	-
Ref. Stress	235.00	Mpa

Table 9.2: Load Case: 7 (YBa)

Fx @JRT	358.62	kN
Fy @JRT	90.88	kN
Fz @JRT	330.25	kN
py	0.26	kN/m^2
ax @JRM	0.58	m/s^2
ay @JRM	3.48	m/s^2
az @JRM	−16.20	m/s^2
FoS	1.00	-
Ref. Stress	235.00	Mpa

An additional load case in operating or dropped down condition is composed, using only the response maps of RP:5 and the workability criteria and finding the highest response in table 9.3. For an impression on the positioning and the JR, see appendix D.

Table 9.3: Load Case: 8 (DD)

Fx @JRT	358.62	kN
Fy @JRT	90.88	kN
Fz @JRT	330.25	kN
py	0.26	kN/m^2
ax @JRM	0.58	m/s^2
ay @JRM	3.48	m/s^2
az @JRM	−16.20	m/s^2
FoS	1.00	-
Ref. Stress	235.00	Mpa

Where RP:3 is denoted as *@JRT* or jib rest top and RP:4 as *@JRM* or jib rest mid.

9.2 Load Cases Accepted By Class

Again, the conservative method of LR is used to provide a set of load cases to which the JR design has to comply, in order to assure the useability and applicability of the structure and can be found in tables 9.4 to 9.8. Any wind or loads due to acceleration are not simplyfied as one force at the jib rest top, but at their respective COGs, because the pedestal itself is also evaluated in the FEM calculations found in section 12. All details on their locations and denotations can be found in appendix D.

Table 9.4: Load Case: 1 (LRLC1)

m @JIB	39.20	t
m @CT	213.00	t
Fx @JIB	0.00	kN
Px @PED	0.00	kN/m^2
Fx @PT	0.00	kN
Fy @JIB	260.16	kN
Py @PED	1.58	kN/m^2
Fy @PT	125.50	kN
Mxx @PT	582.18	kNm
Myy @PT	0.00	kNm
Mzz @PT	-266.69	kNm
Fz @JRT	-45.00	kN
Px @JRCOG	0.00	kN/m^2
Py @JRCOG	3.92	kN/m^2
ax	0.00	m/s^2
ay	9.81	m/s^2
az	-18.31	m/s^2
FoS	1.56	-
Ref. Stress	235.00	Mpa

Table 9.5: Load Case: 2 (LRLC2)

m @JIB	39.20	t
m @CT	213.00	t
Fx @JIB	0.00	kN
Px @PED	1.58	kN/m^2
Fx @PT	109.09	kN
Fy @JIB	0.00	kN
Py @PED	0.00	kN/m^2
Fy @PT	0.00	kN
Mxx @PT	0.00	kNm
Myy @PT	566.32	kNm
Mzz @PT	0.00	kNm
Fz @JRT	-45.00	kN
Px @JRCOG	3.92	kN/m^2
Py @JRCOG	0.00	kN/m^2
ax	9.81	m/s^2
ay	0.00	m/s^2
az	-18.31	m/s^2
FoS	1.56	-
Ref. Stress	235.00	Mpa

Table 9.6: Load Case: 3 (LRDD1)

Px	0.20	kN/m^2
Py	0.00	kN/m^2
ax	9.81	m/s^2
ay	0.00	m/s^2
az	-18.31	m/s^2
FoS	1.56	-
Ref. Stress	235.00	Mpa

Table 9.7: Load Case: 4 (LRDD2)

Px	0.00	kN/m^2
Py	3.92	kN/m^2
ax	0.00	m/s^2
ay	9.81	m/s^2
az	-18.31	m/s^2
FoS	1.56	-
Ref. Stress	235.00	Mpa

Table 9.8: Load Case: 5 (LRUP)

Px	0.39	kN/m^2
Py	0.00	kN/m^2
ax	9.81	m/s^2
ay	0.00	m/s^2
az	-18.31	m/s^2
FoS	1.56	-
Ref. Stress	235.00	Mpa

10 Dimensioning

This section describes the usage and output of Multiframe (MF), a tool used to define the basic geometry of the concept, built up from a set of nodes, which are connected by members. The programme allows these members to have a section shape and material properties, and by applying loadcases, it calculates forces, moments and stresses in these members or nodes. In every design iteration, section geometry is adjusted to comply with any criteria or assumption, and nodes or members are added, altered or removed accordingly. Its output consists of forces, bending moments, bending and axial stresses, deflection and mass per node or member. Additionally to the static analysis, the software is capable of buckling and nodal analyses. Only the final and old design are relevant and described here.

10.1 Nominal Stress

A set of static analyses are made by applying the load cases to determine the nominal stresses per member in the new JR design. The same is done with the old design as a reference for the fatigue analysis done in section 11. The member numbering can be found in figs. 10.1 and 10.2.

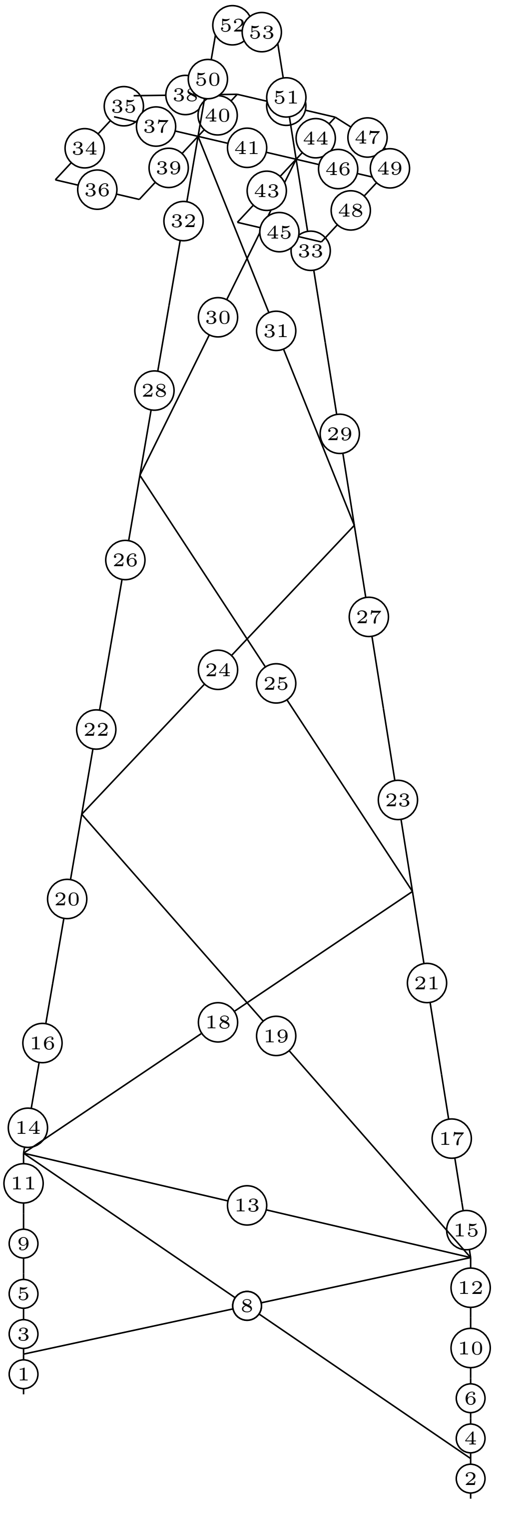


Figure 10.1: Member numbering of the new design

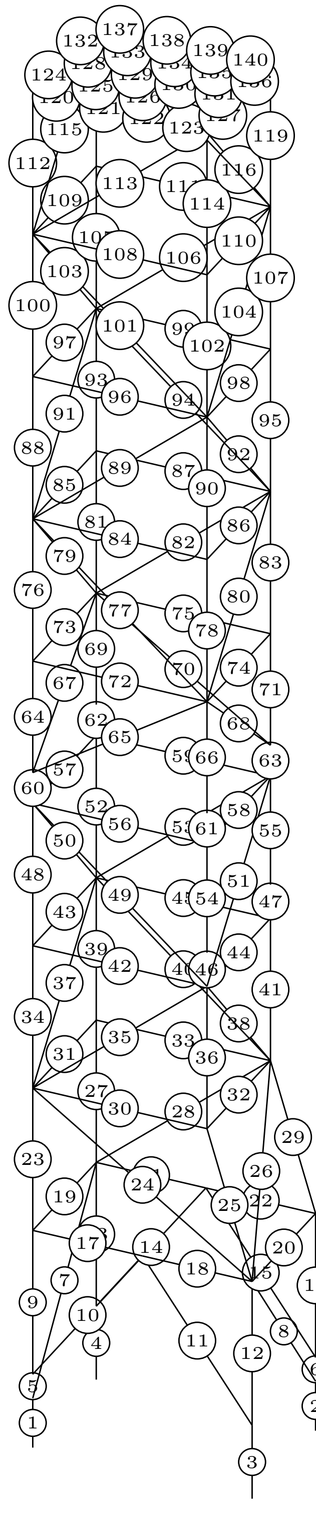


Figure 10.2: Member numbering of the old design

Per iteration, the outputs are parsed in the tool, to determine the maximum stresses in four outer locations on the perimiter of each member. The section and shape is then changed until the stresses are below a range of approx. 100 MPa, set as a first rough estimation of the allowable stress to minimize fatigue damage. The member numbers and their maximum stress levels are described as in fig. 10.3. Showing that all members satisfy the rough stress criterium.

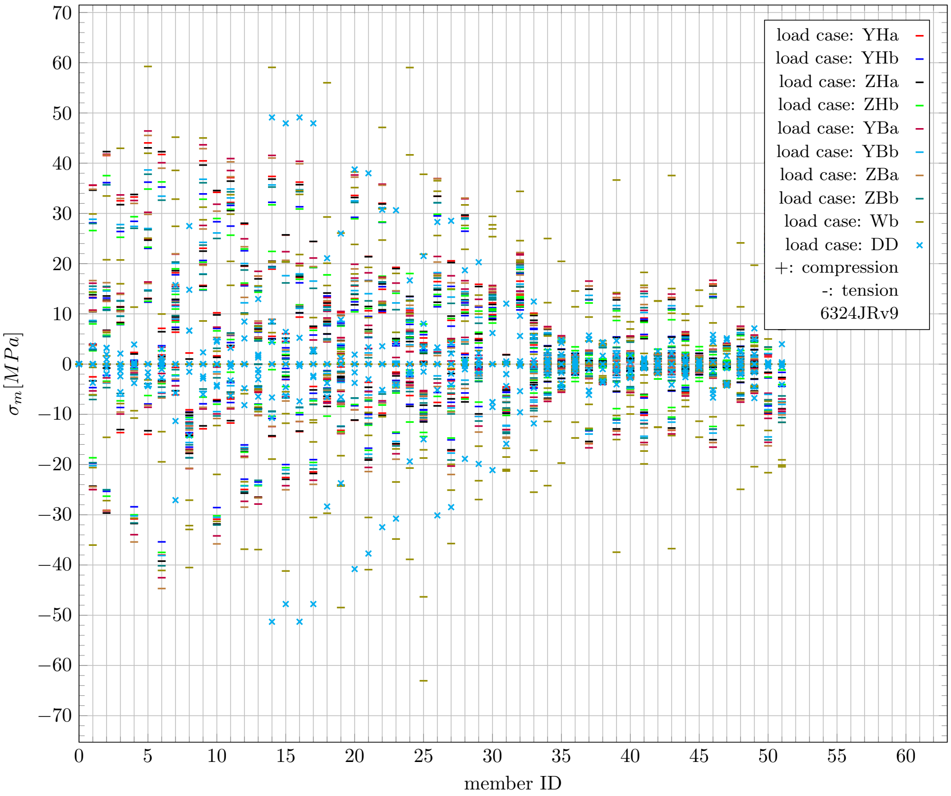


Figure 10.3: Stresses per member

10.2 Buckling

A second step in the iteration is a buckling analysis, shown as a buckling ratio (λ_b) relative to the load applied on top of the JR. In all iteration steps this factor stays well above 3.

Table 10.1: Buckling factor (λ_b) per load case

Load Case	λ_b
YHa	10.70
YHb	11.94
ZHa	11.18
ZHb	12.54
YBa	8.33
YBb	9.42
ZBa	8.59
ZBb	9.69
Wb	6.86
DD	

10.3 Natural Frequency

The last step in the design iteration is the modal analysis of the complete structure. As an additional criterium, P&B has defined a lower boundary of allowable modal frequency (f_r). Frequencies above f_r, will not be significantly generated by waves, engines or any other equipment on board and assures no resonance will occur. Tables 10.2 to 10.4 show the frequencies found for the JR in stowed, upright and free, and dropped down postition, respectfully.

Table 10.2: Natural frequencies in stowed condition

Mode	Freq. [Hz]	Freq. [rad/s]	mod. mass [kg]	mod. mass [%]
1.0	5.14	0.31	1547.06	0.10
2.0	7.31	0.21	832.07	0.05
3.0	7.42	0.21	1468.74	0.10
4.0	9.46	0.17	3893.15	0.25
5.0	12.67	0.12	689.31	0.04
6.0	12.73	0.12	662.09	0.04
7.0	16.61	0.09	2978.04	0.19
8.0	17.42	0.09	109.47	0.01
9.0	17.72	0.09	1017.57	0.07
10.0	18.16	0.09	33.26	0.00

Table 10.3: Natural frequencies in operating, upright position

Mode	Freq. [Hz]	Freq. [rad/s]	mod. mass [kg]	mod. mass [%]
1.0	2.44	0.64	2353.93	0.15
2.0	4.79	0.33	1712.88	0.11
3.0	7.31	0.21	826.75	0.05
4.0	7.42	0.21	1556.06	0.10
5.0	12.19	0.13	1237.98	0.08
6.0	12.54	0.13	826.03	0.05
7.0	13.33	0.12	1886.90	0.12
8.0	14.60	0.11	609.82	0.04
9.0	17.01	0.09	141.00	0.01
10.0	17.73	0.09	1018.22	0.07

Table 10.4: Natural frequencies in operating, dropped down position

Mode	Freq. [Hz]	Freq. [rad/s]	mod. mass [kg]	mod. mass [%]
1.0	4.37	0.36	1709.18	0.11
2.0	5.36	0.29	1264.17	0.08
3.0	7.32	0.21	821.01	0.05
4.0	9.54	0.16	921.76	0.06
5.0	12.61	0.12	431.82	0.03
6.0	12.62	0.12	703.44	0.05
7.0	14.69	0.11	1331.40	0.09
8.0	17.69	0.09	1065.47	0.07
9.0	17.89	0.09	36.92	0.00
10.0	18.26	0.09	32.57	0.00

During the design iterations, the JR is stiffened up significantly in x-direction, to get the natural frequency above f_r, mainly in the dropped down condition. The free upright position is disregarded in this case, since bringing the frequency up would mean too much addition of stiffness by changeing the complete design. Additionally, this position would only occur when transitioning from stowed to operating condition.

10.4 Weight

In each cycle, the mass of the total construction is checked to be below the mass of the old JR. And in the last cycle, the total mass of the JR is approximated on $m_{JR} = 14.46\,\text{ton}$, includeing 2 ton additional weight due to stairs and other structural details. This is a couple of tonnes below the benchmarked mass of $= 19.40\,\text{ton}$.

DETAILING PHASE

11 Fatigue

The fatigue cracks found in the first generation of the JR is found to be originated in the stiff ans slender design. This causes high stress concentrations or hotspots in and near joints of the K-frame. During the transport, these hotspots are loaded cyclically allowing existing cracks or imperfections to grow until failure occurs. Making an accurate estimation of this crack growth and consequentially the fatigue life estimation is dependent on many factors like the number and amplitude of load cycles, compressive or tensional stress, mean stress, initial crack size, shape, material properties and temperature [1, 7, 25, 11, 13].

Different approaches are used to determine fatigue life, either based on cumulative fatigue damage theory (CFD) - like rainflow counting in combination with the Palmgren-Miner rule - or in combination with fatigue crack propagation theory (FCP) [20]. Commonly, stress concentrations are represented as a stress concentration factor (SCF) and a local nominal stress [13] and its cycles are determined by the load cycle history or time-domain simulation, which would need detailed history data or extensive simulations. Dirlik proposed an emperical model which provides the rainflow ranges using the spectral moments of an expected stress spectrum [18, 12] and its formula can be found in section 11.3.

11.1 FAT Codes and SN-Curves

At this design stage, only the basic shape and dimensioning is set up for the JR. The detailed construction is as recursively dependant on any stress concentration, as it is on the ULS. A prediction on the nominal stresses in the members can be made, which are dominated by the loads and dimensioning of the individual members and any connection between the members is dominated by SCFs and the nominal stresses. For a large number of different (welded) connections, the allowable nominal stress is standardized into stress cycle curves (S-N curves) and fatigue class (FAT) proposed by International Institue of Welding (IIW) [13], where any SCF is taken into account. For every connection or beam element, a type of standard connection is selected, and its FAT or S-N curve used as reference for further fatigue analysis.

11.2 Stress Spectra

Nominal stresses found in the JR depend on individual member geometry and loads. Assuming the JR will be double hinged on both top and bottom, it will only be able to withstand loads in lateral and vertical directions. This implies that the structure will only need to withstand load cycles in those directions, presuming the stiffness of the hinge is negligible in comparison to the stiffness of the jib rest itself. By applying an acceleration of one in one of the directions, ratio between acceleration and stress (γ_σ) can be found per member per direction. This ratio is used as a transfer function, to get a probable stress spectrum per area according eq. (24) and plotted in fig. 11.1.

$$S_\sigma = |\gamma_\sigma|^2 S_a \tag{24}$$

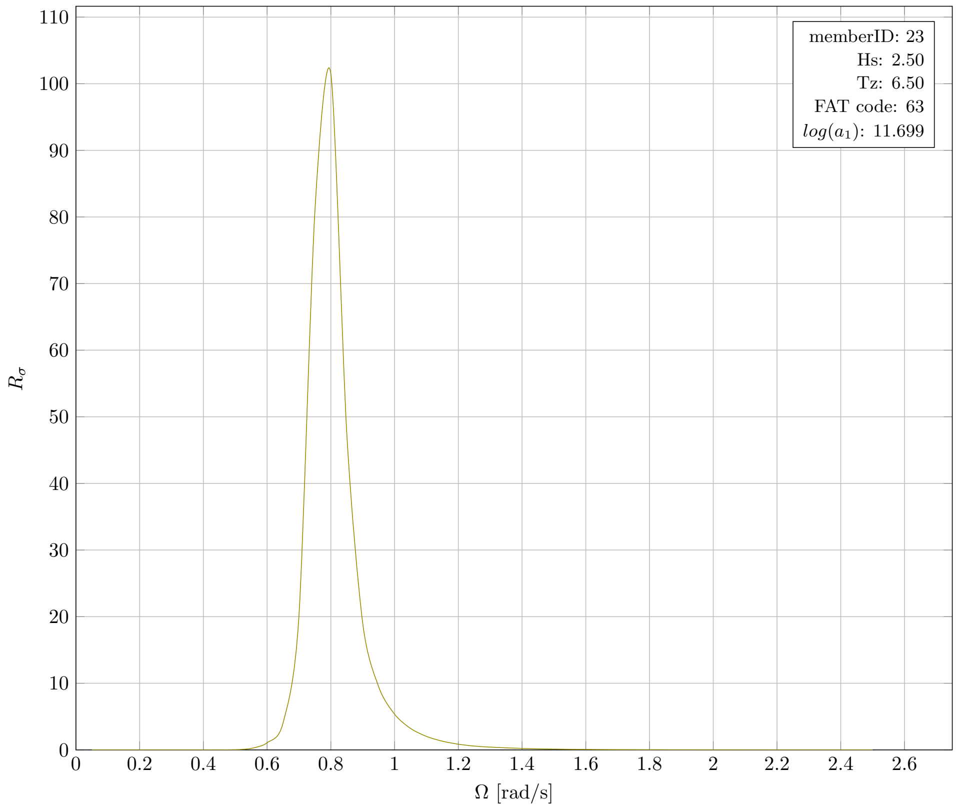


Figure 11.1: Stress spectrum in member no. 23 due to lateral acceleration

11.3 Fatigue Lifetime Estimation

Dirlik's has developed an emperical model to estimate the stress cycles encountered during as structure's lifetime, using the a broad band load spectrum [12]. Its formula, using the spectral moments, representing the number of cycles per stress range is given in eq. (25) [12].

$$N(S) = E[P] \cdot T \cdot p(S) \tag{25}$$

Where:

$$p(S) = \frac{\frac{D_1}{Q} \cdot e^{\frac{-Z}{Q}} + \frac{D_2 \cdot Z}{R^2} \cdot e^{\frac{-Z^2}{2R^2}} + D_3 \cdot Z \cdot e^{\frac{-Z^2}{2}}}{2\sqrt{m_0}}$$

$$D_1 = \frac{2(x_m - \gamma_d^2)}{1 + \gamma_d^2}$$

$$D_2 = \frac{1 - \gamma_d - D_1 + D_1^2}{1 - R}$$

$$D_3 = 1 - D_1 - D_2$$

$$Z = \frac{S}{2\sqrt{m_0}}$$

$$Q = \frac{1.25 \cdot (\gamma_d - D_3 - D_2 \cdot R)}{D1}$$

$$R = \frac{\gamma_d - x_m - D_1^2}{1 - \gamma_d - D_1 + D_1^2}$$

$$\gamma_d = \frac{m_2}{\sqrt{m_0 \cdot m_4}}$$

$$x_m = \frac{m_1}{m_0} \cdot \sqrt{\frac{m_2}{m_4}}$$

$T :$ time in seconds

At this detailing phase of the design, this formula is used for every member in every seastate, resulting in a histogram showing the expected cycles per seastate. Figure 11.2 shows the expected cycles for one of the members in the new JR design.

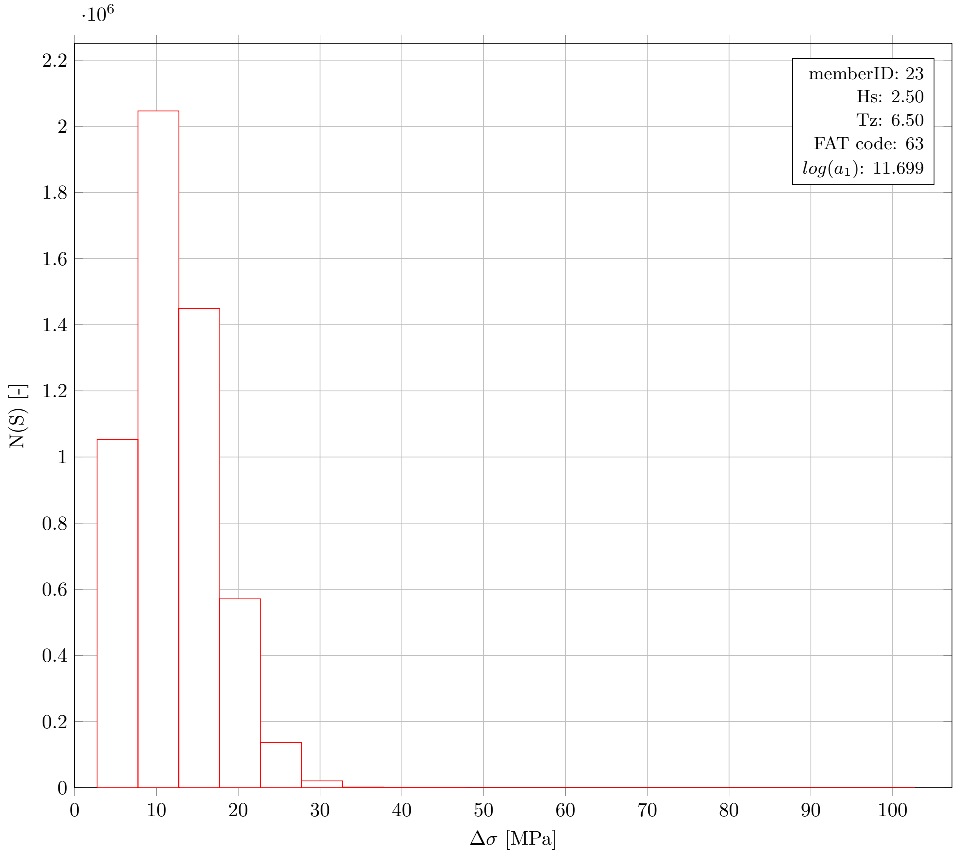


Figure 11.2: Expected cycles per stress range per year

The cycle estimation can directly be multiplied by the $p_{N.E.}$ per sea state and the probability of encountering such a sea state from the scatters. Taking the duration of a towing trip and its passed nautical areas from the voyage reports, and cumulating the damage using the Palmgren-Miner rule, an estimation of surpassed lifetime can be made. At this point, the accuracy of the total probability is highly debatable since this model uses the long term probability of a seastate and would not be very representable for a short term tow, especially since weather and tow conditions are monitored day by day.

As one case study, a rough towing trip is considered as a voyage of two weeks, in which $Hs_{tow}^{max} = 2.30\,\text{m}$ as from the reports and $T_{\lambda=l} = 6.35\,\text{s}$ as the response in head waves would be the highest in waves with this period. In this case, the two designs are subjected to such a tow and the expected damage per member is calculated. The old design shows significant damage in almost all of the vertical members, in both wave directions. Several would even exceed their allowable expected cycles by two or three times. The expected damage of the new design is a lot lower, with some members reaching approx. 7% of their total cycles. More detailed data on the cycles and estimations can be found in appendix B, along with two other fatigue cases, simulating a lifetime in or near one nautical area.

VERIFICATION PHASE

12 Final Design and Design Verification

As a last step, the complete detailed design is drawn with precise dimensioning of the main elements, such as the members, flanges and stiffners, and rough dimensioning of the hinges, connecting eyes and secondary additions to the structure. The MF basic dimensioning, like node coördinates and section data is exported to Autocad and accompanied by any further data, making sure the materials with those dimensions are standardized and available worldwide. This results in the final design shown in fig. 12.1. Any further data, drawings or calculations made to get to this design can be found in appendix D.

Figure 12.1: Drawings of the final design (6324JRv9) *Blurred because it is considered classified*

Using these final dimensions, a FEM model is made in Siemens NX (NX) by Marine Design Engineering Mykolayiv (MDEM) and with the added loadcases defined in section 9 used as a verification of the ULS. Any minor adjustments to the structural detail of the design are done, mainly by looking at the peak stresses as shown in an example in fig. 12.2 and the natural frequencies as in an example shown fig. 12.3.

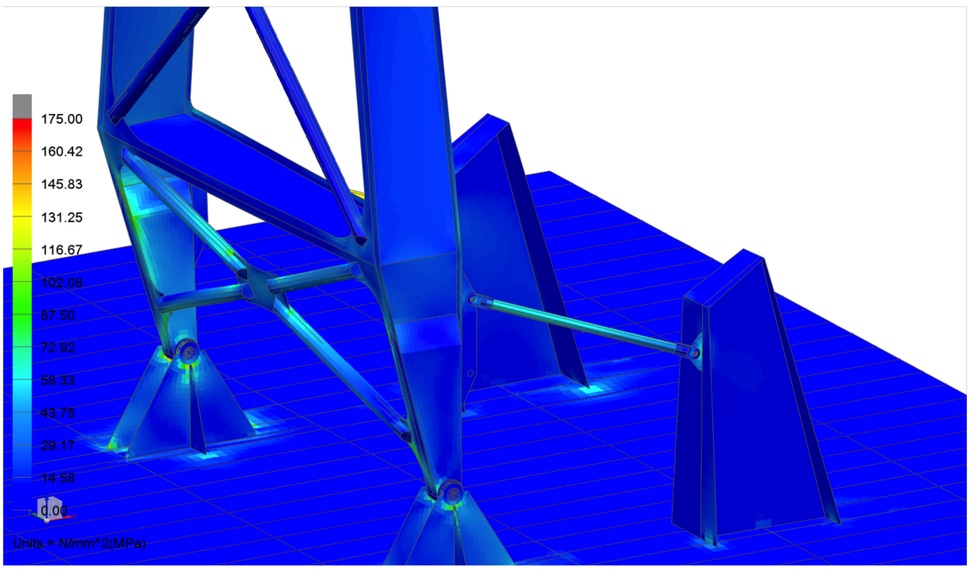


Figure 12.2: Stresses in bottom part

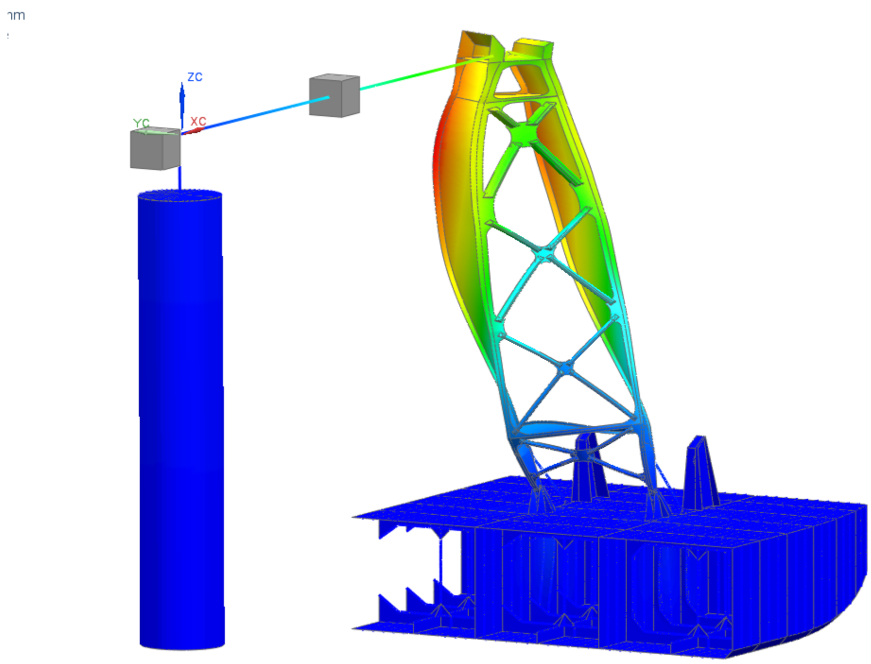


Figure 12.3: Natural frequency analysis

13 Conclusions

13.1 Discussion

13.1.1 Model Transparency

The barge is developed by combining a standard pontoon, standard crane and additional equipment. Using the standardized components enables an efficient and fast development, but also introduces a need for transparency, as standardization of the seperate products could bring in different Class regulations and restrictions. Both the crane and pontoon are subjected to extreme design conditions, but their interation may bring up a different insight on the overall design or even the applicability of the Class regulations. Identifying a bottle-neck in the design and knowing where its dimensioning or choice is originated, speeds up the development and optimisation. For example: If the average operability in an area is well below optimum, should the barge still be designed for this area? And one used as a criteria in this thesis: In what extreme sea state is the crane pedestal expected to fail, and should the JR be designed to withstand even higher accelerations? As standardization is one of the key design phylosophies, it is beneficial to see what effect addition or adaptation of components on the barge. Depicting this in a graphically per sea state and allowing quick adjustments provides a fast way of gaining insight on the overall design.

13.1.2 Sea States

To get a more detailed idea about the forces the JR has to withstand, more detailed information about the location of operation is needed. Detailed information on local weather and sea states to get to a more accurate estimation on the loads. Knowing the loads are highly dependant on the location where the barge would need to operate, and a design criterium of worldwide operability, suggests that detailed information of all around the world would need to be analysed. As analysing all areas would be a difficult task, a representation of all sea states is chosen to be a JONSWAP-spectrum. These spectra are the base of finding the maximum responses and are considered to be sufficient for finding a reasonable worldwide ULS. In some areas and in some seastates, this JONSWAP-spectrum would not be the best fit, and it would be interesting to see if and how a different spectrum would result in different accelerations, and consequently a different ULS.

13.1.3 Worldwide Operability

Using the long term wave statistics and responses as a determination on operability, the barge shows promise in near coastal areas just above the equator and more sheltered seas, as long as the main wave direction are head waves. In this operability determination, only the responses of the barge itself are used, without interaction with other vessels, mooring or sheltering. If the barge would be moored in between a bulk carrier and feeder vessel, or would be transhipping in a harbour, this operability would increase. P&B states an ideal operation time would be around 90% and could be met in the more sheltered conditions. The operability seems to decrease below the equator and the barge would be less usable in near coastal areas around Argentina, South Africa and Australia, looking at a yearly average.

The first restricting condition in this operability analysis are the accelerations in the crane top, at the drivers position, ensuring comfortable operation. The following restriction is the allowable roll dictated by Class. According to this regulation, the crane will enter a safety mode when this angle is met. In this analysis, only responses of the barge are used to calculate the accelerations and roll or heel angles, and rotating and operation of the crane would add to these responses, reducing the operability.

13.1.4 Load Cases

As seen in section 9, there is a significant difference between the load cases found following Class and the load cases found using the method described in section 8. Lower loads are expected in the latter, because the more conservative method of LR is based on extreme weather conditions and the one proposed in this report takes more detailed information on the barge and its operational locations into account. Accepting the assumptions and simplifications of the more detailed method, would mean a large decrease in JR size. Maybe other structures on deck could be downsized as well. If this method is not accepted, it still shows the benefit of an alternative calculation method, especially when weight reduction is one of the design criteria.

13.1.5 Jib Rest Design

A JR design is found, which folds down, and is lighter than the design of the first generation. The frequency analysis showed a need to increase the stiffness of the frame. The criteria of f_r for an eigenfrequency is originated in standards within Damen, to stay well away from frequencies induced by waves, propellor or engine. The applicability of this standard should be investigated more thoroughly, because the barge will not be engine propelled, and allowing a lower frequency would result in a less stiff frame and more weight reduction.

13.1.6 Pontoon Dimensions

The preliminary calculations show relatively low deformations (section 4.1), even in waves of approx. 4 m. In further calculations the deck is considered to have an infinite stiffness. This allows simplification of the modeling of the barge, crane and JR, but also shows a hull construction that might be too stiff for its application. If the deck load prequisites and further analysis would allow this, there is a large potential for weight reduction.

13.1.7 Fatigue Lifetime Estimation

The estimation on fatigue lifetime is based on the stress spectra in every member by using an acceleration in either the y or z direction. The cumulative damage is then calculated by adding the stress cycles of both directions. Since the accelerations of in both directions are not coupled - neither phases are taken into consideration - this gives a rough estimation. A better estimation can be found by relating the two directions and by summing the stress levels instead of their cycles. The relating of the accelerations would need an introduction of a relation of the two directions by using the reponse phases. An other way to add detail to this estimation is by taking the effects of mean, tensional and compressive stresses into account.

13.2 Conclusions

LR provides a conservative method to determine loads acting on a structure on a floating crane, based on extreme values. As specific data on the CB6324 is available, a different method is proposed, using motion responses, sea state data and criteria as an input. This method is based on probability of encountering a sea state, the probability of non-exceedance of a wave in such a state and the operability, to predict accelerations. For this, a tool is developed, which automates the calculations for the maximum expected load in worldwide near shore operation and transport. It takes wave data, direction and spreading, wind speed, roll, pitch, flooding angle and pedestal (bending) stresses into account to find these loads. Therefore, objective item A is considered reached. The loads are as expected significantly lower than found by applying the method proposed by LR.

A new JR design follows from the load cases found with the proposed method. Peak stress levels in this design are well below the yield stress. The modal frequency in two of the three positionings comply with the criteria, being above $f_r = 4.00\,\mathrm{Hz}$. The buckling factor per member are all well above $\lambda_b = 3.00$. The JR is placed or can move out of the working zone and the total mass of the construction is reduced by $\Delta m = 4.94\,\mathrm{ton}$, making the JR comply to the main design criteria.

The tool is extended with a fatigue prediction component, which again uses the sea state data plus the geometry of the JR. The old design showed fatigue crack growth within two weeks in rough weather, as well as in the prediction as in the actual situation twice. The new is predicted to have spent a maximum of 7% of its fatigue lifetime.

References

[1] M. El-Zein J. Qian A. Chattopadhyay, G. Glinka and R. Formas. Stress analysis and fatigue of welded structures. *Welding in the World*, 55(07-08), 2011. 53

[2] Michael J. Griffin Barbara M. Haward, Christopher H. Lewis. Motions and crew responses on an offshore oil production and storage vessel. *Applied Ergonomics*, 40, 2009. 39

[3] Sea Contractors. email daily position reports, 2014. 20

[4] DNV. *DNV-RP-C205: Environmental Conditions and Environmental Loads*. Det Norske Veritas AS, 2010. 4, 5, 8, 35, 36, 37

[5] DNV. *DNV-OS-C102: Structural Design of Offshore Ships*. Det Norske Veritas AS, 2011. 4, 5

[6] DNV. *DNV-OS-C101: Design of Offshore Steel Structures, General (LRFD Method)*. Det Norske Veritas AS, 2014. 46

[7] Norman E. Dowling. Mean stress effects in stress-life and strain-life fatigue. 2004. 53

[8] T.H.J. Bunnik E.F.G. van Daalen. $Precal_RTheoryManual.Marin, 2014.32$

[9] Michael C. Johnson Elzbieta M. Bitner-Gregersen, Keven C. Ewans. Some uncertainties associated with wind and wave description and their importance for engineering applications. *Ocean Engineering*, 86, 2014. 37

[10] C. Graham. The parameterisation and prediction of wave height and wind speed persistance statistics for oil industry operational planning purposes. *Coastal Engineering*, 6, 1982. 5

[11] P. J. Haagensen and S J. Maddox. *IIW Recommendations on Post Weld Improvement of Steel and Aluminium Structures*. The International Institute of Welding, 2001. 53

[12] Andrew Halfpenny. A frequency domain approach for fatigue life estimation from finite element analysis. 1999. 53, 54

[13] A. Hobbacher. *Recommendations for Fatigue Design Of Welded Joints and Components*. The International Institute of Welding, 2008. 53

[14] Hugo Hoekstra. Mom toekomstvisie crane barge. Technical report, Damen, 2014. 5

[15] Leo H. Holthuijsen. *Waves in Oceanic and Coastal Waters*. Cambridge University Press, Cambridge CB2 8RU, UK, 2007. 5, 14, 32

[16] Mikael Huss. Notes on the modeling of irregular seas in time simulations. 2010. 36

[17] Masahiko Isobe. On joint distribution of wave heights and directions. 1988. 35

[18] Xinzhong Chen Jie Ding. Fatigue damage evaluation of broad-band gaussian and non-gaussian wind load effects by a spectral method. *Probabilistic Engineering Mechanics*, 41, 2015. 53

[19] J.M.J. Journée and W.W. Massie. *OFFSHORE HYDROMECHANICS*. Delft University of Technology, Delft, 1 edition, 2001. 32, 36, 37

[20] Kyung-Su Kim Jun-Bum Park, Joonmo Choung. Fatigue damage evaluation of broad-band gaussian and non-gaussian wind load effects by a spectral method. *Ocean Engineering*, 76, 2014. 53

[21] Liebherr. *CBG 350 Technical Data Floating Crane Damen Shipyards-Liebherr Floating crane*. 20, 39

[22] Burcharth H. F. Liu, Z. Encounter probability of significant wave height. 1998. 35

[23] Constantine D. Memos. On the theory of the joint probability of heights and periods of sea waves. *Coastal Engineering*, 22, 1994. 35

[24] Mertins. *Reactions to foundation at boom rest*. Liebherr, 10 2010. 46

[25] Jan Zuidema Michael Janssen and Russell Wanhill. *Fracture Mechanics*. Spon Press, 2 Park Square, Milton Park, Abingdon, Oxfordshire, OX14 4RN, 2 edition, 2004. 53

[26] Muller. email daily position reports, 2015. 20

[27] NEN. *NEN-EN 13001-1*. Nederlands Normalisatie Instituut, 2015. 4

[28] Met Office. National meteorological library and archive fact sheet 6 the beaufort scale. 2010. 37

[29] Lloyd's Register. *Code for Lifting Appliances in a Marine Environment*. Lloyd's Register Group, 2013. 4, 39

[30] Mohamed Shama. *Torsion and Shear Stresses in Ships*. Springer, Berlin Heidelberg, 2010. 13

[31] Carlos Guedes Soares. Probabilistic models of waves in the coastal zone. *Advances in Coastal Modeling*, 2003. 35

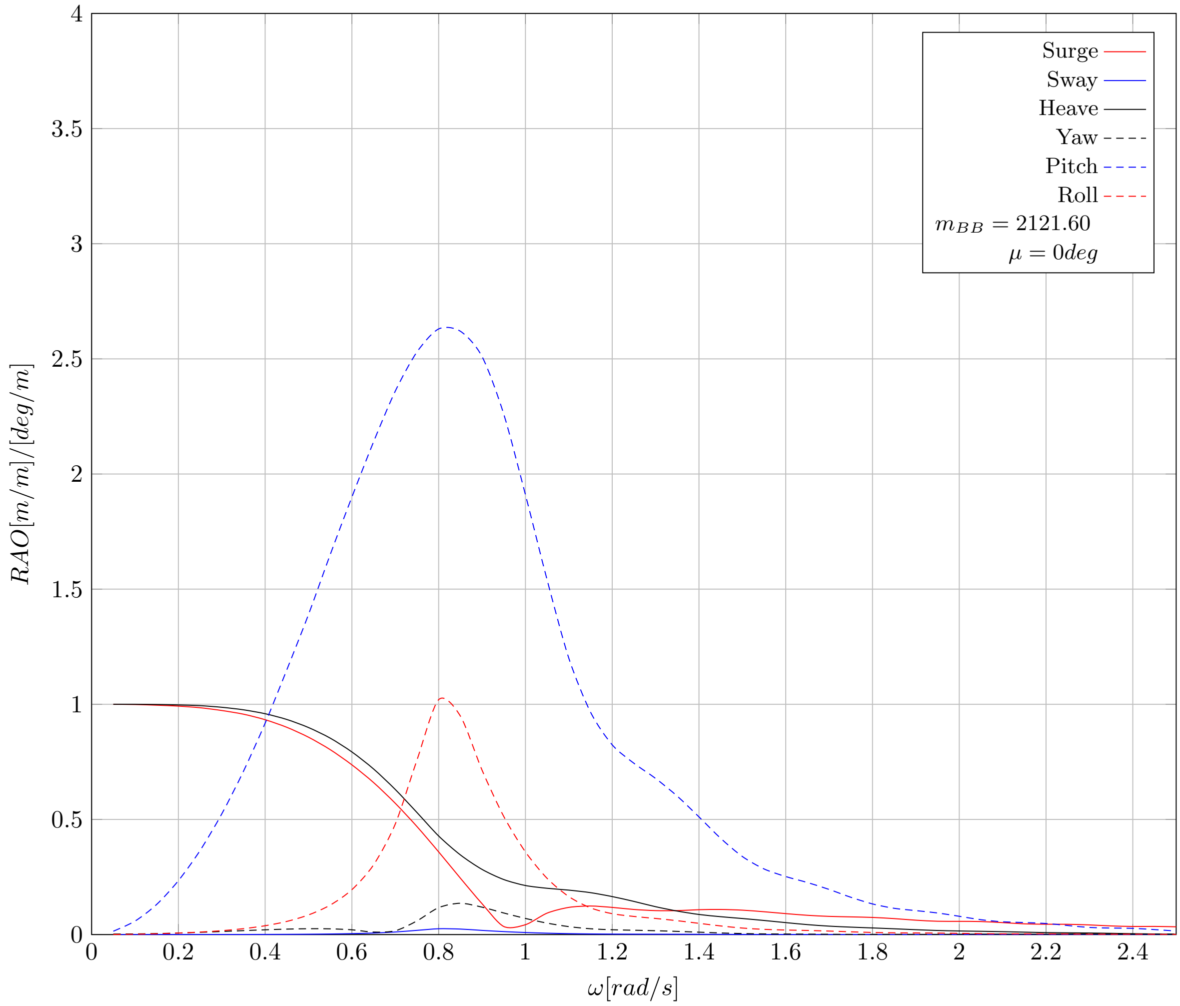
[32] Eric Tupper. *The Ship Environments*. Elsevier, 2013. 37

A Responses

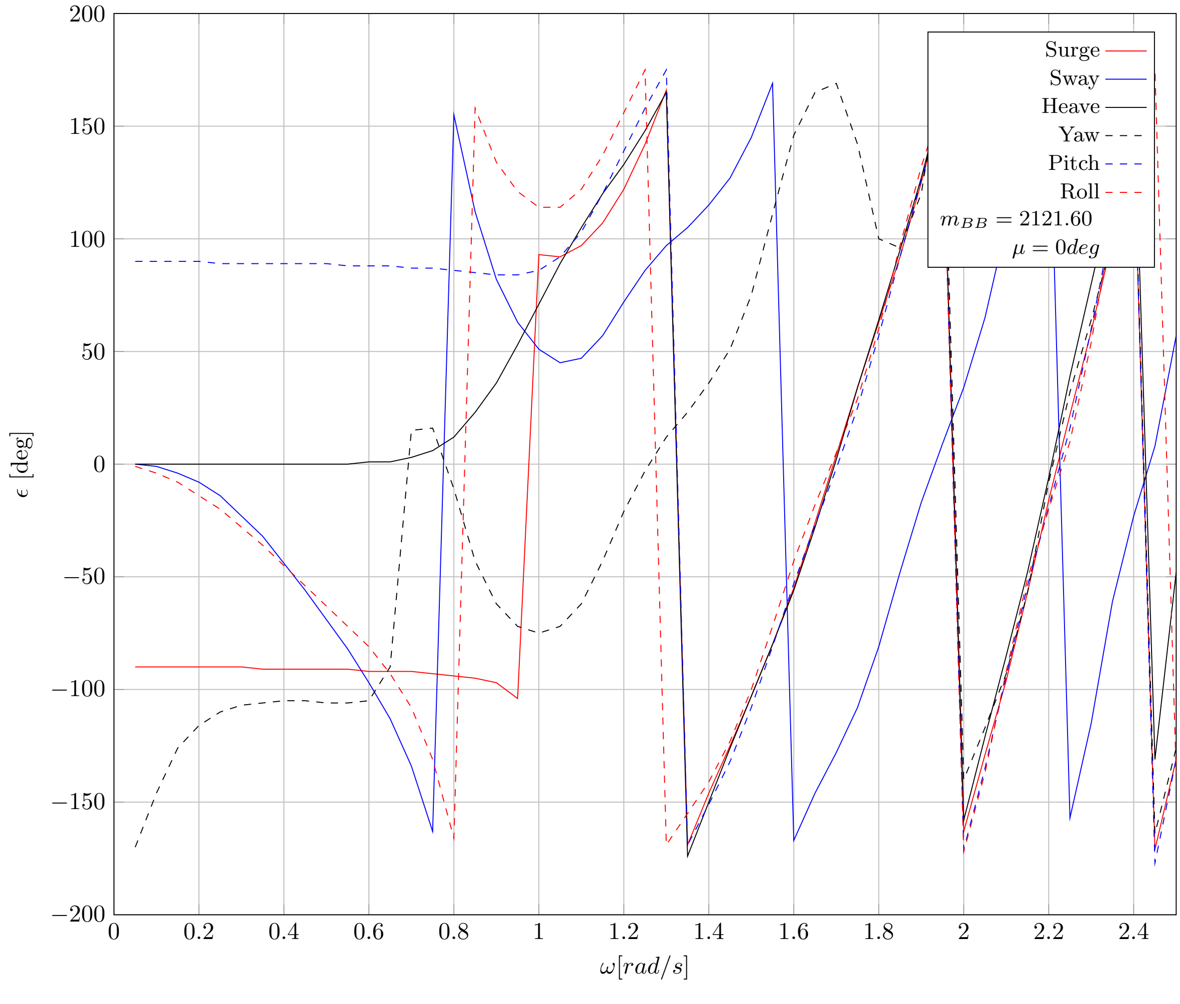
The following pages show the graphes relating to the barge motion responses found using the methodology described in section 2, in the following order:

- RAOs
- Phases
- Criteria
- Motion responses calculated with head waves as main direction
- Motion responses calculated with beam waves as main direction

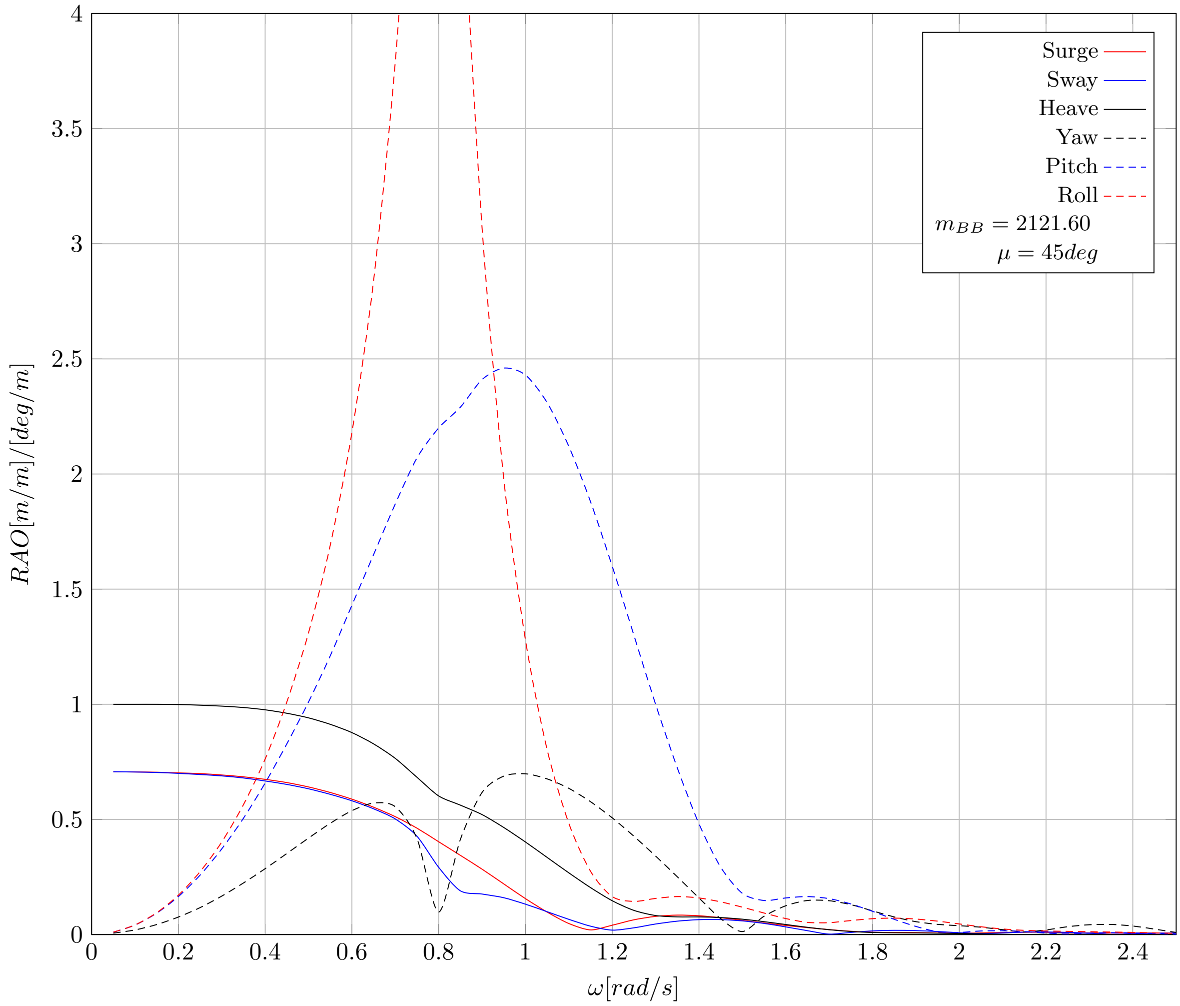
In these graphs, roll, pitch and yaw are in [deg], accelerations in [m/s^2] and stresses in [MPa]



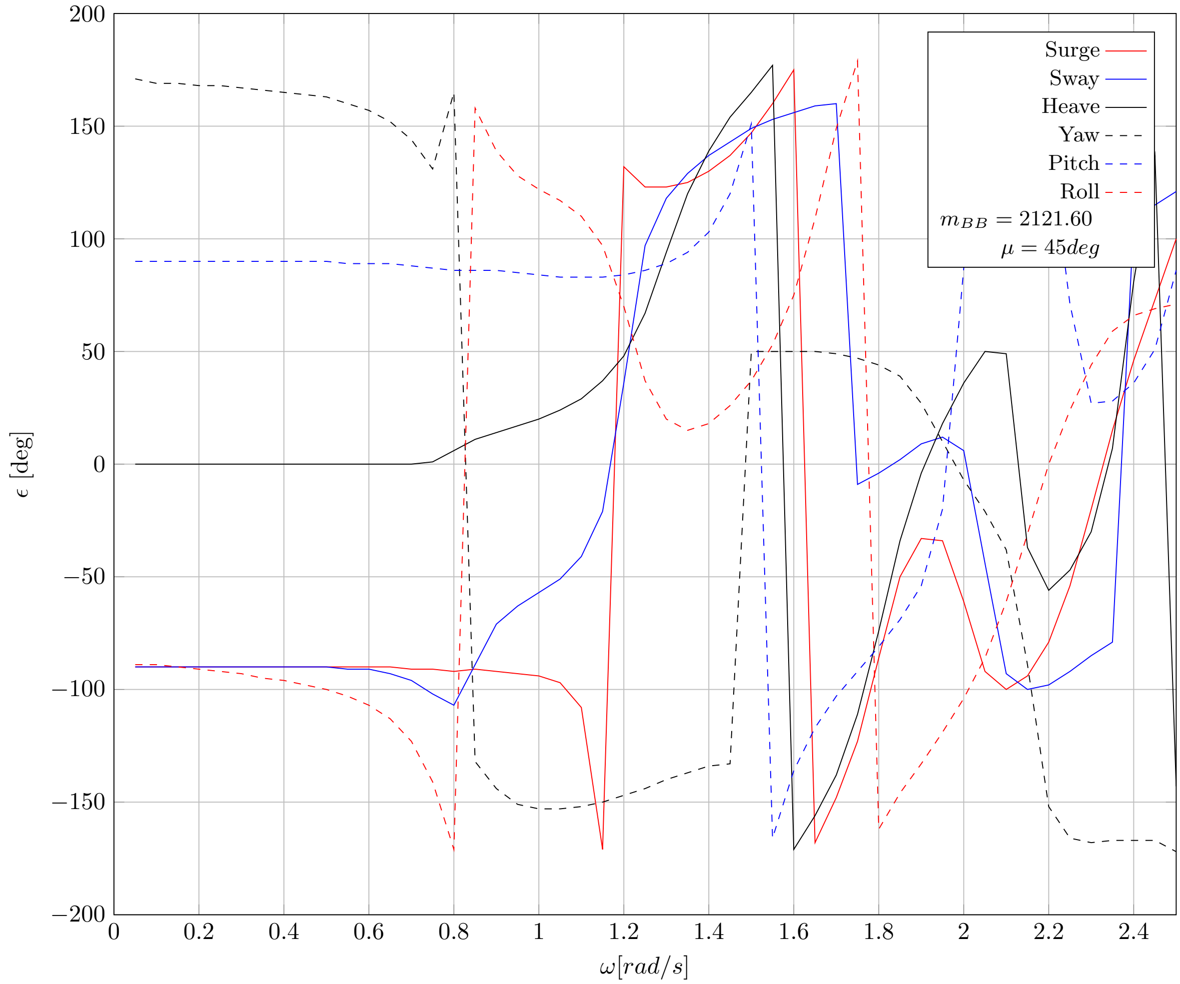
Surge
Sway
Heave
Yaw
Pitch
Roll
$m_{BB} = 2121.60$
$\mu = 0deg$
$RAO[m/m]/[deg/m]$
$\omega[rad/s]$
0
0.5
1
1.5
2
2.5
3
3.5
4
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4



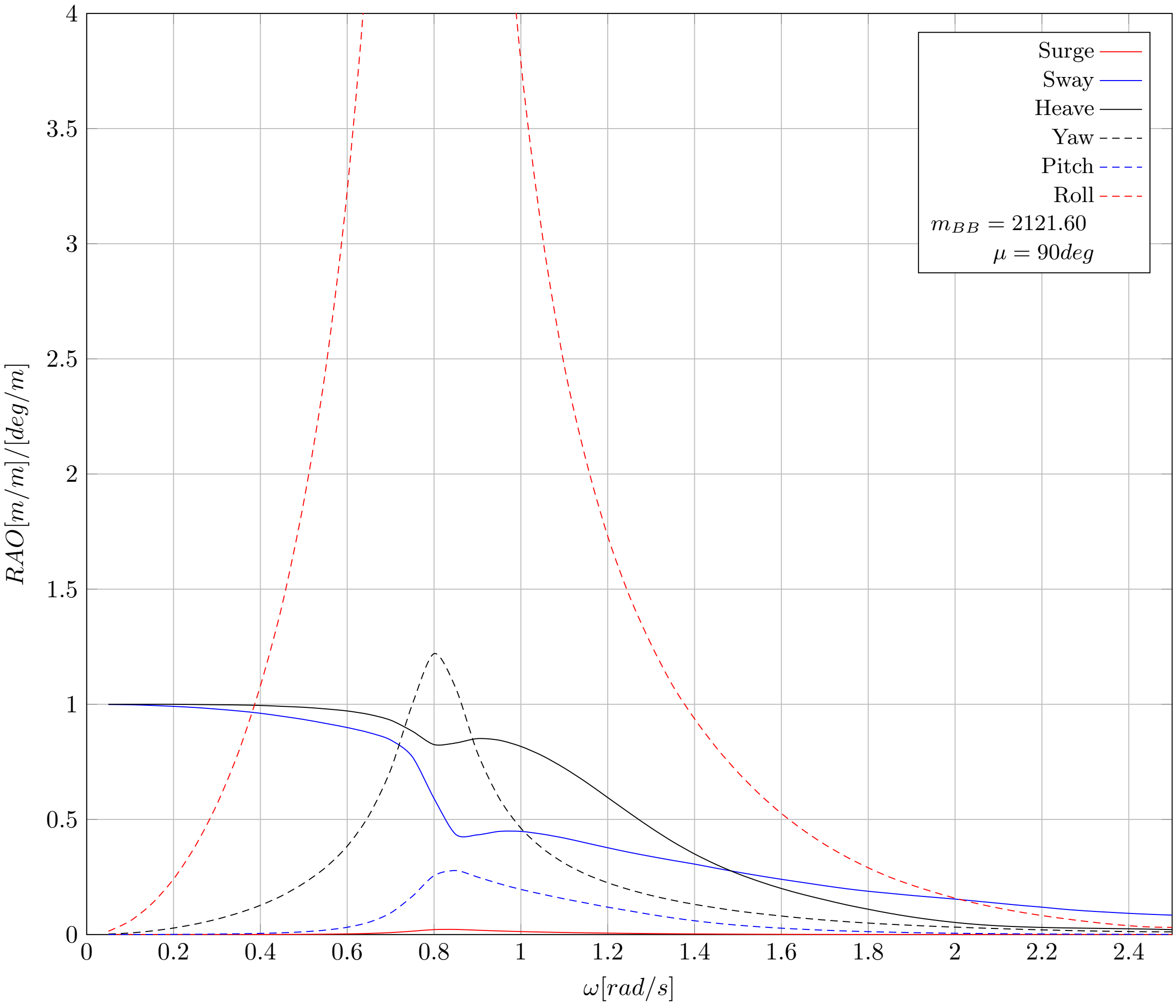
Surge
Sway
Heave
Yaw
Pitch
Roll
$m_{BB} = 2121.60$
$\mu = 0deg$
ϵ [deg]
$\omega[rad/s]$
200
150
100
50
0
−50
−100
−150
−200
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4



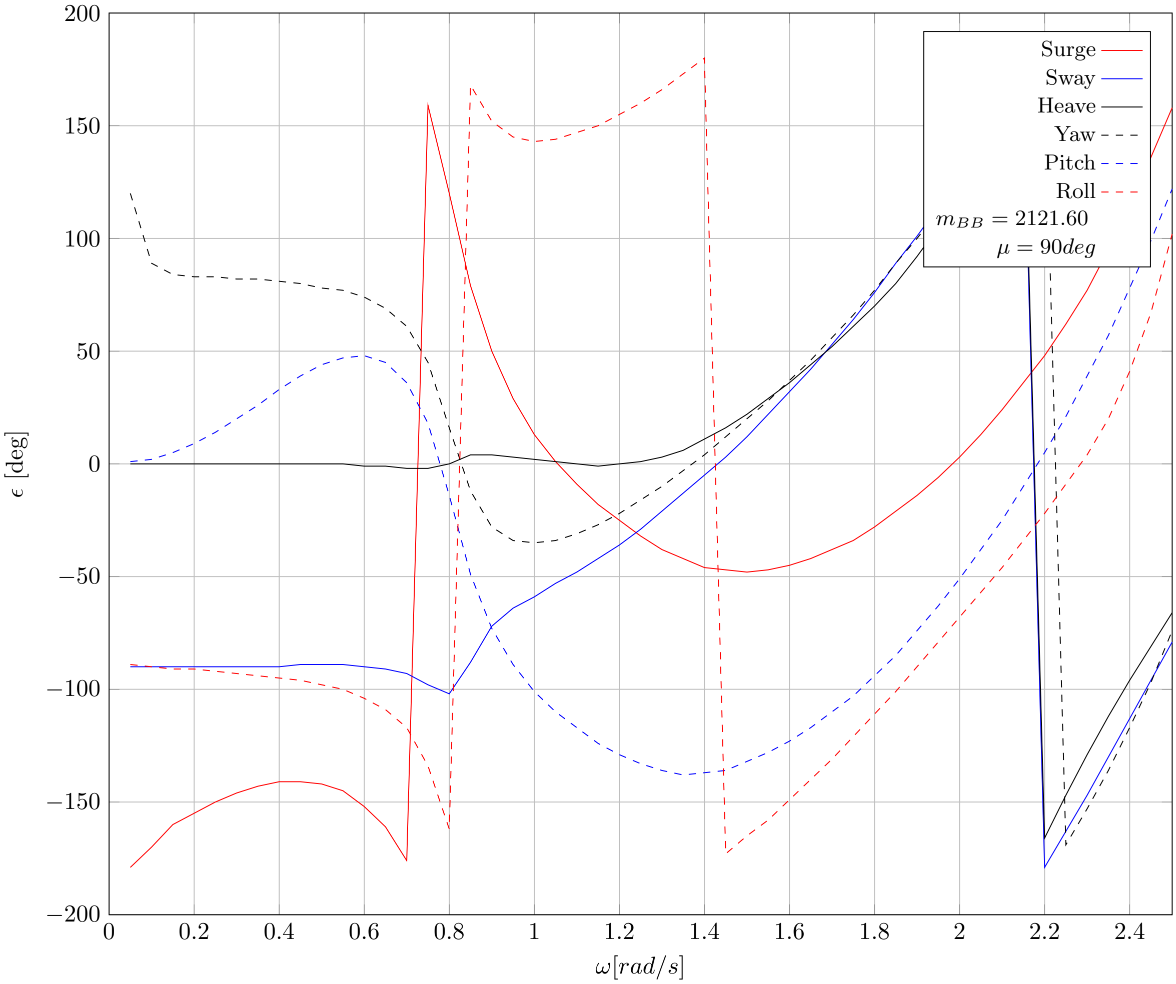
Surge
Sway
Heave
Yaw
Pitch
Roll
$m_{BB} = 2121.60$
$\mu = 45deg$
$RAO[m/m]/[deg/m]$
$\omega[rad/s]$



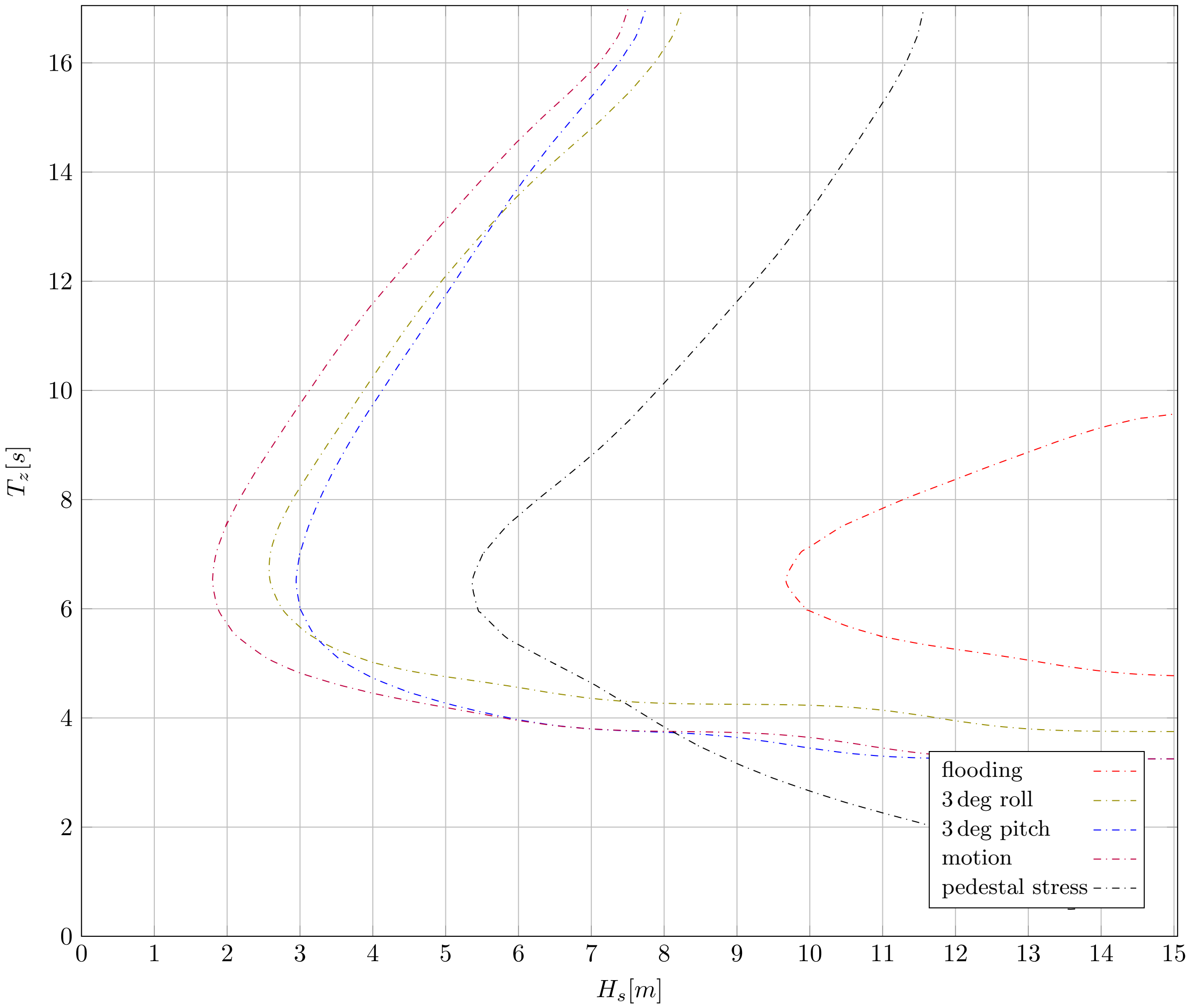
Surge
Sway
Heave
Yaw
Pitch
Roll
$m_{BB} = 2121.60$
$\mu = 45deg$
ϵ [deg]
$\omega[rad/s]$



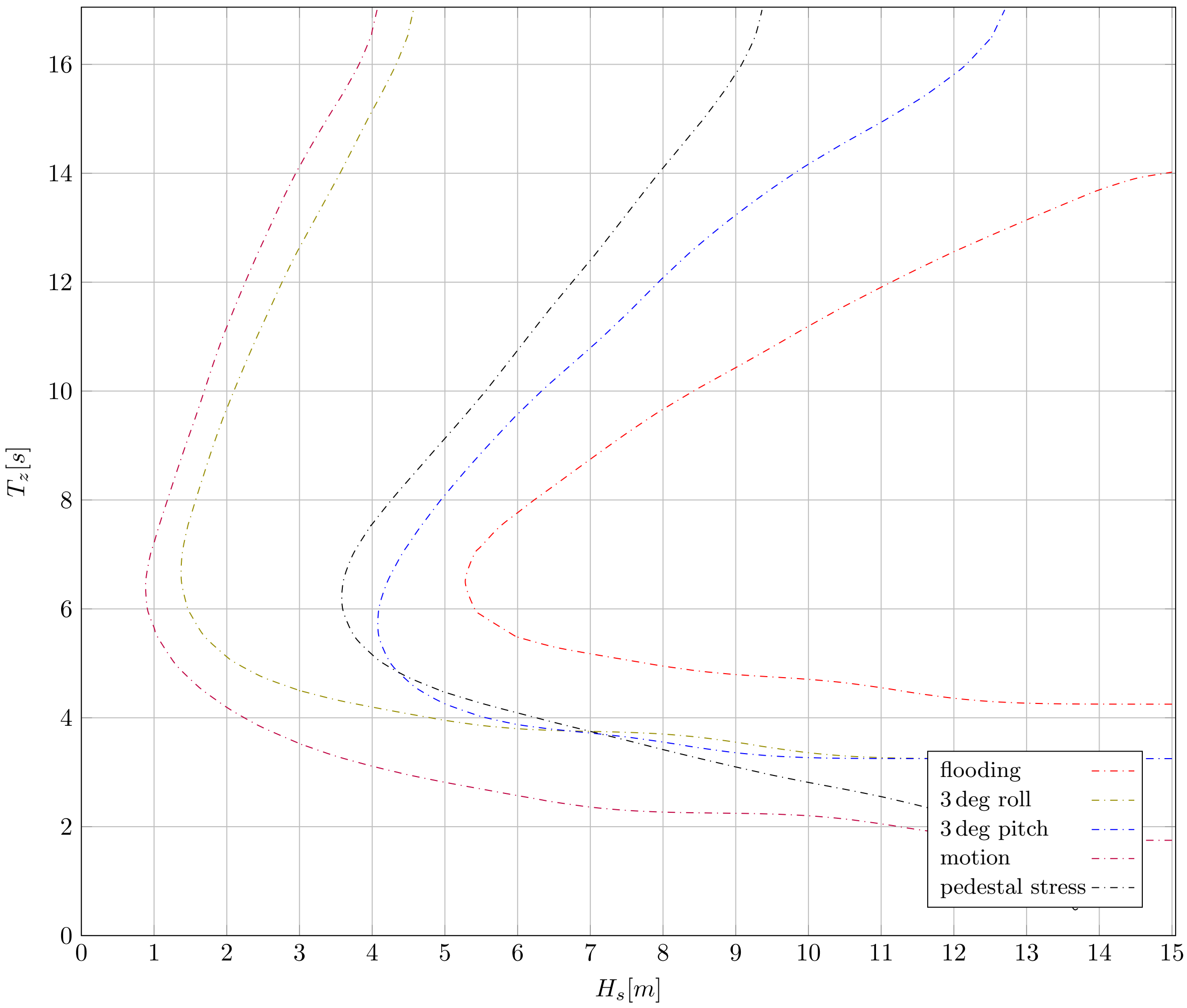
Surge
Sway
Heave
Yaw
Pitch
Roll
$m_{BB} = 2121.60$
$\mu = 90deg$
$RAO[m/m]/[deg/m]$
$\omega[rad/s]$



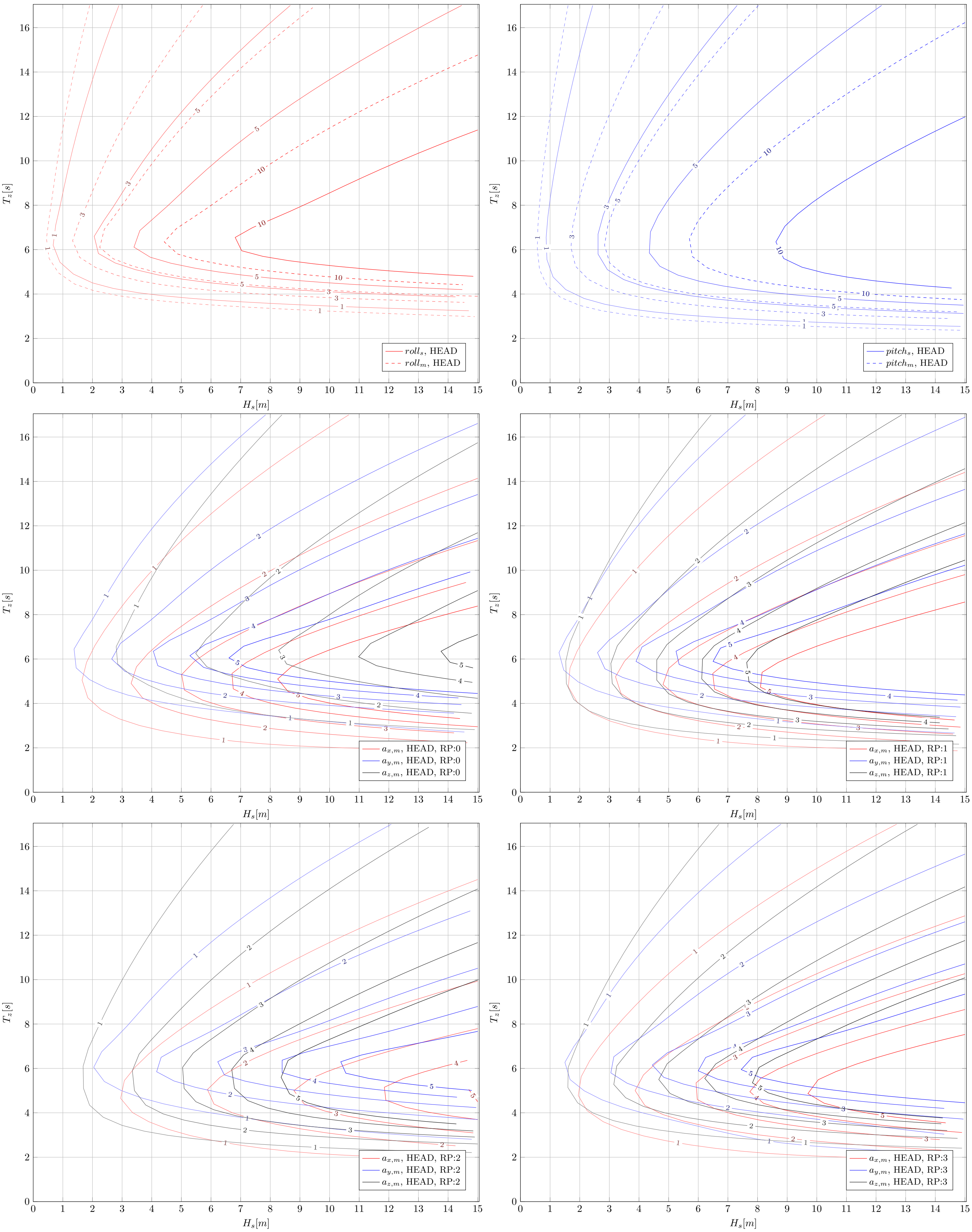
Surge
Sway
Heave
Yaw
Pitch
Roll
$m_{BB} = 2121.60$
$\mu = 90deg$
ϵ [deg]
$\omega[rad/s]$



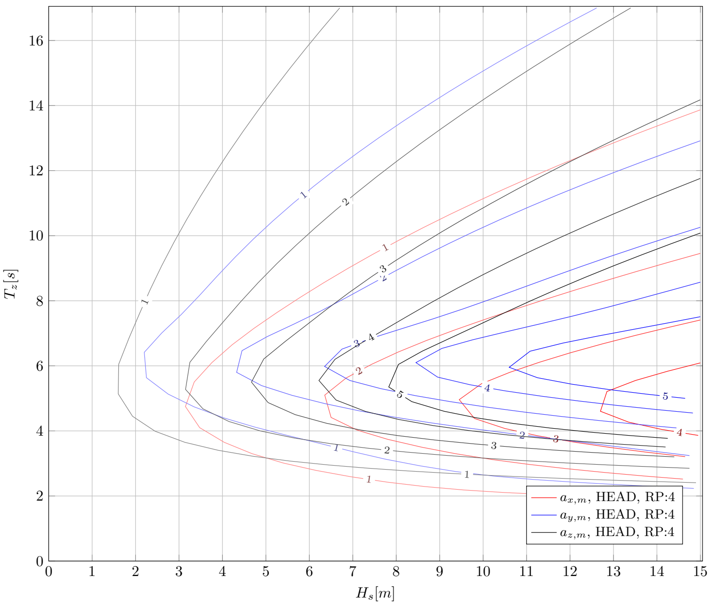
flooding
3 deg roll
3 deg pitch
motion
pedestal stress
$T_z[s]$
$H_s[m]$

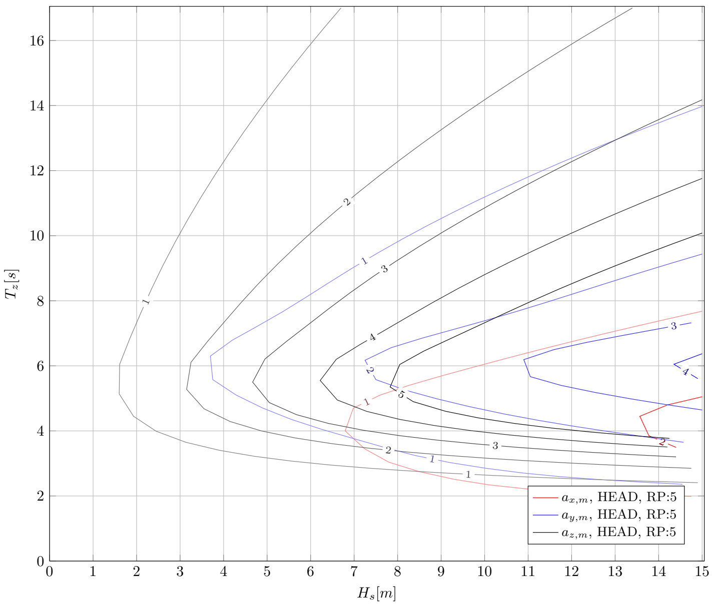


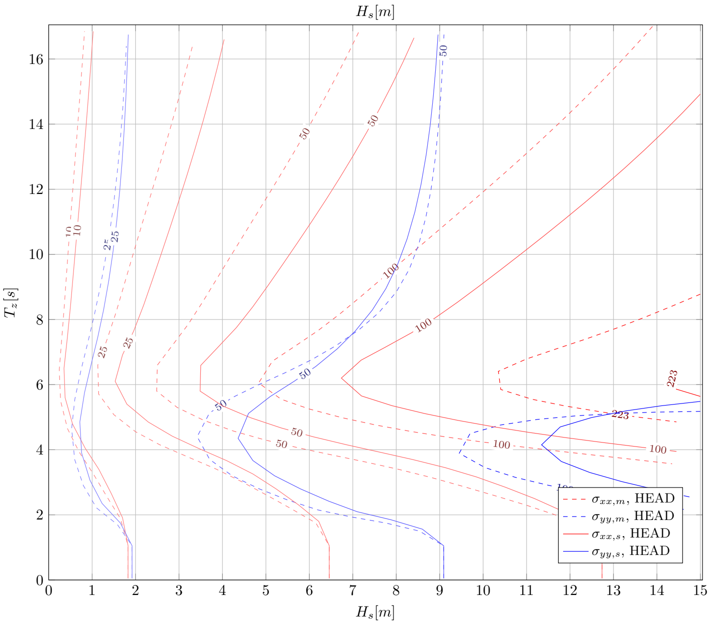
flooding
3 deg roll
3 deg pitch
motion
pedestal stress
$T_z[s]$
$H_s[m]$

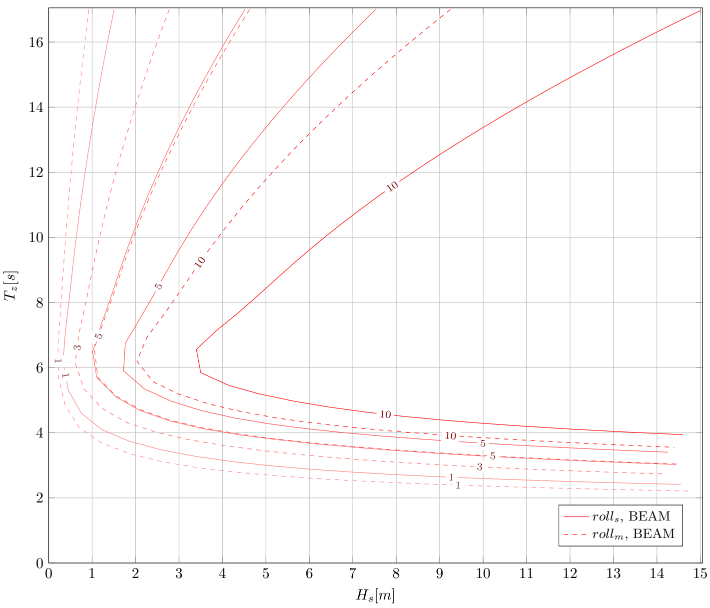


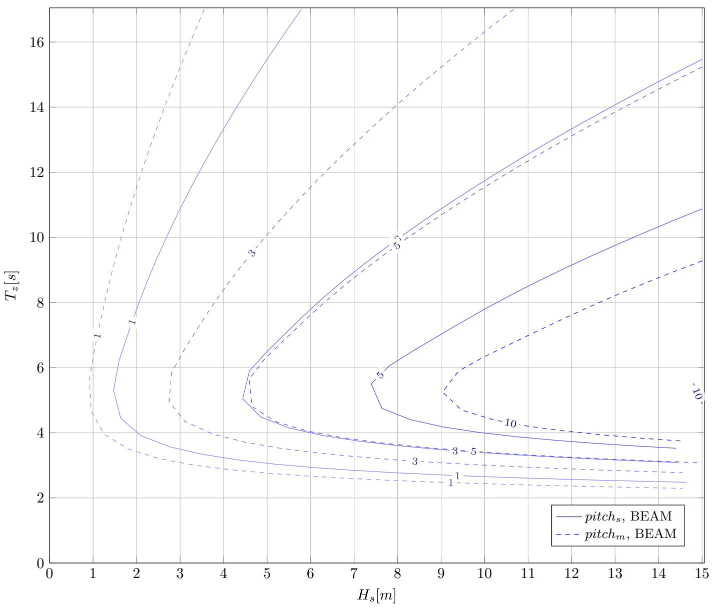
$T_z[s]$
$H_s[m]$
$roll_s$, HEAD
$roll_m$, HEAD
$pitch_s$, HEAD
$pitch_m$, HEAD
$a_{x,m}$, HEAD, RP:0
$a_{y,m}$, HEAD, RP:0
$a_{z,m}$, HEAD, RP:0
$a_{x,m}$, HEAD, RP:1
$a_{y,m}$, HEAD, RP:1
$a_{z,m}$, HEAD, RP:1
$a_{x,m}$, HEAD, RP:2
$a_{y,m}$, HEAD, RP:2
$a_{z,m}$, HEAD, RP:2
$a_{x,m}$, HEAD, RP:3
$a_{y,m}$, HEAD, RP:3
$a_{z,m}$, HEAD, RP:3

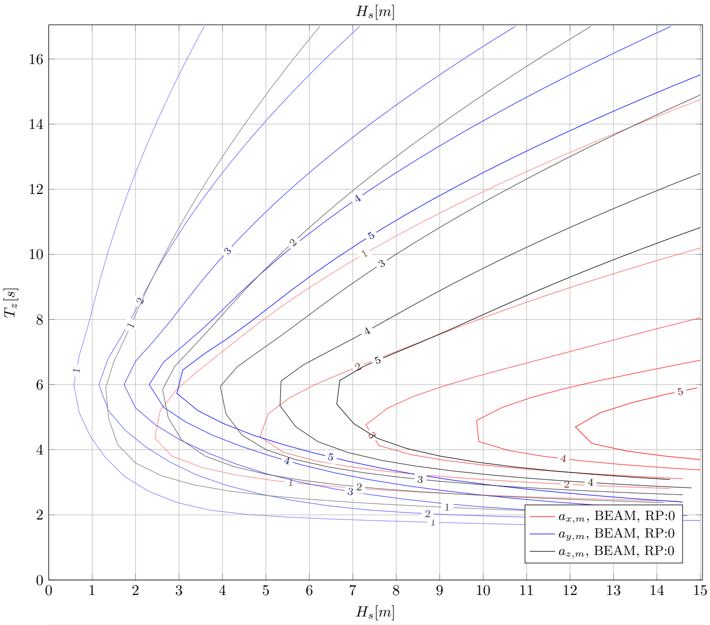

$T_z[s]$
$H_s[m]$
$a_{x,m}$, HEAD, RP:4
$a_{y,m}$, HEAD, RP:4
$a_{z,m}$, HEAD, RP:4

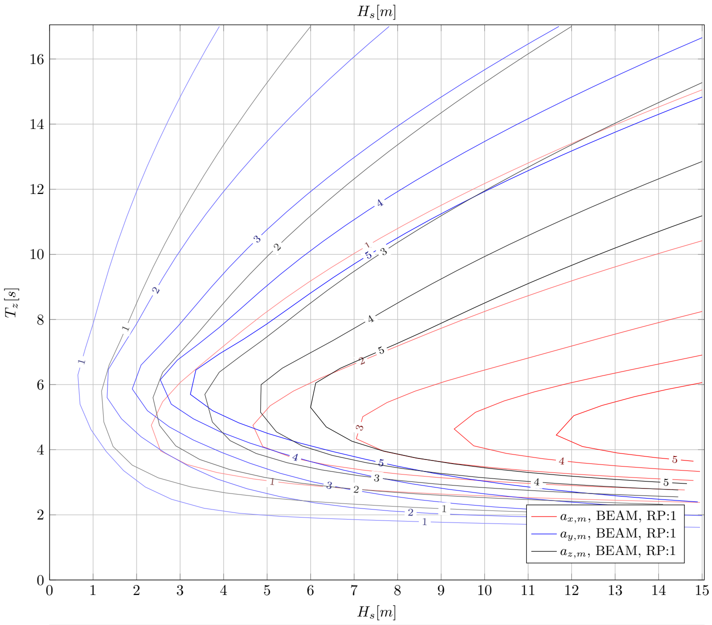

$T_z[s]$
$H_s[m]$
$a_{x,m}$, HEAD, RP:5
$a_{y,m}$, HEAD, RP:5
$a_{z,m}$, HEAD, RP:5

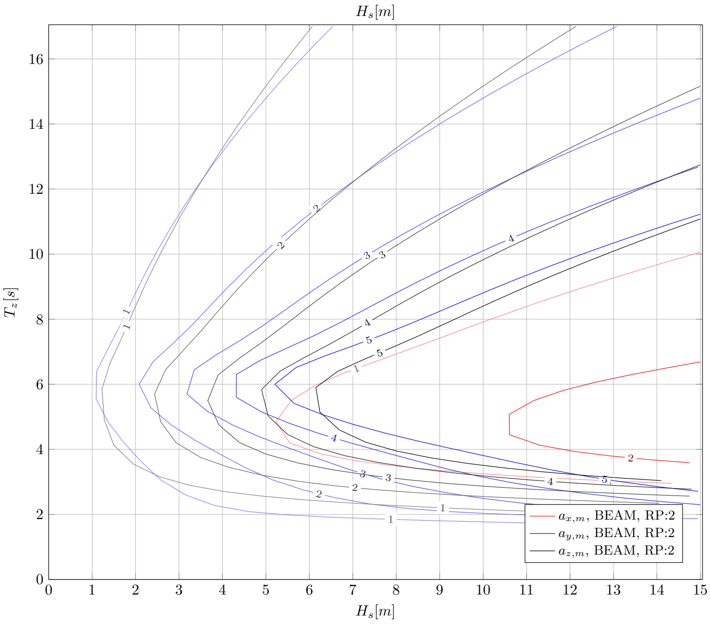

$T_z[s]$
$H_s[m]$
$\sigma_{xx,m}$, HEAD
$\sigma_{yy,m}$, HEAD
$\sigma_{xx,s}$, HEAD
$\sigma_{yy,s}$, HEAD

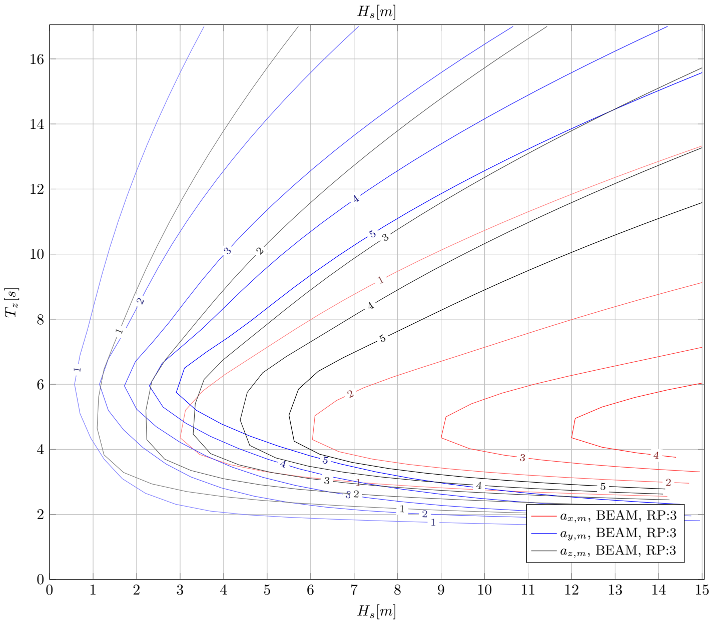

$roll_s$, BEAM
$roll_m$, BEAM
$T_z[s]$
$H_s[m]$

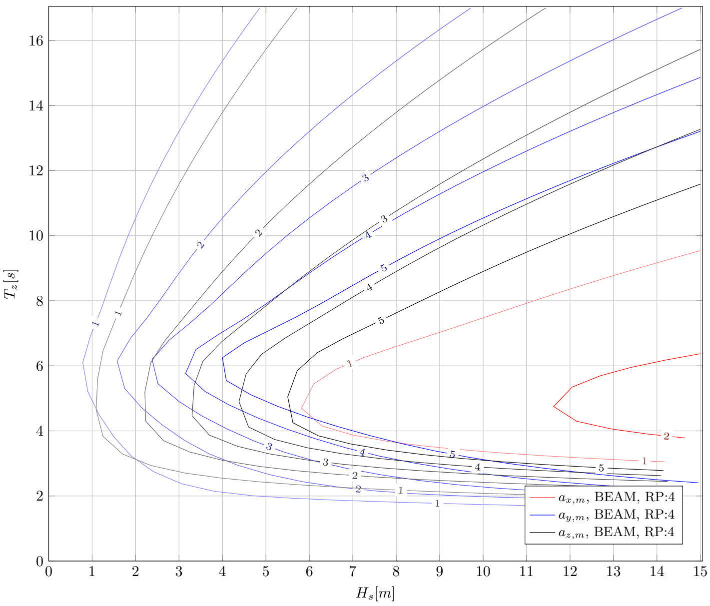

$pitch_s$, BEAM
$pitch_m$, BEAM
$T_z[s]$
$H_s[m]$


$a_{x,m}$, BEAM, RP:0
$a_{y,m}$, BEAM, RP:0
$a_{z,m}$, BEAM, RP:0
$T_z[s]$
$H_s[m]$

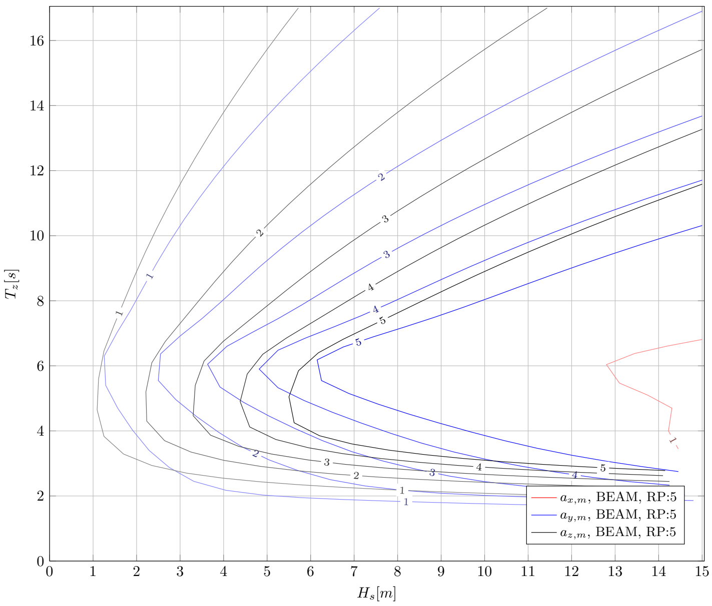

$a_{x,m}$, BEAM, RP:1
$a_{y,m}$, BEAM, RP:1
$a_{z,m}$, BEAM, RP:1
$T_z[s]$
$H_s[m]$


$a_{x,m}$, BEAM, RP:2
$a_{y,m}$, BEAM, RP:2
$a_{z,m}$, BEAM, RP:2
$T_z[s]$
$H_s[m]$

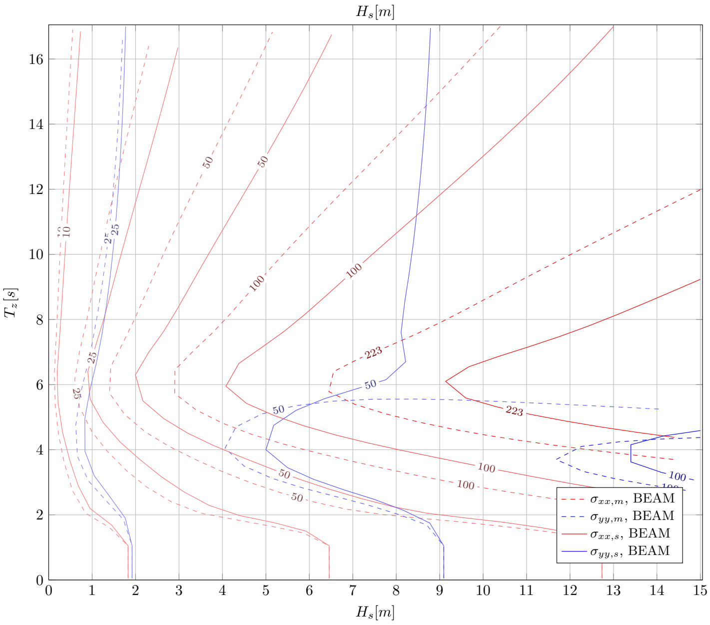

$a_{x,m}$, BEAM, RP:3
$a_{y,m}$, BEAM, RP:3
$a_{z,m}$, BEAM, RP:3
$T_z[s]$
$H_s[m]$



$T_z[s]$
$H_s[m]$
$a_{x,m}$, BEAM, RP:4
$a_{y,m}$, BEAM, RP:4
$a_{z,m}$, BEAM, RP:4



$T_z[s]$
$H_s[m]$
$a_{x,m}$, BEAM, RP:5
$a_{y,m}$, BEAM, RP:5
$a_{z,m}$, BEAM, RP:5



$T_z[s]$
$H_s[m]$
$\sigma_{xx,m}$, BEAM
$\sigma_{yy,m}$, BEAM
$\sigma_{xx,s}$, BEAM
$\sigma_{yy,s}$, BEAM

B Fatigue

In this section, more detailed information on the fatigue estimation is described. An analysis of the stress spectrum per member is done, using the final JR design, with the member numbering as shown in fig. B.1:

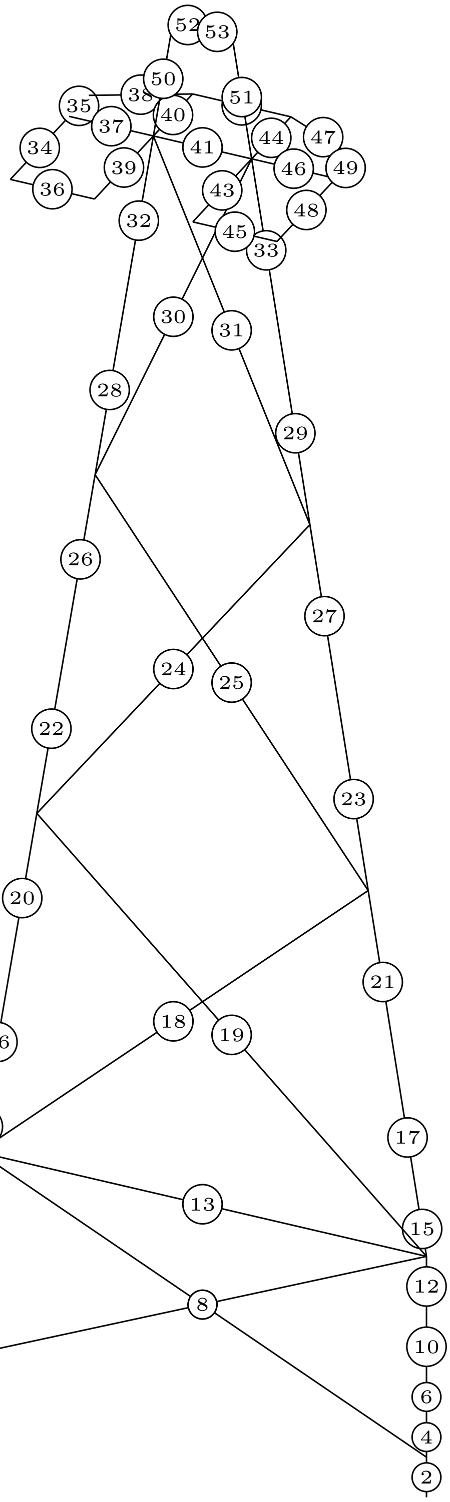


Figure B.1: Member numbering of the new design

Making a distinction between y an z direction, an acceleration of one is applied to simplified system as shown in section 9.1. Because the jib is governing in the loads, the mass of the JR itself is neglected and only the acceleration on the jib (RP:0) is considered relevant. In every member, the ratio between maximum stress and the acceleration can be found and can be multiplied by the acceleration spectrum to get the stress spectrum. Finally Dirlik's method is used to get to a range of cycles.

At this stage, the detailed design provides information on member types and location of welds in the connections. With the weld connection types known, the FAT codes can be selected per member as seen in fig. B.2.

ID	loga	gammaB	gammaN	FAT	loga2	two sides!, transverse attacheme	penetration weld, r<1/3	plate r< 1/6 or overlap	pinched pipe / hinge
1	11.855	3.028	0.116	71	15.091	E	F		
2	11.855	3.028	0.116	71	15.091	E	F		
3	11.699	3.229	0.478	63	14.832	E	F	F1	
4	11.699	3.229	0.478	63	14.832	E	F	F1	
5	12.01	4.468	0.93	80	15.35	E			
6	12.01	4.468	0.93	80	15.35	E			
7	11.398	3.481	0.449	50	14.33	E		F1	G
8	11.398	3.481	0.449	50	14.33	E		F1	G
9	12.01	3.37	1.015	80	15.35	E			
10	12.01	3.37	1.015	80	15.35	E			
11	11.699	2.498	1.584	63	14.832	E	F	F1	
12	11.699	2.498	1.584	63	14.832	E	F	F1	
13	12.01	0	0.2658	80	15.35	E			
14	11.855	2.997	0.755	71	15.091	E	F		
15	11.855	2.997	0.755	71	15.091	E	F		
16	12.01	2.345	0.754	80	15.35	E			
17	12.01	2.345	0.754	80	15.35	E			
18	11.398	2.643	0.43	50	14.33	E		F1	G
19	11.398	2.643	0.43	50	14.33	E		F1	G
20	11.699	2.711	0.85	63	14.832	E		F1	
21	11.699	2.711	0.85	63	14.832	E		F1	
22	11.699	2.875	0.525	63	14.832	E		F1	
23	11.699	2.875	0.525	63	14.832	E		F1	
24	11.398	3.093	0.1686	50	14.33	E		F1	G
25	11.398	3.093	0.1686	50	14.33	E		F1	G
26	11.699	2.644	0.831	63	14.832	E		F1	
27	11.699	2.644	0.831	63	14.832	E		F1	
28	11.699	2.402	0.52	63	14.832	E		F1	
29	11.699	2.402	0.52	63	14.832	E		F1	
30	11.398	2.581	0.567	50	14.33	E		F1	G
31	11.398	2.581	0.568	50	14.33	E		F1	G
32	11.699	2.659	0.767	63	14.832	E	F	F1	
33	11.699	2.659	0.767	63	14.832	E	F	F1	
34	12.01	0.25	0.458	80	15.35	E			
35	12.01	0.375	0.356	80	15.35	E			
36	12.01	0.042	0.081	80	15.35	E			
37	12.01	0.12	0.857	80	15.35	E			
38	12.01	0.066	0.256	80	15.35	E			
39	12.01	0.367	0.867	80	15.35	E			
40	12.01	0.068	0.233	80	15.35	E			
41	12.01	0	0.093	80	15.35	E			
42	12.01	0	0.119	80	15.35	E			
43	12.01	0.367	0.867	80	15.35	E			
44	12.01	0.068	0.233	80	15.35	E			
45	12.01	0.042	0.081	80	15.35	E			
46	12.01	0.12	0.857	80	15.35	E			
47	12.01	0.066	0.256	80	15.35	E			
48	12.01	0.25	0.458	80	15.35	E			
49	12.01	0.375	0.356	80	15.35	E			
50	11.855	1.771	1.183	71	15.091	E	F		
51	11.855	1.771	1.182	71	15.091	E	F		
52	11.855	3.53	1.195	71	15.091	E	F		
53	11.855	3.53	1.195	71	15.091	E	F		

Figure B.2: Member FAT code selection

The spectra and FAT codes provide the amount of cycles per time unit and the maximum number of cylces, respectfully. By selecting a timerange - and acceleration spectrum - the fatigue lifetime can be estimated. For this, three different cases are examined; two cases with 20yr lifespan in different locations and one case with a two week transport in rough weather. Figure B.3 shows a sheet of the results of the analysis of the cases for the new design, where the first column represents the member number, the next three the percentage lifetime expected to be spent for the three different load cases, under head waves. The three following columns represent the cases under beam waves and the final three show a cumulative damage expected under 70% head waves.

member ID	case	case	rough trip	case	case	rough trip			
		head			beam			70/30	
0	1	2		1	2		0.7	0.3	
1	0.01	0.01	0	0.06	0.07	0.02	0.025	0.028	0.006
2	0.01	0.01	0	0.06	0.07	0.02	0.025	0.028	0.006
3	0.02	0.02	0	0.14	0.17	0.03	0.056	0.065	0.009
4	0.02	0.02	0	0.14	0.17	0.03	0.056	0.065	0.009
5	0.02	0.03	0.01	0.21	0.26	0.02	0.077	0.099	0.013
6	0.02	0.03	0.01	0.21	0.26	0.02	0.077	0.099	0.013
7	0.07	0.08	0.01	0.61	0.74	0.07	0.232	0.278	0.028
8	0.07	0.08	0.01	0.61	0.74	0.07	0.232	0.278	0.028
9	0.01	0.01	0	0.05	0.06	0.02	0.022	0.025	0.006
10	0.01	0.01	0	0.05	0.06	0.02	0.022	0.025	0.006
11	0	0	0	0.04	0.05	0.01	0.012	0.015	0.003
12	0	0	0	0.04	0.05	0.01	0.012	0.015	0.003
13	0	0	0	0	0	0	0	0	0

Figure B.3: Expected cumulative damage per member

C Pedestal Stress

Two outer stresses in the pedestal are analysed. One maximum over the local x-axis of the pedestal in the outer edge of the circular section, where bending and normal stresses are combined. And the same over the y-axis. The calculations are constructed into a function which returns both stresses. Wind forces are taken into account by using wind speeds found from the wind coupling and assumed to be working pependicular on the selected axis. This is combined with the forces coming from the accelerations in all respective reference points and the crane or jib rest elements modeled as point masses. The function and its calculations can be found in the following code snippet:

```
    if 3.6 > Tp / math.sqrt(Hs) or 5 < Tp / math.sqrt(Hs):
        rep = 0

    SPM = np.zeros(len(omegas))
    SJ = SPM.copy()
    for iW, w in enumerate(omegas):
        if w == 0:w=0.05;
        SPM[iW] = 5/16 * Hs ** 2 * math.pow(wp, 4) * math.pow(w, -5) * np.exp(-5 / 4 * math.pow(w / wp, -4))
        if w <= wp:
            s = sa
        else:
            s = sb
        SJ[iW] = Ag * SPM[iW] * math.pow(gamma, np.exp(-0.5 * math.pow((w - wp) / (s * wp), 2)))

    if representable:
        return (SJ, rep)
    else:
        return SJ

def windSpeed(Hs:float):
    return 7.3515 * math.pow(Hs, 0.6187) #made with a fit from excel

def windLoad(U, Cd, rho, A):
    return 0.5 * rho * math.pow(U, 2) * Cd * A * 9.81

def pedestalStress(aRP0, aRP1, aRP2, roll, pitch, U, par:dict):

    APed = math.pi * (pow(par['crRadiusPedistal']['value'], 2) -
                      pow(par['crRadiusPedistalInner']['value'], 2))
    forcesPed = np.zeros((3, 3)) # iRP, xyz, mu, Omega :: RAOa= # xyz, refpoint, U, mu, Omega, RAO

    armPed = np.zeros((3, 3)) #iRP, xyz
    armPed[0, 0] = 0 #par['pcalJibX']['value'] - par['pcalPedX']['value']
    armPed[0, 1] = 0 #par['pcalJibY']['value'] - par['pcalPedY']['value']
    armPed[0, 2] = abs(par['pcalJibZ']['value'] - par['baHeight']['value'])
    armPed[1, 0] = abs(par['pcalCraneTopX']['value'] - par['pcalPedX']['value'])
    armPed[1, 1] = 0 #par['pcalCraneTopY']['value'] - par['pcalPedY']['value'] # No moment due to swivel
    armPed[1, 2] = abs(par['pcalCraneTopZ']['value'] - par['baHeight']['value'])
    armPed[2, 0] = 0 #abs(par['pcalPedX']['value'] - par['pcalPedX']['value'])
    armPed[2, 1] = 0 #abs(par['pcalPedY']['value'] - par['pcalPedY']['value'])
    armPed[2, 2] = abs(par['pcalPedZ']['value'] - par['baHeight']['value'])
    #iRP, xyz
    # also presuming wind gusts from x and y direction!!!!
    forcesPed[0, 0] = par['pcalMassJib']['value'] * aRP0[0] * 1000
    forcesPed[0, 1] = par['pcalMassJib']['value'] * aRP0[1] *\
        (1 - par['pllForceRatio']['value']) * 1000
    forcesPed[0, 2] = - 1 * par['pcalMassJib']['value'] * (aRP0[2] + 9.81) *\
        (1 - par['pllForceRatio']['value']) * 1000
    forcesPed[1, 0] = par['pcalMassCraneTop']['value'] * aRP1[0] * 1000
    forcesPed[1, 1] = par['pcalMassCraneTop']['value'] * aRP1[1] * 1000
    forcesPed[1, 2] = - 1 * par['pcalMassCraneTop']['value'] * (aRP1[2] + 9.81) * 1000
    forcesPed[2, 0] = par['pcalMassPed']['value'] * aRP2[0] * 1000
    forcesPed[2, 1] = par['pcalMassPed']['value'] * aRP2[1] * 1000
    forcesPed[2, 2] = - 1 * par['pcalMassPed']['value'] * (aRP2[2] + 9.81) * 1000

    #adding wind
    forcesPed[0, 1] += (1 - par['pllForceRatio']['value']) * windLoad(U,
                                                                     par['pllDragCoeff']['value'],
                                                                     par['pllRhoAir']['value'],
```

```
                                                                   par['crAreaJibSide']['value'])
    forcesPed[2, 0] += windLoad(U,
                                par['pllDragCoeffPed']['value'],
                                par['pllRhoAir']['value'],
                                par['crRadiusPedistal']['value'] * 2 * par['crHr']['value']  )

    forcesPed[2, 1] += windLoad(U,
                                par['pllDragCoeffPed']['value'],
                                par['pllRhoAir']['value'],
                                par['crRadiusPedistal']['value'] * 2 * par['crHr']['value']  )

    forcesPed[1, 0] += windLoad(U,
                                par['pllDragCoeffPed']['value'],
                                par['pllRhoAir']['value'],
                                par['crRadiusPedistal']['value'] * 2 * par['crHr']['value']  )

    forcesPed[1, 0] += windLoad(U,
                                par['pllDragCoeffPed']['value'],
                                par['pllRhoAir']['value'],
                                par['crRadiusPedistal']['value'] * 2 * par['crHtot']['value']  )

    forcesPed[1, 1] += windLoad(U,
                                par['pllDragCoeffPed']['value'],
                                par['pllRhoAir']['value'],
                                par['crRadiusPedistal']['value'] * 2 * par['crHtot']['value']  )

    transPitch = np.zeros((3, 3))
    transRoll = transPitch.copy()
    relativeForces = forcesPed.copy()


    for iRP in (0, 1, 2):
        transPitch = rotation_matrix([0, 1, 0], math.radians(pitch))
        transRoll = rotation_matrix([1, 0, 0], math.radians(roll))
        relativeForces[iRP, :] = np.dot(transPitch, relativeForces[iRP, :])
        relativeForces[iRP, :] = np.dot(transRoll, relativeForces[iRP, :])

    forcesPedZ = np.sum(relativeForces[:, 2]) #iRP, xyz, mu, Omega
    momentsPed = np.empty((3,)) # XYZ, Mu, Omegas
    momentsPed[0] = relativeForces[0, 1] * armPed[0, 2] +\
                    relativeForces[1, 1] * armPed[1, 2] +\
                    relativeForces[2, 1] * armPed[2, 2] +\
                    relativeForces[0, 2] * armPed[0, 1] +\
                    relativeForces[1, 2] * armPed[1, 1] +\
                    relativeForces[2, 2] * armPed[2, 1] # OK! #iRP, XYZ

    momentsPed[1] = relativeForces[0, 0] * armPed[0, 2] +\
                    relativeForces[1, 0] * armPed[1, 2] +\
                    relativeForces[2, 0] * armPed[2, 2] +\
                    relativeForces[0, 2] * armPed[0, 0] -\
                    relativeForces[1, 2] * armPed[1, 0] +\
                    relativeForces[2, 2] * armPed[2, 0]

    momentsPed[2] = relativeForces[0, 0] * armPed[0, 1] +\
                    relativeForces[1, 0] * armPed[1, 1] +\
                    relativeForces[2, 0] * armPed[2, 1] +\
                    relativeForces[0, 1] * armPed[0, 0] +\
                    relativeForces[1, 1] * armPed[1, 0] +\
                    relativeForces[2, 1] * armPed[2, 0]


    stressXX = abs(momentsPed[0] * par['crRadiusPedistal']['value'] /\
                   par['pllIrrPed']['value']) +\
                   abs(forcesPedZ / APed)

    stressYY = abs(momentsPed[1] * par['crRadiusPedistal']['value'] /\
                    par['pllIrrPed']['value']) +\
```

D Engineering

This appendix holds detailed information on the structural details of the jib rest with its calculations. This section is omitted intentionally, as the information it contains is considered classified.

E Code

This section shows the most relevant code that is used to program the solver and any related sub functions. Its basic object structure in graphed in an UML-diagram in fig. E.1. Followed by the actual Python code of the solver.

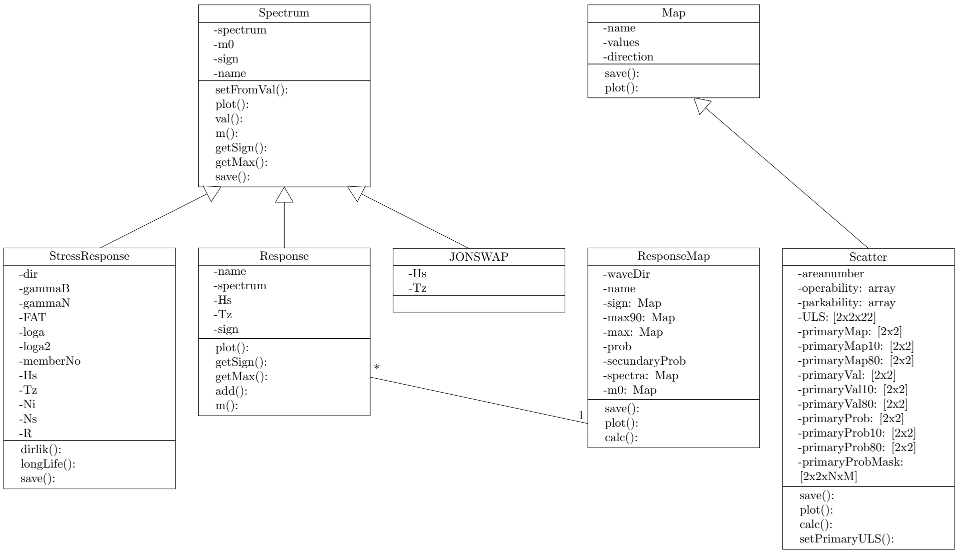


Figure E.1: UML diagram of the solver

```python
from CalcFunctions import jonswap, pedestalStress, windSpeed, windLoad,\
buildScatter, combinedProb, savePickle, loadPickle, rotation_matrix, multiframe
import pickle as pk
import matplotlib.pyplot as plt
import numpy as np
import math as math
import TikzWriter as tk
import CalcFunctions as cf
import warnings



class RAO:

    def __init__(self, location):
        self.valueArray = np.loadtxt(location, delimiter = ",", skiprows=1)
        self.valueArray[:, 7] = self.valueArray[:, 7] / 180 * math.pi

    def val(self, mu):
        #RAO, phase
        return self.valueArray[self.valueArray[:, 3] == mu, 6], \
            self.valueArray[self.valueArray[:, 3] == mu, 7]

    def plot(self, mu):
        global omegas
        plt.plot(omegas, self.val(mu)[0])


class Spectrum:

    def __init__(self, spectrum, name=""):
        self.spectrum = spectrum
        self.m0 = self.m(0)
        self.sign = self.getSign()
        self.name = name

    def spectrum(self):
        return self.spectrum

    def set(self, spectrum):
```

```
        self.spectrum = spectrum

    def setFromVal(self, val):
        self.spectrum = 0.5 * np.power(np.abs(val), 2) / deltaOmega

    def plot(self):
        global omegas
        plt.plot(omegas, self.spectrum, label="S(" + self.name + ") " +\
                 "m0:%.2f"%self.m0)

    def val(self):
        global deltaOmega
        return np.sqrt(2 * self.spectrum * deltaOmega)

    def m(self, n):
        global omegas, deltaOmega
        return np.sum(np.multiply(np.power(omegas, n), self.spectrum) * deltaOmega)

    def getSign(self):
        global deltaOmega
        # AMPLITUDE!!!
        #return 2 * math.sqrt(self.m(0))
        return 2 * math.sqrt(np.sum(self.spectrum * deltaOmega))

    def getMax(self, prob):
        return math.sqrt(-math.log(1 - prob) * 2 * self.m0)

    def save(self):
        global savePath
        np.savetxt(savePath + self.name + "Spectrum.csv", self.sign,
                   delimiter=",")

class JONSWAP(Spectrum):

    def __init__(self, Hs, Tz):
        global omegas
        global gamma
        self.Hs = Hs
        self.Tz = Tz
        Spectrum.__init__(self, cf.jonswap(omegas, Hs, Tz, gamma),
                          name="JONSWAP" + str(int(np.argmax(heights >= Hs))) +\
                          "_" + str(int(np.argmax(periods >= Tz))))

class Response:

    def __init__(self, response, Hs, Tz, dirW, name):

        self.responseDir = response
        if dirW == 0:
            self.waveDir = "HEAD"
        elif dirW == 90:
            self.waveDir = "BEAM"
        else:
            self.waveDir = str(int(dirW))

        if len(np.shape(response)) > 1:
            self.spectrum = Spectrum(np.sum(response, 0), name + self.waveDir)
        else:
            self.spectrum = Spectrum(response, name + self.waveDir)
        self.sign = self.spectrum.getSign()
        self.Tz = Tz
        self.Hs = Hs
        self.name = name

    def plot(self):
        plt.title("Hs:{:.2f} Tz:{:.2f} dir:{}".format(self.Hs, self.Tz,
                                                       self.waveDir))
        self.spectrum.plot()

    def getSign(self):
        return self.spectrum.getSign()

    def getMax(self, prob):
```

```
        return self.spectrum.getMax(prob)

    def add(self, value):
        self.spectrum.spectrum += value

    def m(self, n):
        return self.spectrum.m(n)

class Map:

    global heights
    global periods

    def __init__(self, values=None, name="Nameless", dirW=""):

        if values == None:
            values = np.zeros((len(heights), len(periods)))

        self.values = values
        self.name = name
        self.dirW = dirW

    def save(self, subfolder="", header=""):
        global savePath
        np.savetxt(savePath + subfolder + "" + self.name + "Map" +\
                   str(self.dirW) + ".csv",
                   self.values, delimiter=",",
                   header=header)

    def plot(self, levels=[0.5]):
        plt.contour(heights, periods, self.values.transpose(),
                    label=self.name, levels=levels)

    def __str__(self):
        return str(self.values)

class ResponseMap:

    def __init__(self, name="Response", dirW="", prob=0.95, secundaryProb=0.90):
        self.waveDir = dirW
        self.name = name
        self.sign = Map(name=self.name + "Sign")
        self.max90 = Map(name=self.name + "Max90")
        self.max = Map(name=self.name + "Max")
        self.prob = prob
        self.secundaryProb=secundaryProb
        self.spectra = Map(values=np.zeros((len(heights), len(periods)),
                                             dtype=object), name=self.name)
        self.m0 = np.zeros((len(heights), len(periods)))

    def save(self):
        global savePath
        np.savetxt(savePath + 'Responses/' + self.name + "SignMap" +\
                   str(self.waveDir) + ".csv", self.sign.values, delimiter=",")
        np.savetxt(savePath + 'Responses/' + self.name + "MaxMap" +\
                   str(self.waveDir) + ".csv", self.max.values, delimiter=",")
#         selfFile = bz2.BZ2File(savePath + 'Responses/' + self.name + str(self.waveDir) + ".json.bz2", 'wb')
#         json.dump(self, selfFile)
#         selfFile.close()
        cf.savePickle(self, self.name + str(self.waveDir),
                      savePath + 'Responses/')

    def plot(self):
        self.sign.plot()
        self.max.plot()

    def calc(self):
        print("calculating response map for: " + self.name + " " + self.waveDir)
        for iM, spec in np.ndenumerate(self.spectra.values):
            if  not spec == 0:
                self.sign.values[iM] = spec.getSign()
                self.max.values[iM] = spec.getMax(self.prob)
                self.max90.values[iM] = spec.getMax(self.secundaryProb)
```

```
                self.m0[iM] = spec.m(0)

    def load(self):
#         selfFile = bz2.BZ2File(savePath + 'Responses/' + self.name + str(self.waveDir) + ".json.bz2", 'rb')
#         s = json.load(selfFile)
#         selfFile.close()
#         return cf.loadPickle(self.name + str(self.waveDir), savePath + 'Responses/')
        s = cf.loadPickle(self.name + str(self.waveDir), savePath + 'Responses/')
        self.waveDir = s.waveDir
        self.name = s.name
        self.sign = s.sign
        self.max90 = s.max90
        self.max = s.max
        self.prob = s.prob
        self.secundaryProb = s.secundaryProb
        self.spectra = s.spectra
#         print(self.spectra)
        self.m0 = s.m0


class Scatter(Map):

    global heights
    global periods

    def __init__(self, values, areaNumber):
        Map.__init__(self, values, name="Scatter" + str(int(areaNumber)))
        self.areaNumber = areaNumber
        self.operability = [0, 0]
        self.parkability = [0, 0]
        self.ULS = np.zeros((2, 2, 22)) #[direction; Y/Z; xRP0, yRP0, ..., zRP5, roll, pitch, Hs, U]
        self.primaryMap80 = np.empty((2, 2), dtype=object)
        self.primaryMap10 = np.empty((2, 2), dtype=object)
        self.primaryMap = np.empty((2, 2), dtype=object)
        self.primaryVal80 = np.empty((2, 2), dtype=object)
        self.primaryVal10 = np.empty((2, 2), dtype=object)
        self.primaryVal = np.empty((2, 2), dtype=object)
        self.primaryProb80 = np.empty((2, 2), dtype=object)
        self.primaryProb10 = np.empty((2, 2), dtype=object)
        self.primaryProb = np.empty((2, 2), dtype=object)
        self.primaryProbMask = np.empty((2, 2, len(heights), len(periods)))

    def plot(self, maskMap = None, levels=[1e-6, 1e-5, 1e-4]):
        global heights
        global periods

        if maskMap == None:
            maskMap = np.ones((len(heights), len(periods)))
        elif isinstance(maskMap, Map):
                maskMap = maskMap.values

        output = np.multiply(self.values, maskMap)
        plt.contour(heights, periods, output.transpose(), label="Area " +\
                    str(self.areaNumber), levels=levels)
    def setPrimaryULS(self):
        self.ULS[0, 0, 1] = self.primaryVal[0, 0]
        self.ULS[1, 0, 1] = self.primaryVal[1, 0]
        self.ULS[0, 1, 2] = self.primaryVal[0, 1]
        self.ULS[1, 1, 2] = self.primaryVal[1, 1]

    def save(self):
        global savePath
        for iD, dirW in enumerate(("HEAD", "BEAM")):
            for iYZ, YZ in enumerate(("Y", "Z")):
                np.savetxt(savePath + 'Scatter/Area' +
                           str(int(self.areaNumber)) + ".csv",
                           self.values, delimiter=",")
                self.primaryMap[iD, iYZ].save(subfolder="Areas/" + dirW + "/" +
                                              YZ + "/" + self.name, header="v," +
                                              str(round(self.primaryVal[iD, iYZ], 2)) +
                                              ",p," + str(round(self.primaryProb[iD, iYZ], 3)))

                self.primaryMap10[iD, iYZ].save(subfolder="Areas/" + dirW + "/" +
```

```
                                                YZ + "/" + self.name, header="v," +
                                                str(round(self.primaryVal10[iD, iYZ], 2)) +
                                                ",p," + str(round(self.primaryProb10[iD, iYZ], 3)))
                self.primaryMap80[iD, iYZ].save(subfolder="Areas/" + dirW + "/" +
                                                YZ + "/" + self.name, header="v," +
                                                str(round(self.primaryVal80[iD, iYZ], 2)) +
                                                ",p," + str(round(self.primaryProb80[iD, iYZ], 3)))

                ULSfile = open(savePath + "ULS/ULS.csv", 'a')
                valStr = ""
                for val in self.ULS[iD, iYZ, :]: valStr += str(round(val, 2)) + ",";
                ULSfile.write(dirW + "," + YZ + "," + str(int(self.areaNumber)) +
                              "," + valStr + "\n")
                ULSfile.close()

                probVal = [[round(self.primaryVal[iD, iYZ], 2),
                            round(self.primaryVal10[iD, iYZ], 2),
                            round(self.primaryVal80[iD, iYZ], 2)],
                           [round(self.primaryProb[iD, iYZ], 3),
                            round(self.primaryProb10[iD, iYZ], 3),
                            round(self.primaryProb80[iD, iYZ], 3)]]
                np.savetxt(savePath + "Areas/" + dirW + "/" + YZ + "/" +
                           self.name + "Values.csv", probVal, delimiter=',')
                np.savetxt(savePath + "Areas/" + dirW + "/" + YZ + "/" +
                           self.name + "ProbMask.csv", self.primaryProbMask[iD, iYZ], delimiter=',')

    def saveOperability(self):
        global savePath
        np.savetxt(savePath + "Operability/Area" + self.name + ".csv", [self.operability, self.parkability],
                   delimiter=",",
                   header="OPHead, OPBeam\n PAHead, PABeam]")


class StressResponse(Spectrum):
    global Srange
    global deltaStress
    global fatSlope

    def __init__(self, spectrum=None, factor = 1, memberNo=0, Hs=10, Tz=10,
                 gammaB=0.1, gammaN=0.1, loga=10, loga2=14, FAT=32,
                 dirW="DIRECTION", comment="BN"):
        self.accelResponse = spectrum
        Spectrum.__init__(self, spectrum=spectrum * factor,
                          name="StressResponse" + comment + str(int(memberNo)) +
                          "_" + str("%.1f"%Hs) + "_" + str("%.1f"%Tz))
        self.dirW = dirW
        self.gammaB = gammaB
        self.gammaN = gammaN
        self.FAT = FAT
        self.loga = loga
        self.loga2 = loga2
        self.memberNo = memberNo
        self.Hs = Hs
        self.Tz = Tz
        if np.max(spectrum) == 0.0:
            self.Ni = np.zeros(np.shape(Srange))
            self.NS = np.zeros(np.shape(Srange))
        else:
            self.Ni = self.longLife() # longlife cycles
            self.NS = self.dirlik(Srange) # dirlik cycles
        self.R = np.sum(np.divide(self.NS, self.Ni)) * deltaStress #Damage per second per sea state
#         print(self.Ni)
#         print(self.NS)

    def dirlik(self, Srange):

        N = np.zeros(len(Srange))
        pSLower = 0
        if self.m0 > 0:
            for iS, S in enumerate(Srange):
                m0 = self.m0
                m1 = self.m(1)
                m2 = self.m(2)
```

```
                m4 = self.m(4)
                xm = m1 / m0 * math.sqrt(m2 / m4)
                gam = m2 / math.sqrt(m0 * m4)
                D1 = 2 * (xm - gam ** 2) / (1 + gam ** 2)
                R_A = (gam - xm - D1 ** 2) / (1 - gam - D1 + D1 ** 2)
                D2 = (1 - gam - D1 + D1 ** 2) / (1 - R_A)
                D3 = 1 - D1 - D2
                Z = S / (2 * math.sqrt(m0))
                Q = 1.25 * (gam - D3 - D2 * R_A) / D1
                try:
                    P1 = D1 / Q * math.exp(-Z / Q)
                except OverflowError:
                    P1 = 0
                P2 = D2 * Z / R_A ** 2 * math.exp(-Z ** 2 / (2 * R_A ** 2))
                P3 = D3 * Z * math.exp(-Z ** 2 / 2)


                pS = (P1 + P2 + P3) / (2 * math.sqrt(m0))

                EP = math.sqrt(m4 / m2)

                N[iS] = EP * pS * 3.15e7
        return N

    def longLife(self):

#         mMatrix = fatSlope * np.ones(np.shape(Srange))
#         mMatrix[Srange <= self.FAT] = 5
#         logaMatrix = self.loga * np.ones(np.shape(Srange))
#         logaMatrix[Srange <= self.FAT] = self.loga2
#
#         Ni = np.power(10, (logaMatrix - np.multiply(mMatrix, np.log10(Srange))))
#         Ni[Srange <= self.FAT] = 1e100
        Ni = np.power(10, (self.loga - np.multiply(3, np.log10(Srange))))
        Ni[Ni > 1e7] = np.power(10, (self.loga2 -
                                     np.multiply(5,np.log10(Srange[Ni > 1e7]))))

        return Ni


    def save(self):
        global savePath
        global omegas

        with open(savePath + "Members/Fatigue/" + self.name + self.dirW +
                  ".csv", 'w') as fileStress:
            stressString = "memberID,{0}".format(str(int(self.memberNo)))
            stressString += ",Hs,{:.2f}".format(self.Hs)
            stressString += ",Tz,{:.2f}".format(self.Tz)
            stressString += "\nS [MPa],"
            for val in Srange: stressString += str(val) + ","
            stressString += "\nNS (dirlik) [1/s],"
            for val in self.NS: stressString += str(val) + ","
            stressString += "\nNi (long life),"
            for val in self.Ni: stressString += str(val) + ","
            stressString += "\nA Spectrum,"
            for val in self.accelResponse: stressString += str(val) + ","
            stressString += "\nS Spectrum:,"
            for val in self.spectrum: stressString += str(val) + ","
            fileStress.write(stressString)

class Member:

    def __init__(self, number, FAT=32, gamma=[0, 0, 0, 0]):
        self.number = number
        self.FAT = FAT
        self.gamma = gamma

# jons can either be a spectrum (JONSWAP object) or just an array

def directionalResponse(rao, jons, dirW, wavePhase=-1, directionality=True):
    global mus
    global deltaMu
```

```
    global deltaOmega
    global omegas
    indexMu = np.argmax(mus == dirW)

    #dirRad = dirW / 180 * math.pi
    cosFactor = 0
    cosCheck = 0

    directions = (np.arange(-0.5 * math.pi, 0.5 * math.pi, deltaMu / 180 * math.pi))
    respSpec = np.zeros((len(directions), len(omegas))) * 1j

    for iMu, mu in enumerate(directions):
        if directionality:
            cosFactor = math.pow(math.cos(mu), 2) * 2 / math.pi * \
                (mus[1] - mus[0]) / 180 * math.pi
            cosCheck += cosFactor
            #print(cosCheck)
        else:
            if round(mu,1) == 0.0:
                cosFactor = 1
            else:
                cosFactor = 0

        if wavePhase == -1:
            #plt.plot(rao.val(mus[abs(indexMu + iMu - 6)])[0], label=str(int(mus[abs(indexMu + iMu - 6)])) + " "
            respSpec[iMu] = np.multiply(jons.spectrum * cosFactor,
                                        np.power(rao.val(mus[abs(indexMu + iMu - 6)])[0], 2)) + \
                                        rao.val(mus[abs(indexMu + iMu - 6)])[1] * 1j
        else:
            respSpec[iMu] = np.multiply(np.multiply(jons.spectrum * cosFactor,
                                                    np.power(rao.val(mus[abs(indexMu + iMu - 6)])[0], 2)),
                                     np.cos(rao.val(mus[abs(indexMu + iMu - 6)])[1] + wavePhase))

    #exit(0)
    #plt.plot(omegas, np.sum(respSpec, 0))
    return np.abs(respSpec.real)

def solveProb(m0, pAccept, scatterValues, step=0.1):
    pa = 1
    val = 0.5
    stepVal = step
    p = np.zeros(scatterValues.shape)
    pMap = np.zeros(scatterValues.shape)
    pMap80 = pMap.copy
    pMap10 = pMap.copy
    sw80 = True
    sw10 = True
#     print("Solving")
    m0[m0 == 0] = np.max(m0) / 10000000
    while pa >= (1 - pAccept):
        #t = tim.time()
        p_r = np.exp(np.divide(-val ** 2, 2 * m0))

        p_r[p_r > 1] = 0
        p = np.multiply(scatterValues, p_r)
        p [0,:] = 0
        pa = np.sum(p)
        val += stepVal


        if pa < 0.80 and sw80:
            sw80=False
            pMap80 = p.copy()
            val80 = val
            p80 = pa

        if pa < 0.10 and sw10:
            sw10=False
            pMap10 = p.copy()
            val10 = val
            p10 = pa

        if pa <= 0.08:
```

```
            stepVal = 0.005

#         print(pa)
    pMap = p
    mask = pMap > 1e-4
    mask = mask.reshape(np.shape(scatterValues))
    return ([val, val10, val80], [pa, p10, p80], [Map(values=pMap, name="Prob"),
                                                  Map(values=pMap10, name="Prob10"),
                                                  Map(values=pMap80, name="Prob80")], mask)


def main(rootPath, code, par, scatterPar, scatterCoastPar):

    global omegas
    global gamma; gamma = 3.3
    global g; g = 9.81
    global deltaOmega, deltaMu
    global mus, heights, periods
    global HEAD; HEAD = 0
    global BEAM; BEAM = 90
    designVersion = "6324JRv8"
    checkMultiframeVersion = "6324JRv9"  # designVersion
    global savePath; savePath = rootPath + "/Data/Final/" + designVersion + "/"
    global Srange
    global fatSlope
    global deltaStress

    exceedProb = par['pllRespMaxPOper']['value']
    secundaryExceedProb = par['pllProbAccelSecondary']['value']
    compFatRatio = par['pllCompressionFatigueRatio']['value']
    fatSlope = par['pllFatSlope']['value']


    omegas = loadPickle("omegas2121ton", rootPath + code['PATH_PRECAL'] + \
                        code['PRECAL_NAME'])
    deltaHeights = 0.5
    heights = np.arange(0, code['WEIBULL_H_END'], deltaHeights, dtype=np.ndarray)
    heights[0] = 0.5
    deltaPeriods = 0.5
    periods = np.arange(0, code['WEIBULL_T_END'], deltaPeriods, dtype=np.ndarray)
    periods[0] = 0.5
    heightsMesh = heights.reshape((len(heights), 1)) * np.ones((1, len(periods)))
    mus = loadPickle("mus2121ton", rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'])
    wavePhases = np.arange(0, 2*math.pi, 0.25 * math.pi)
    Srange = np.arange(2.5, 100, 5)
    deltaStress = Srange[1] - Srange[0]

    deltaMu = mus[2] - mus[1]
    deltaOmega = omegas[1] - omegas[0]

#     cf.buildScatter2(["prob","combProb"], fileName="CombinedProbabilityExampleNew", direction="BEAM", savePath=
#     cf.buildScatter2(["accel"], RP = [0], fileName="AccelResponseRP0New" , direction="BEAM",  savePath=savePath
#     exit(0)

    commentString = "2121ton"
    roll = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
               'Roll' + commentString + '.csv')
    pitch = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'Pitch' + commentString + '.csv')
    surge = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'Surge' + commentString + '.csv')
    sway = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
               'Sway' + commentString + '.csv')
    heave = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'Heave' + commentString + '.csv')
    yaw = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] + 'Yaw'
              + commentString + '.csv')

    ay = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
             'aYRP0' +commentString + '.csv')
    az = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
             'aZRP0' + commentString + '.csv')

```

```
    axRP0 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aXRP0' + commentString + '.csv')

    axRP1 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aXRP1' + commentString + '.csv')
    ayRP1 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aYRP1' + commentString + '.csv')
    azRP1 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aZRP1' + commentString + '.csv')
    axRP2 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aXRP2' + commentString + '.csv')
    ayRP2 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aYRP2' + commentString + '.csv')
    azRP2 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aZRP2' + commentString + '.csv')
    axRP3 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aXRP3' + commentString + '.csv')
    ayRP3 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aYRP3' + commentString + '.csv')
    azRP3 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aZRP3' + commentString + '.csv')
    axRP4 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aXRP4' + commentString + '.csv')
    ayRP4 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aYRP4' + commentString + '.csv')
    azRP4 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aZRP4' + commentString + '.csv')
    axRP5 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aXRP5' + commentString + '.csv')
    ayRP5 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aYRP5' + commentString + '.csv')
    azRP5 = RAO(rootPath + code['PATH_PRECAL'] + code['PRECAL_NAME'] +
                'aZRP5' + commentString + '.csv')

    scatterItems = []
    scatterSet = []

    for scat in scatterPar.values():
        scatterItems.append(int(float(scat['value'])))
        scatterSet.append(scat)
    for scat in scatterCoastPar.values():
        scatterItems.append(int(float(scat['value'])))
        scatterSet.append(scat)
    scatterItems = set(scatterItems)
#     scatterItems = [10, 47]

    #SWITCHES
    loopULS = False
    loopFatigue = False
    calculateResponses = False
    calculateULS = False
    calculateOperability = False
    readMultiframe = False
    postProc = True

    emptyArray = np.empty((len(heights), len(periods)), dtype=object)

    waveSpectra = np.empty((len(heights), len(periods)), dtype=object)

    rollResponseMap = [ResponseMap(name="Roll", dirW="HEAD", prob=exceedProb,
                               secundaryProb=secundaryExceedProb),
                       ResponseMap(name="Roll", dirW="BEAM", prob=exceedProb,
                               secundaryProb=secundaryExceedProb)]
    pitchResponseMap = [ResponseMap(name="Pitch", dirW="HEAD", prob=exceedProb,
                                secundaryProb=secundaryExceedProb),
                        ResponseMap(name="Pitch", dirW="BEAM", prob=exceedProb,
                                secundaryProb=secundaryExceedProb)]
    surgeResponseMap = [ResponseMap(name="Surge", dirW="HEAD", prob=exceedProb,
                                secundaryProb=secundaryExceedProb),
                        ResponseMap(name="Surge", dirW="BEAM", prob=exceedProb,
                                secundaryProb=secundaryExceedProb)]
    swayResponseMap = [ResponseMap(name="Sway", dirW="HEAD", prob=exceedProb,
                               secundaryProb=secundaryExceedProb),
```

```
                       ResponseMap(name="Sway", dirW="BEAM", prob=exceedProb,
                                   secundaryProb=secundaryExceedProb)]
    heaveResponseMap = [ResponseMap(name="Heave", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="Heave", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    yawResponseMap = [ResponseMap(name="Yaw", dirW="HEAD", prob=exceedProb,
                                  secundaryProb=secundaryExceedProb),
                      ResponseMap(name="Yaw", dirW="BEAM", prob=exceedProb, secundaryProb=secundaryExceedProb)]

    axRP0ResponseMap = [ResponseMap(name="AXRP0", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AXRP0", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    ayRP0ResponseMap = [ResponseMap(name="AYRP0", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AYRP0", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    azRP0ResponseMap = [ResponseMap(name="AZRP0", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AZRP0", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]

    axRP1ResponseMap = [ResponseMap(name="AXRP1", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AXRP1", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    ayRP1ResponseMap = [ResponseMap(name="AYRP1", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AYRP1", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    azRP1ResponseMap = [ResponseMap(name="AZRP1", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AZRP1", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]

    axRP2ResponseMap = [ResponseMap(name="AXRP2", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AXRP2", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    ayRP2ResponseMap = [ResponseMap(name="AYRP2", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AYRP2", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    azRP2ResponseMap = [ResponseMap(name="AZRP2", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AZRP2", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]

    axRP3ResponseMap = [ResponseMap(name="AXRP3", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AXRP3", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    ayRP3ResponseMap = [ResponseMap(name="AYRP3", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AYRP3", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    azRP3ResponseMap = [ResponseMap(name="AZRP3", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AZRP3", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]

    axRP4ResponseMap = [ResponseMap(name="AXRP4", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AXRP4", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    ayRP4ResponseMap = [ResponseMap(name="AYRP4", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AYRP4", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    azRP4ResponseMap = [ResponseMap(name="AZRP4", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AZRP4", dirW="BEAM", prob=exceedProb,
```

```
                                    secundaryProb=secundaryExceedProb)]

    axRP5ResponseMap = [ResponseMap(name="AXRP5", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AXRP5", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    ayRP5ResponseMap = [ResponseMap(name="AYRP5", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AYRP5", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    azRP5ResponseMap = [ResponseMap(name="AZRP5", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AZRP5", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]

    ayRelResponseMap = [ResponseMap(name="AYRel", dirW="HEAD", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb),
                        ResponseMap(name="AYRel", dirW="BEAM", prob=exceedProb,
                                    secundaryProb=secundaryExceedProb)]
    #azRelResponseMap = [ResponseMap(name="AZRel", dirW="HEAD", prob=exceedProb), ResponseMap(name="AZRel", dirW=

    ULSfile = open(savePath + "ULS/ULS.csv", 'w')
    ULSfile.write("dir,y/z,area,RP0aX,RP0aY,RP0aZ,RP1aX,RP1aY,RP1aZ,RP2aX," +
                  "RP2aY,RP2aZ,RP3aX,RP3aY,RP3aZ,RP4aX,RP4aY,RP4aZ,RP5aX," +
                  "RP5aY,RP5aZ,roll,pitch,Hs,U\n")
    ULSfile.close()

    ###########################################################################
    # SECOND ITERATION LOOP
    ###########################################################################
    if loopULS:
        if calculateResponses:
            for iH, Hs in enumerate(heights):
                for iT, Tz in enumerate(periods):
                    waveSpectra[iH, iT] = JONSWAP(Hs, Tz)
                    for iD, dirW in enumerate((HEAD, BEAM)):
                        print("\rULS:Responses for Hs:{0} Tz:{1}".format(str(Hs),
                                                                         str(Tz)))

                        #######################################################
                        # MOTIONS AND ACCELERATIONS
                        #######################################################

                        #######################################################
                        # calc resp
                        rollResponseMap[iD].spectra.values[iH, iT] =\
                            Response(directionalResponse(roll,
                                                         waveSpectra[iH, iT],
                                                         dirW,
                                                         directionality=True),
                                     name="Roll", Hs=Hs, Tz=Tz, dirW=dirW)
                        pitchResponseMap[iD].spectra.values[iH, iT] =\
                            Response(directionalResponse(pitch,
                                                         waveSpectra[iH, iT],
                                                         dirW,
                                                         directionality=True),
                                     name="Pitch", Hs=Hs, Tz=Tz, dirW=dirW)
                        surgeResponseMap[iD].spectra.values[iH, iT] =\
                            Response(directionalResponse(surge,
                                                        waveSpectra[iH, iT],
                                                        dirW,
                                                        directionality=True),
                                     name="Surge", Hs=Hs, Tz=Tz, dirW=dirW)
                        swayResponseMap[iD].spectra.values[iH, iT] =\
                            Response(directionalResponse(sway,
                                                         waveSpectra[iH, iT],
                                                         dirW,
                                                         directionality=True),
                                     name="Sway", Hs=Hs, Tz=Tz, dirW=dirW)
                        heaveResponseMap[iD].spectra.values[iH, iT] =\
                            Response(directionalResponse(heave,
                                                         waveSpectra[iH, iT],
                                                         dirW,
```

```
                                        directionality=True),
                    name="Heave", Hs=Hs, Tz=Tz, dirW=dirW)
                yawResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(yaw,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="Yaw", Hs=Hs, Tz=Tz, dirW=dirW)

                axRP0ResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(axRP0,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AXRP0", Hs=Hs, Tz=Tz, dirW=dirW)
                ayRP0ResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(ay,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AYRP0", Hs=Hs, Tz=Tz, dirW=dirW)
                azRP0ResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(az,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AZRP0", Hs=Hs, Tz=Tz, dirW=dirW)

                ayRelResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(ay,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AYRP0", Hs=Hs, Tz=Tz, dirW=dirW)

                axRP1ResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(axRP1,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AXRP1", Hs=Hs, Tz=Tz, dirW=dirW)
                ayRP1ResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(ayRP1,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AYRP1", Hs=Hs, Tz=Tz, dirW=dirW)
                azRP1ResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(azRP1,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AZRP1", Hs=Hs, Tz=Tz, dirW=dirW)

                axRP2ResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(axRP2,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AXRP2", Hs=Hs, Tz=Tz, dirW=dirW)
                ayRP2ResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(ayRP2,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AYRP2", Hs=Hs, Tz=Tz, dirW=dirW)
                azRP2ResponseMap[iD].spectra.values[iH, iT] =\
                    Response(directionalResponse(azRP2,
                                        waveSpectra[iH, iT],
                                        dirW,
                                        directionality=True),
                    name="AZRP2", Hs=Hs, Tz=Tz, dirW=dirW)

```

```
            axRP3ResponseMap[iD].spectra.values[iH, iT] =\
                Response(directionalResponse(axRP3,
                                             waveSpectra[iH, iT],
                                             dirW,
                                             directionality=True),
                         name="AXRP3", Hs=Hs, Tz=Tz, dirW=dirW)
            ayRP3ResponseMap[iD].spectra.values[iH, iT] =\
                Response(directionalResponse(ayRP3,
                                             waveSpectra[iH, iT],
                                             dirW,
                                             directionality=True),
                         name="AYRP3", Hs=Hs, Tz=Tz, dirW=dirW)
            azRP3ResponseMap[iD].spectra.values[iH, iT] =\
                Response(directionalResponse(azRP3,
                                             waveSpectra[iH, iT],
                                             dirW,
                                             directionality=True),
                         name="AZRP3", Hs=Hs, Tz=Tz, dirW=dirW)

            axRP4ResponseMap[iD].spectra.values[iH, iT] =\
                Response(directionalResponse(axRP4,
                                             waveSpectra[iH, iT],
                                             dirW,
                                             directionality=True),
                         name="AXRP4", Hs=Hs, Tz=Tz, dirW=dirW)
            ayRP4ResponseMap[iD].spectra.values[iH, iT] =\
                Response(directionalResponse(ayRP4,
                                             waveSpectra[iH, iT],
                                             dirW,
                                             directionality=True),
                         name="AYRP4", Hs=Hs, Tz=Tz, dirW=dirW)
            azRP4ResponseMap[iD].spectra.values[iH, iT] =\
                Response(directionalResponse(azRP4,
                                             waveSpectra[iH, iT],
                                             dirW,
                                             directionality=True),
                         name="AZRP4", Hs=Hs, Tz=Tz, dirW=dirW)

            axRP5ResponseMap[iD].spectra.values[iH, iT] =\
                Response(directionalResponse(axRP5,
                                             waveSpectra[iH, iT],
                                             dirW,
                                             directionality=True),
                         name="AXRP5", Hs=Hs, Tz=Tz, dirW=dirW)
            ayRP5ResponseMap[iD].spectra.values[iH, iT] =\
                Response(directionalResponse(ayRP5,
                                             waveSpectra[iH, iT],
                                             dirW,
                                             directionality=True),
                         name="AYRP5", Hs=Hs, Tz=Tz, dirW=dirW)
            azRP5ResponseMap[iD].spectra.values[iH, iT] =\
                Response(directionalResponse(azRP5,
                                             waveSpectra[iH, iT],
                                             dirW,
                                             directionality=True),
                         name="AZRP5", Hs=Hs, Tz=Tz, dirW=dirW)

####################################################################
# CALC AND SAVE MAPS
####################################################################
for iD in (0, 1):
    rollResponseMap[iD].calc()
    pitchResponseMap[iD].calc()
    surgeResponseMap[iD].calc()
    swayResponseMap[iD].calc()
    heaveResponseMap[iD].calc()
    axRP0ResponseMap[iD].calc()
    ayRP0ResponseMap[iD].calc()
    azRP0ResponseMap[iD].calc()
    axRP1ResponseMap[iD].calc()
    ayRP1ResponseMap[iD].calc()
    azRP1ResponseMap[iD].calc()
    axRP2ResponseMap[iD].calc()
```

```
                ayRP2ResponseMap[iD].calc()
                azRP2ResponseMap[iD].calc()
                axRP3ResponseMap[iD].calc()
                ayRP3ResponseMap[iD].calc()
                azRP3ResponseMap[iD].calc()
                axRP4ResponseMap[iD].calc()
                ayRP4ResponseMap[iD].calc()
                azRP4ResponseMap[iD].calc()
                axRP5ResponseMap[iD].calc()
                ayRP5ResponseMap[iD].calc()
                azRP5ResponseMap[iD].calc()

                ayRelResponseMap[iD].calc()

                print("ULS:saving")
                # save all maps
                rollResponseMap[iD].save()
                pitchResponseMap[iD].save()
                surgeResponseMap[iD].save()
                swayResponseMap[iD].save()
                heaveResponseMap[iD].save()
                axRP0ResponseMap[iD].save()
                ayRP0ResponseMap[iD].save()
                azRP0ResponseMap[iD].save()
                axRP1ResponseMap[iD].save()
                ayRP1ResponseMap[iD].save()
                azRP1ResponseMap[iD].save()
                axRP2ResponseMap[iD].save()
                ayRP2ResponseMap[iD].save()
                azRP2ResponseMap[iD].save()
                axRP3ResponseMap[iD].save()
                ayRP3ResponseMap[iD].save()
                azRP3ResponseMap[iD].save()
                axRP4ResponseMap[iD].save()
                ayRP4ResponseMap[iD].save()
                azRP4ResponseMap[iD].save()
                axRP5ResponseMap[iD].save()
                ayRP5ResponseMap[iD].save()
                azRP5ResponseMap[iD].save()

                ayRelResponseMap[iD].save()
                #plt.show()

        else:
            print("ULS:loading")
            for iD, dirW in enumerate((HEAD, BEAM)):
                rollResponseMap[iD].load()
                pitchResponseMap[iD].load()
                surgeResponseMap[iD].load()
                swayResponseMap[iD].load()
                heaveResponseMap[iD].load()
                axRP0ResponseMap[iD].load()
                ayRP0ResponseMap[iD].load()
                azRP0ResponseMap[iD].load()
                ayRP1ResponseMap[iD].load()
                axRP1ResponseMap[iD].load()
                azRP1ResponseMap[iD].load()
                axRP2ResponseMap[iD].load()
                ayRP2ResponseMap[iD].load()
                azRP2ResponseMap[iD].load()
                axRP3ResponseMap[iD].load()
                ayRP3ResponseMap[iD].load()
                azRP3ResponseMap[iD].load()
                axRP4ResponseMap[iD].load()
                ayRP4ResponseMap[iD].load()
                azRP4ResponseMap[iD].load()
                axRP5ResponseMap[iD].load()
                ayRP5ResponseMap[iD].load()
                azRP5ResponseMap[iD].load()

                ayRelResponseMap[iD].load()
#                 ayRelResponseMap[iD].spectra.values[0, 10].plot()
#                 plt.show()
```

```
#                 print(s.spectra.values[10, 10].sign)
#                 print(self.spectra.values[10, 10].sign)
#                 print(self.sign)
#                 print(s.sign)

        #######################################################################
        # Stresses in pedestal
        #######################################################################

        stressResponseMapXX = [Map(values=None, dirW="HEAD", name="StressXX"),
                               Map(values=None, dirW="BEAM", name="StressXX")]
        stressResponseMapYY = [Map(values=None, dirW="HEAD", name="StressYY"),
                               Map(values=None, dirW="BEAM", name="StressYY")]
        for iD, dirW in enumerate((HEAD, BEAM)):
            stressArrayXX = np.zeros((len(heights), len(periods)))
            stressArrayYY = np.zeros((len(heights), len(periods)))
            for iH, Hs in enumerate(heights):
                for iT, Tz in enumerate(periods):
                    stressArrayXX[iH, iT], stressArrayYY[iH, iT] = \
                    pedestalStress([axRP0ResponseMap[iD].sign.values[iH, iT],
                                    ayRP0ResponseMap[iD].sign.values[iH, iT],
                                    azRP0ResponseMap[iD].sign.values[iH, iT]],
                                   [axRP1ResponseMap[iD].sign.values[iH, iT],
                                    ayRP1ResponseMap[iD].sign.values[iH, iT],
                                    azRP1ResponseMap[iD].sign.values[iH, iT]],
                                   [axRP2ResponseMap[iD].sign.values[iH, iT],
                                    ayRP2ResponseMap[iD].sign.values[iH, iT],
                                    azRP2ResponseMap[iD].sign.values[iH, iT]],
                                   rollResponseMap[iD].sign.values[iH, iT],
                                   pitchResponseMap[iD].sign.values[iH, iT],
                                   windSpeed(Hs),
                                   par)
            stressResponseMapXX[iD].values = stressArrayXX / 1000000
            stressResponseMapYY[iD].values = stressArrayYY / 1000000

            stressResponseMapXX[iD].save(subfolder = "Responses/")
            stressResponseMapYY[iD].save(subfolder = "Responses/")
            #stressResponseMapXX[iD].plot(levels=[30, 40, 50, 60 ,70, 80])


        #######################################################################
        # MASKS
        #######################################################################
        motionMap = np.empty(2, dtype=object)
        rollMap = motionMap.copy()
        floodMap = motionMap.copy()
        pitchMap = motionMap.copy()
        stressMap = motionMap.copy()

        for iD, dirW in enumerate((HEAD, BEAM)):
            motionMap[iD] = Map(values=(ayRelResponseMap[iD].m0 <=
                                         par['pllRMSAccelWork']['value'] / 1000) *
                                (azRP0ResponseMap[iD].m0 <=
                                 par['pllRMSAccelWork']['value'] / 1000),
                                dirW=dirW,
                                name="MaskMotion")
            rollMap[iD] = Map(values=rollResponseMap[iD].sign.values <=
                              par['crMaxHeelGrab']['value'],
                              dirW=dirW,
                              name="MaskRoll")
            pitchMap[iD] = Map(values=pitchResponseMap[iD].sign.values <=
                               par['crMaxTrimGrab']['value'],
                               dirW=dirW,
                               name="MaskPitch")
            floodMap[iD] = Map(values=((pitchResponseMap[iD].max.values <=
                                        par['pllMaxHeel']['value']) *
                               (rollResponseMap[iD].max.values <=
                                par['pllMaxHeel']['value'])),
                               dirW=dirW,
                               name="MaskFlood")
            stressMap[iD] = Map(values=((stressResponseMapXX[iD].values <=
                                         par['pllAllowableStressFAT']['value']) *
```

```
                                (stressResponseMapYY[iD].values <=
                                 par['pllAllowableStressFAT']['value'])),
                                dirW=dirW,
                                name="MaskStress")

            motionMap[iD].save(subfolder="Masks/")
            rollMap[iD].save(subfolder="Masks/")
            pitchMap[iD].save(subfolder="Masks/")
            floodMap[iD].save(subfolder="Masks/")
            stressMap[iD].save(subfolder="Masks/")
#             #motionMap[iD].plot()
#             #rollMap[iD].plot()
#             #pitchMap[iD].plot()
#             #floodMap[iD].plot()
#             stressMap[iD].plot()
#             #print(stressResponseMapXX[iD].values)
#             #exit(0)
#             plt.show()
#         exit(0)

        ######################################################################
        # SCATTER
        ######################################################################

        areas = list()
        newAreas = areas.copy()
        ULSArea = np.zeros((2, 2))
        ULSValues = np.zeros((2, 2, 22))
        ULSMax = np.zeros((2, 2))
        for iA, area in enumerate(scatterSet):
            areaNumber = int(float(area['value']))
#             scatterValues = np.genfromtxt(rootPath + code['PATH_SCATTER'] + '/Scatter' + str(areaNumber) + '.cs
#             print(np.sum(scatterValues))
#             #newScatterValues = cf.rebinOfficial(scatterValues, (len(heights), len(periods)))
#             newScatterValues = cf.rebin(scatterValues, (len(heights), len(periods)))
#             print(np.sum(newScatterValues))
#             newScatterValues = newScatterValues / np.sum(newScatterValues)
#             print(np.sum(newScatterValues))
#             exit(0)
#             if np.sum(newScatterValues) < 0.999:
#                 print("ERROR sum probability values of area " + str(areaNumber) + " is " + str(np.sum(scatterVa
            value = scatterSet[iA]
            scatterArray = cf.scatter2(heights, periods, value['extra'][1],
                                         value['extra'][2],
                                         value['extra'][3],
                                         value['extra'][4],
                                         value['extra'][5],
                                         value['extra'][6],
                                         deltaHeights,
                                         deltaPeriods, returnfHs=False)

#             print(scatterArray)
#             print(np.sum(scatterArray))
#             plt.contour(heights, periods, scatterArray.transpose(), levels=[1e-7, 1e-6, 1e-5])
#             plt.show()
#             exit(0)

            newScatterValues = scatterArray
            currentScatter = Scatter(newScatterValues, areaNumber)
#             currentScatter.plot(maskMap=motionMap[1], levels = [1e-6, 1e-5, 1e-4])
#             motionMap[1].plot()
#             plt.show()
#             exit(0)
            if calculateULS or calculateOperability:
                ##################################################################
                # PARKABILITY AND WORKABILITY
                ##################################################################
                for iD, dirW in enumerate((HEAD, BEAM)):
                    workability = motionMap[iD].values * rollMap[iD].values * \
                    pitchMap[iD].values * floodMap[iD].values * stressMap[iD].values
                    parkability = floodMap[iD].values * stressMap[iD].values
                    currentScatter.operability[iD] = np.sum(workability * newScatterValues)
                    currentScatter.parkability[iD] = np.sum(parkability * newScatterValues)
```

```
                    currentScatter.saveOperability()


            if calculateULS:
                ################################################################
                # ULS
                ################################################################

                for iD, dirW in enumerate((HEAD, BEAM)):
                    parkability = floodMap[iD].values * stressMap[iD].values
                    for iYZ in (0, 1):
                        if iYZ == 0: # =Y
                            m0array = ayRelResponseMap[iD].m0
                        else:
                            m0array = azRP0ResponseMap[iD].m0
                        [[currentScatter.primaryVal[iD, iYZ],
                          currentScatter.primaryVal10[iD, iYZ],
                          currentScatter.primaryVal80[iD, iYZ]],
                         [currentScatter.primaryProb[iD, iYZ],
                          currentScatter.primaryProb10[iD, iYZ],
                          currentScatter.primaryProb80[iD, iYZ]],
                         [currentScatter.primaryMap[iD, iYZ],
                          currentScatter.primaryMap10[iD, iYZ],
                          currentScatter.primaryMap80[iD, iYZ]],
                         currentScatter.primaryProbMask[iD, iYZ]] = \
                            solveProb(m0array,
                                      par['pllProbAccel']['value'],
                                      np.multiply(newScatterValues, parkability))
#                         plt.title(areaNumber)
#                         currentScatter.plot(levels=[1e-5, 1e-6, 1e-7])
#                         currentScatter.primaryMap80[iD, iYZ].plot(levels=[1e-6])
#                         currentScatter.primaryMap10[iD, iYZ].plot(levels=[1e-6])
#                         currentScatter.primaryMap[iD, iYZ].plot(levels=[1e-6])
#                         plt.contour(heights, periods, currentScatter.primaryProbMask[iD, iYZ].transpose(), colo
#                         plt.show()
#                         exit(0)
        #                     plt.contour(heights, periods, currentScatter.primaryProbMask[iD, iYZ].transpose(),
        #                     plt.show()
        #                     exit(0)
                        currentScatter.ULS[iD, iYZ] = \
                            [np.max(np.multiply(axRP0ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(ayRP0ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(azRP0ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(axRP1ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(ayRP1ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(azRP1ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(axRP2ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(ayRP2ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(azRP2ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(axRP3ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(ayRP3ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(azRP3ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(axRP4ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(ayRP4ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(azRP4ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(axRP5ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(ayRP5ResponseMap[iD].max90.values,
```

```
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(azRP5ResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(rollResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(pitchResponseMap[iD].max90.values,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             np.max(np.multiply(heightsMesh,
                                                currentScatter.primaryProbMask[iD, iYZ])),
                             windSpeed(np.max(np.multiply(heightsMesh,
                                                          currentScatter.primaryProbMask[iD, iYZ])))
                            ]  # roll, pitch, Hs, U
                currentScatter.setPrimaryULS()
                currentScatter.save()

                for iD, dirW in enumerate((HEAD, BEAM)):
                    for iYZ in (0, 1):
                        print(":")
                        print(ULSMax[iD, iYZ])
                        print(currentScatter.primaryVal[iD, iYZ])
                        if ULSMax[iD, iYZ] < currentScatter.primaryVal[iD, iYZ]:
                            ULSArea[iD, iYZ] = areaNumber
                            ULSValues[iD, iYZ] = currentScatter.ULS[iD, iYZ]
                            ULSMax[iD, iYZ] = currentScatter.primaryVal[iD, iYZ]
            newAreas.append(currentScatter)

        #print(ULSArea)

        if calculateULS:
            ULSfile = open(savePath + "ULS/LoadCases.csv", 'w')
            ULSfile.write(",RP0aX,RP0aY/aYRel,RP0aZ,RP1aX,RP1aY,RP1aZ,RP2aX," +
                          "RP2aY,RP2aZ,RP3aX,RP3aY,RP3aZ,RP4aX,RP4aY,RP4aZ," +
                          "RP5aX,RP5aY,RP5aZ,roll,pitch,Hs,U\n")
            ULSCase = np.empty((2,2), dtype=str)
            ULSCase[0, 0] = "YH"
            ULSCase[0, 1] = "ZH"
            ULSCase[1, 0] = "YB"
            ULSCase[1, 1] = "ZB"

            for iD, dirW in enumerate((HEAD, BEAM)):
                for iYZ in (0, 1):
                    ULSfile.write(ULSCase[iD, iYZ] + ",")
                    for val in ULSValues[iD, iYZ]: ULSfile.write(str(round(val, 2)) + ",");
                    ULSfile.write("\n")
            ULSfile.close()
        ########################################################################
        # FoS
        ########################################################################
        # TODO: include FoS here, then write to xlsx

    ############################################################################
    # SECOND ITERATION LOOP
    ############################################################################
    if loopFatigue:
        savePath = rootPath + "/Data/Final/" + checkMultiframeVersion + "/"
        # TODO: write function that loads the response arrays again....
        ########################################################################
        # FATIGUE
        ########################################################################
        memberData = np.loadtxt(savePath + "Members/MemberData.csv",
                                delimiter=",",
                                skiprows=1,
                                ndmin=2).transpose() #[ID, FAT, gammaB, gammaN]
#         print("Fatigue member Data:")
#         print(memberData)
        members = memberData[0, :]
        loga = memberData[1, :]
        ##########################################
        # WARNING: in the old version, the mass is taken into account here, but should be done in the 'gamma' dir
        gammaB = memberData[2, :] #* par['pllForceRatio']['value'] * par['pcalMassJib']['value']
        gammaN = memberData[3, :] #* par['pllForceRatio']['value'] * par['pcalMassJib']['value'] * par['pllCompre
        FAT = memberData[4,:]
        loga2 = memberData[5,:]
```

```
        Sb = np.empty((len(heights), len(periods), len(members)), dtype=object)
        Sn = Sb.copy()
        iD = 1 # BEAM!!
#         ayRelResponseMap[0].spectra.values[0, 10].plot()
#         print(ayRelResponseMap[0].spectra.values[0, 5].spectrum.spectrum)
#         plt.show()
#         ayRelResponseMap[0].spectra.values[0, 1].plot()
#         ayRelResponseMap[0].spectra.values[0, 8].plot()
#         ayRelResponseMap[0].spectra.values[0, 10].plot()
#         print(ayRelResponseMap[1].spectra.values[0, 0].spectrum.spectrum)
#         plt.show()
#         exit(0)

        for iD, dirW in enumerate(("HEAD", "BEAM")):
            for iH, Hs in enumerate(heights):
                for iT, Tz in enumerate(periods):
                    if iT > 2:
                        try:
                            aySpec = ayRelResponseMap[iD].spectra.values[iH, iT].spectrum.spectrum
                            azSpec = azRP0ResponseMap[iD].spectra.values[iH, iT].spectrum.spectrum
                        except AttributeError as e:
                            print("empty spectrum (aRel) found in Hs:{0}({2}) Tz:{1}({3}) ".format(Hs, Tz, iH, iT
#                             ayRelResponseMap[iD].spectra.values[iH, iT].plot()
#                             exit(0)
                        #print(ayRelResponseMap[iD].spectra.values[iH, iT].spectrum.spectrum)
                        #plt.show()

#                             print("empty spectrum (aRel) found in Hs:{0} Tz:{1} ".format(Hs, Tz, fmt="%.2f"))
                    else:
                        aySpec = np.zeros((np.shape(omegas)))
                        azSpec = np.zeros((np.shape(omegas)))

                    for iMF, memberID in enumerate(members):
                        if isinstance(ayRelResponseMap[iD].spectra.values[iH, iT], Response) \
                            and isinstance(azRP0ResponseMap[iD].spectra.values[iH, iT], Response):
                            Sb[iH, iT, iMF] = StressResponse(spectrum=aySpec,
                                                             factor = ((1 + compFatRatio) / 2) **2 * gammaB[iMF]
                                                             memberNo=memberID,
                                                             Hs=Hs,
                                                             Tz=Tz,
                                                             gammaB=gammaB[iMF],
                                                             gammaN=0,
                                                             loga=loga[iMF],
                                                             loga2=loga2[iMF],
                                                             FAT=FAT[iMF],
                                                             dirW=dirW,
                                                             comment="Bend")

                            Sn[iH, iT, iMF] = StressResponse(spectrum=azSpec,
                                                             factor = compFatRatio **2 * gammaN[iMF] **2,
                                                             memberNo=memberID,
                                                             Hs=Hs,
                                                             Tz=Tz,
                                                             gammaB=0,
                                                             gammaN=gammaN[iMF],
                                                             loga=loga[iMF],
                                                             loga2=loga2[iMF],
                                                             FAT=FAT[iMF],
                                                             dirW=dirW,
                                                             comment="Normal")

                            if iH in (5, 10, 15, 20):
                                if iT in (12, 14):
                                    if iMF in (0, 2, 7, 14, 20):
                                        Sb[iH, iT, iMF].save()
                                        Sn[iH, iT, iMF].save()


            fatigueTime = np.loadtxt(savePath + "Members/FatigueTime.csv",
                                     delimiter=",",
                                     skiprows=1,
                                     ndmin=2).transpose() #[ID, FAT, gammaB, gammaN]
            maxFatCases = np.max(fatigueTime[0, :])
```

```python
        fatCaseDamage = np.zeros((maxFatCases, len(members)))
        fatTowDamage = np.zeros(len(members))

        parkabilityMap = floodMap[iD].values * stressMap[iD].values
        SbMap = np.zeros(np.shape(Sb[:,:,0]))
        SnMap = SbMap.copy()
#         print(Sb[10, 10, 0].R)
        for iM, memberID in enumerate(members):
            for iH, Hs in enumerate(heights):
                for iT, Tz in enumerate(periods):
                    try:
                        SbMap[iH, iT] = Sb[iH, iT, iM].R
                    except AttributeError:
                        print("no damage found for Hs:{0} Tz:{1} member:{2} sigmaB".format(Hs, Tz, memberID))
                    try:
                        SnMap[iH, iT] = Sn[iH, iT, iM].R
                    except AttributeError:
                        print("no damage found for Hs:{0} Tz:{1} member:{2} sigmaN".format(Hs, Tz, memberID))

            SbMap[np.isnan(SbMap)] = 0
            SnMap[np.isnan(SnMap)] = 0
#             print(SbMap[0::4])
#             print(SnMap[0::4])
#             print(SbMap[15, 12] * 3.16e7 * 20)


#             SbMap = ReturnDamage(Sb[:, :, iM])

#             SnMap = ReturnDamage(Sn[:, :, iM])

            for iA, areaNumber in enumerate(fatigueTime[1,:]):
#                 print("area: {0}".format(areaNumber))
                probMap = [area.values for area in newAreas if area.areaNumber == areaNumber][0]
                fatTime = fatigueTime[2, iA]
#                 print("time: {0}s".format(fatTime))
#                 damageB = np.sum(np.multiply(np.multiply(parkabilityMap, SbMap), probMap)) * fatTime

                parkProbMap = np.multiply(parkabilityMap, probMap)
                normalizedParkProbMap = np.divide(parkProbMap,
                                                  np.sum(parkProbMap)) # = will always moved into a sea state
                damageB = np.sum(np.multiply(normalizedParkProbMap,SbMap)) * fatTime
                damageN = np.sum(np.multiply(np.multiply(parkabilityMap, SnMap), probMap)) * fatTime
#                 print("damageB: {0}".format(damageB))

                fatCaseDamage[fatigueTime[0, iA] - 1, iM] += damageB
                fatCaseDamage[fatigueTime[0, iA] - 1, iM] += damageN


            ################################################################################
            # TOUGH SEA TRIP
            ################################################################################
            tripLength = 1/2/12 #2 weeks! -> year
            HsTrip = par['pllSignWaveHeightWeather']['value']
            TzTrip = par['pllMaxDeformationPeriod']['value']
            SbTrip = Sb[np.argmax(heights >= HsTrip), np.argmax(periods >= TzTrip), iM]
            SnTrip = Sn[np.argmax(heights >= HsTrip), np.argmax(periods >= TzTrip), iM]
#             plt.figure()
#             SbTrip.plot()
#             plt.legend()
#             plt.show()
            SbTrip.name += "TowingTrip"
            SnTrip.name += "TowingTrip"
            SbTrip.save()
            SnTrip.save()

            if SbTrip.R > 0:
                fatTowDamage[iM] += SbTrip.R * tripLength
            if SnTrip.R > 0:
                fatTowDamage[iM] += SnTrip.R * tripLength
            print(fatTowDamage)

        outputFatDamage = np.vstack((members, fatCaseDamage))
        topRow = np.c_[0:maxFatCases+1]
```

```
            outputFatDamage = np.hstack((topRow, outputFatDamage))
            header = "membersID/fatigueCases and cumulative damage"
            np.savetxt(savePath + "Members/FatigueDamage" + dirW + ".csv",
                       np.round(outputFatDamage.transpose(), 2), delimiter=",", fmt="%.2f", header=header)
            header = "membersID/Towing and cumulative damage"
            np.savetxt(savePath + "Members/FatigueDamageTowingTrip" + dirW +".csv",
                       np.round(np.vstack((members, fatTowDamage)).transpose(), 2),
                       delimiter=",",
                       fmt="%.2f",
                       header=header)

    if readMultiframe:
        multiframe(rootPath=rootPath, code=code,
                   inputName=checkMultiframeVersion,
                   name=checkMultiframeVersion + "MultiframeOutput")

    if postProc:
        ""
        mu = mus[6]
        rao = [np.loadtxt(rootPath + code['PATH_PRECAL'] + "6324Surge2121ton.csv", delimiter = ",", comments="#",
               np.loadtxt(rootPath + code['PATH_PRECAL'] + "6324Sway2121ton.csv", delimiter = ",", comments="#",
               np.loadtxt(rootPath + code['PATH_PRECAL'] + "6324Heave2121ton.csv", delimiter = ",", comments="#",
               np.loadtxt(rootPath + code['PATH_PRECAL'] + "6324Yaw2121ton.csv", delimiter = ",", comments="#", s
               np.loadtxt(rootPath + code['PATH_PRECAL'] + "6324Pitch2121ton.csv", delimiter = ",", comments="#",
               np.loadtxt(rootPath + code['PATH_PRECAL'] + "6324Roll2121ton.csv", delimiter = ",", comments="#",

        #650,8
        mot = (0, 1, 2, 3, 4, 5) #surge, sway, ...
        motString = ("Surge", "Sway", "Heave", "Yaw", "Pitch", "Roll")
        lineStyle = ("red", "blue", "black", "black, dashed", "blue, dashed", "red, dashed")
#         print(rao[mot][np.where(rao[mot][:, 3] == mu), -1])
#         exit(0)
        #for iMu in (0, 3, 6):
        #    mu = mus[iMu]

        tikzString = tk.beginFigure()
        tikzString += tk.begin2Daxis(options='xmin=0, xmax=2.5,ymin=-200,ymax=200, grid=major', xLabel = '$\\omeg

        for iMot in mot:
            phase = rao[iMot][np.where(rao[iMot][:, 3] == mu), -1][0]
#             print(phase)
#             print(omegas)
            tikzString += tk.addPlotCoordinates(omegas, phase, lineStyle[iMot])
            tikzString += tk.addLegend(motString[iMot])

        tikzString += tk.addExtraLegend('\\pcalMassBallastedLong')
        tikzString += tk.addExtraLegend('$\\mu = ' + str(int(mu)) + 'deg$')

        tikzString += tk.end2Daxis()
        tikzString += tk.endFigure()


        with open(rootPath + code['PATH_TIKZ'] + "/PHASE-Mu" + str(int(mu)) + "-2121ton.tikz", 'w') as tikzFile:
            tikzFile.write(tikzString)



#         tikzString =[r'''
#             \begin{tikzpicture}
#                 \begin{axis}[axis lines= none, legend pos=south east, x=\textwidth/360, y=\textwidth/360,
#                 xmin=-180, xmax=180,
#                 ymin=-90, ymax=90]
#                 \newcommand*{\unit}{\textwidth/360}
#                 \newcommand*{\workradius}{5\unit}
#                 \node[anchor = south west, inner sep =0] at (-180,-90) {\includegraphics[trim = 2.25cm 2.8cm 1.
#                 width=\textwidth]{../Pictures/worldmapDNVSeaStates.PNG}};
#                 ''',
#                 r'''
#             \begin{tikzpicture}
#                 \begin{axis}[axis lines= none, legend pos=south east, x=\textwidth/360, y=\textwidth/360,
#                 xmin=-180, xmax=180,
#                 ymin=-90, ymax=90]
#                 \newcommand*{\unit}{\textwidth/360}
```

```
#                   \newcommand*{\workradius}{5\unit}
#                   \node[anchor = south west, inner sep =0] at (-180,-90) {\includegraphics[trim = 2.25cm 2.8cm 1.
#                   width=\textwidth]{../Pictures/worldmapDNVSeaStates.PNG}};
#                   '''
#               ]
#
#           for iA, area in enumerate(scatterCoastPar.values()):
#               areaNumber = int(float(area['value']))
#               areaOP = np.loadtxt(savePath + "/Operability/AreaScatter" +
#                                   str(areaNumber) + ".csv", delimiter=",", comments="#") # [[OPH, OPB], [PAH, PAE
#               areaB = float(area['extra'][7])
#               areaH = float(area['extra'][8])
#
#               for iD, dir in enumerate(("HEAD", "BEAM")):
#                   OPdeg = 360 * areaOP[0, iD]
#                   PAdeg = 360 * areaOP[1, iD]
#                   areasOptions = ['fill opacity= 0.7, fill=green', 'fill opacity= 0.7, fill=yellow']
#                   tikzString[iD] += r"\draw [" + areasOptions[1] + "] (" +  str(areaB) + r"," + str(areaH) + r")
#                   tikzString[iD] += r"\draw [" + areasOptions[0] + "] (" +  str(areaB) + r"," + str(areaH) + r")
#
#
#           tikzString[0] += tk.end2Daxis()
#           tikzString[1] += tk.end2Daxis()
#           tikzString[0] += tk.endFigure()
#           tikzString[1] += tk.endFigure()
#
#           with open(savePath + "/Operability/OperabilityMapHead.tikz", 'w') as f:
#               f.write(tikzString[0])
#           with open(rootPath + code['PATH_TIKZ'] +"/OperabilityMapHead.tikz", 'w') as f:
#               f.write(tikzString[0])
#           with open(savePath + "/Operability/OperabilityMapBeam.tikz", 'w') as f:
#               f.write(tikzString[1])
#           with open(rootPath + code['PATH_TIKZ'] +"/OperabilityMapBeam.tikz", 'w') as f:
#               f.write(tikzString[1])

#           multiframe(rootPath=rootPath, code=code, inputName=checkMultiframeVersion, name=checkMultiframeVersion
#           cf.buildScatter2(["prob","combProb"], fileName="CombinedProbabilityExampleNew", direction="HEAD", saveF
#           cf.buildScatter2(["accel"], RP = [0], fileName="AccelResponseRP0New" , direction="BEAM",  savePath=save
#           cf.buildScatter2(["accel"], RP = "Rel", fileName="AccelResponseRelativeNew" , direction="BEAM",  savePa
#           cf.buildScatter2(["accel", "workcrit"], RP = [5], fileName="AccelResponseRP0andCrit" , direction="BEAM"
#           cf.buildScatter2(["accel", "workcrit"], RP = [5], fileName="AccelResponseRP0andCrit" , direction="HEAD"
#           cf.fatigueOutput("StressResponseBend23_2.5_6.5TowingTripBEAM", rootPath, code, par, fileName="FatigueSt
#           cf.fatigueOutput("StressResponseBend1_2.5_6.5TowingTripBEAM", rootPath, code, par, fileName="FatigueStr
#           cf.buildScatter2(["workcrit"], fileName="Criteria", direction="HEAD", savePath=savePath, heights=height
#           cf.buildScatter2(["workcrit"], fileName="Criteria", direction="BEAM", savePath=savePath, heights=height
#           cf.buildScatter2(["accel", "workcrit"], RP = [4], fileName="CylinderSelectionResponseHead" , direction=
```