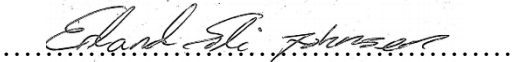




Universitetet
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Structural Engineering and Materials Science / Civil Engineering Structures	Spring semester, 2019 Open
Author: Erland Soli Johnsen	 (signature of author)
Programme coordinator: Supervisor(s):	Sudath Siriwardane Jasna B. Jakobsen (UiS) Bernardo Morais da Costa (External, Statens vegvesen)
Title of master's thesis: Dynamic Analysis and Finite Element Modeling of a Floating Bridge	
Credits: 30	
Keywords: Bjørnafjorden Floating Bridge Cable Stayed Bridge FEM Abaqus	Number of pages:97..... + supplemental material/other:51..... Stavanger,.....14.06/2019..... date/year

Abstract

With the aim of reducing the travel time of the coastal highway E39 by substituting ferries with bridges and tunnels, the National Public Road Administration project "Ferry Free E39" includes several fjord crossings demanding utilization of new technology and concepts. Crossing of the 4,5 km wide and up to 600 m depth Bjørnafjorden is designed with a floating bridge and cable stayed part, which is further studied in this thesis.

The long and slender structure with only fixed ends results in a structure with low stiffness suspect to both wind and wave loads, which in nature are dynamic. Assessing the dynamic properties and response is thus of high importance. Thesis emphasis is on the constructing of a finite element model which will be loaded with simplified buffeting wind loads.

Model geometry and properties are established in MATLAB, while the finite element code Abaqus is utilized for both basic eigenanalysis and the dynamic response analysis in the time-domain.

Due to the curvature of the bridge, a 4 dimensional wind simulation script provided by UiS postdoc Etienne Cheynet, which relaxes Taylor's hypothesis of frozen turbulence is utilized. The model is based on a modified decay coherence function utilizing two decay coefficients. Turbulent wind representing 100 year return period conditions, lasting for 1 hour, are simulated and a total of 20 turbulent wind fields are utilized in a corresponding number of dynamic analysis.

Several simplifications are done on the modeling of the floating bridge. Only simple terms of the buffeting terms is applied, in which the aerodynamic coefficients are not utilized as dependent on the yaw angle. Hydrodynamic effect are extensively simplified, excitation forces, damping, restoring terms nor frequency dependent added mass are included. Only the effect of a constant added mass provided by low frequency waves. However the first 5 eigenmodes obtained with eigenmodes in between 120 s and 16,7 s are similar to prior results. Implying behavior of the obtained model is accurate when analyzing the response governed by the first eigenmodes.

Large deviations in the simulated bridge response are noted by applying wind from several different directions in the 1 hour analysis. Forces are observed to be greatest in wind with cardinal direction 280° and 100°, in which the wind from west provides a large moment about the strong axis of the bridge girder, the latter providing a great compressive axial force in the bridge girder.

To achieve a proper model, implementation of several additional terms in the turbulence wind load application provided by the buffeting theory presented by Davenport. Along with implementation of fundamental hydrodynamics; excitation forces, frequency dependent added mass, damping and restoring terms to further study the coupled effect of hydrodynamic and aerodynamic loads.

This thesis outlines a method in which the stiffness and mass matrices obtained through Abaqus is adequate comparing to prior consultants reports, utilizing the method described will efficiently found the foundation further implementation of aerodynamics and hydrodynamics discussed above.

Preface

This thesis marks the final work of the master's programme Engineering Structures and Materials with specialization in Civil Engineering Structures by the University of Stavanger (UiS), written in cohesion to Statens vegvesen (NPRA).

Scope of the thesis includes constructing a Finite Element Model (FEM) of the floating bridge across Bjørnafjorden and analyzing the application of simplified dynamic loads provided turbulent wind fields, as part of the project Ferry Free E39. Which the model is based one of the alternatives which is under further studies to date by consultant companies, based on prior drawings and reports provided by NPRA. In which most work have been in cohesion to achieve a reliable FEM in Abaqus.

Guidance have been provided by Prof. Jasna B. Jakobsen at UiS along with external supervisor Bernardo Morais de Costa at NPRA. Trough work with the thesis researcher Ibuki Kusano at UiS have contributed in large extent by consultation of constructing the FEM in Abaqus.

Thanks to Post Doc. Etienne Cheynet at UiS which supplied the 4 dimensional turbulence wind simulation script in MATLAB.

Initial knowledge of Abaqus software were non-existent, and the user guide proved fortunately both long and thorough. A complex task indeed, but an intriguing topic to construct such a model; I've gained tremendous amount of knowledge in the field of finite element modeling. Thanks for the opportunity to take part with such insightful and patient people.

Table of Contents

1.	Introduction.....	1
1.1	Ferry Free E39.....	1
1.2	Objective.....	3
1.3	Limitations	3
2	State of Art.....	4
2.1	Floating Structures.....	4
2.2	Existing Floating Bridges.....	4
2.3	Cable Stayed Bridge	6
2.4	Bjørnafjorden Bridge Concepts	7
3	Theory	8
3.1	Dynamic Analysis.....	8
3.2	Finite Element Method	12
3.3	Hydrodynamic and Hydrostatic Effect	18
3.4	Aerodynamic Effect.....	20
4	Bridge Design	30
4.1	General.....	30
4.2	Bridge Girder	32
4.3	Cable Stayed Tower	34
4.4	Cables	36
4.5	Pontoons	37
4.6	Pontoon Columns	38
5	Modelling	39
5.1	General Definitions	39
5.2	MATLAB	39
5.3	Abaqus	40
5.4	Bridge Components	40
5.5	Turbulent Wind Field.....	49
5.6	Wind Load Application	51
5.7	Step Definition	53
5.8	Summary of Analysis	54
6	Results.....	55
6.1	Model Verification	55
6.2	Eigen Frequency and Eigenmode Analysis	58

6.3	Wind Field.....	72
6.4	Bridge Girder Response.....	81
7	Conclusion and Discussion	92
7.1	Result Considerations.....	92
7.2	Recomendation for Further Studies	95
8	Sources	96

List of Tables

Table 4-1: Vertical curvature	31
Table 4-2: Bridge elevation	31
Table 4-3: General bridge girder parameters.....	32
Table 4-4: Girder material properties.....	32
Table 4-5: Girder Cross Section properties	32
Table 4-6: Additional steel weight due to reinforcements.....	33
Table 4-7: Cable stay tower material properties	34
Table 4-8: Geometric specification of tower concrete elements.....	35
Table 4-9: Cable Properties	36
Table 4-10: Pontoon properties	38
Table 4-11: Pontoon column properties.....	38
Table 5-1: Units of measure utilized in model.....	39
Table 5-2: Constant added mass utilized	44
Table 5-3: Boundary conditions.....	47
Table 5-4: Connector element properties	48
Table 5-5: Wind velocity and return periods calculated according to Eurocode 1 [20]	49
Table 5-6: Directional reduction coefficients.....	50
Table 5-7: Decay coefficients utilized	51
Table 5-8: Example utilization of tabular amplitude speciation for load variation	52
Table 5-9: Summary of analysis sets.....	54
Table 6-1: Reaction Forces	56
Table 6-2: First 50 eigenfrequencies with added mass.....	59
Table 6-3: First 50 eigenfrequencies excluding added mass	60
Table 6-4: Magnitude of maximum values of bridge girder responses.....	81
Table 6-5: Maximum forces and moments observed	82

List of Figures

Figure 1-1: European Route E39 [5].....	1
Figure 1-2: The «golden coast”. Population along the planes Highway Route E39 [2].....	2
Figure 2-1: Xerxes bridge [4].....	4
Figure 2-2: Homer M Hadley- (left), Lacey V. Murrow-Memorial Bridge (right) [7].....	5
Figure 2-3: Development of Cable stayed bridges.....	6
Figure 2-4: Storm during construction of Helgeland bridge [10].....	6
Figure 3-1: Material or Solid or Hysteretic Damping [12].....	9
Figure 3-2: Newton-Rapshon Method Illustrated [15].....	14
Figure 3-3: Deifinition of a Euler-Bernouli beam along x-axis [13].....	15
Figure 3-4: Rayleigh damping [12].....	16
Figure 3-5: Marine orientations [16].....	19
Figure 3-6: Ship changes in floating position [17].....	19
Figure 3-7: Wind profile in the atmospheric boundary level [19].....	21
Figure 3-8: Sign convention.....	24
Figure 4-1: Plan view and the horizontal curvature [28].....	30
Figure 4-2: Vertical view of bridge elevation [28].....	30
Figure 4-3: Bridge Girder Cross Section [28].....	32
Figure 4-4: Cable Stay Tower [28].....	34
Figure 4-5: Pontoon type 4 side view [28].....	37
Figure 4-6: Pontoon type 4 top view [28].....	37
Figure 5-1: Bjørnafjorden FE Model in Abaqus CAE.....	39
Figure 5-2: Abaqus definition of beam local axis [14].....	41
Figure 5-3: Local rectangular coordinate system [14].....	43
Figure 5-4: Variable added mass [28].....	44
Figure 5-5: Calculated mean wind velocity.....	50
Figure 6-1: Initial deflection magnitude in meters all directions after applying gravitational load.....	55
Figure 6-2: Initial deflection Z direction in meters after applying gravitational load.....	55
Figure 6-3: Moment about the strong axis and axial forces.....	56
Figure 6-4: Stress points.....	57
Figure 6-5: P1.....	57
Figure 6-6: P2.....	57
Figure 6-7: P3.....	57
Figure 6-8: P4.....	57
Figure 6-9: First 100 eigenfrequencies including added mass.....	58
Figure 6-10: First 100 eigenfrequencies without added mass.....	58
Figure 6-11: Mode 1.....	62
Figure 6-12: Mode 2.....	62
Figure 6-13: Mode 3.....	62
Figure 6-14: Mode 4.....	62
Figure 6-15: Mode 5.....	62
Figure 6-16: Mode 6.....	62
Figure 6-17: Mode 7.....	63
Figure 6-18: Mode 8.....	63
Figure 6-19: Mode 9.....	63
Figure 6-20: Mode 10.....	63

Figure 6-21: Mode 11	63
Figure 6-22: Mode 12	63
Figure 6-23: Mode 13	64
Figure 6-24: Mode 14	64
Figure 6-25: Mode 15	64
Figure 6-26: Mode 16	64
Figure 6-27: Mode 17	64
Figure 6-28: Mode 18	64
Figure 6-29: Mode 19	65
Figure 6-30: Mode 20	65
Figure 6-31: Scaled participation factors for the first 20 eigenmodes.....	65
Figure 6-32: Increase in eigen-periods due to added mass effect for the first 100 modes ...	66
Figure 6-33: Deviation of obtained eigenperiods from those in the previous reports.....	66
Figure 6-34: Eigenperiod comparison of different horizontal bridge girder curvatures	67
Figure 6-35: Effective damping ratio provided by structural damping, applied as Rayleigh damping.....	68
Figure 6-36: Total effective damping ratio	68
Figure 6-37: Effective damping ratio provided by aerodynamic dampers.....	69
Figure 6-38: Mode 56 with abnormal effective damping ratio.....	69
Figure 6-39: Comparing eigenfrequencies of B31 and B33 based structure	70
Figure 6-40: Deviation in eigenfrequency as function of eigenperiods B31 vs B33 element formulation	70
Figure 6-41: Comparing Eigenfrequencies of girder element size	71
Figure 6-42: Deviation of girder element size 20 vs 10 m.....	71
Figure 6-43: Turbulence components at height 228 m	72
Figure 6-44: Turbulence components at height 50 m	72
Figure 6-45: Turbulence components at height 16 m	72
Figure 6-46: Total along wind velocity at three heights.....	73
Figure 6-47: Targeted and simulated Spectra, W255C, seed 1	75
Figure 6-48: Targeted and simulated Spectra, W255C, seed 2	75
Figure 6-49: Targeted and simulated Spectra, W255C, seed 3	75
Figure 6-50: Targeted and simulated Spectra, W255C, seed 4	75
Figure 6-51: Targeted and simulated Spectra, W280C, seed 1	76
Figure 6-52: Targeted and simulated Spectra, W280C, seed 2	76
Figure 6-53: Targeted and simulated Spectra, W280C, seed 3	76
Figure 6-54: Targeted and simulated Spectra, W280C, seed 4	76
Figure 6-55: Targeted and simulated Spectra, W345C, seed 1	77
Figure 6-56: Targeted and simulated Spectra, W345C, seed 2	77
Figure 6-57: Targeted and simulated Spectra, W345C, seed 3	77
Figure 6-58: Targeted and simulated Spectra, W345C, seed 4	77
Figure 6-59: Targeted and simulated Spectra, W100C, seed 1	78
Figure 6-60: Targeted and simulated Spectra, W100C, seed 2	78
Figure 6-61: Targeted and simulated Spectra, W100C, seed 3	78
Figure 6-62: Targeted and simulated Spectra, W100C, seed 4	78
Figure 6-63: Targeted and simulated Spectra, W280L, 3D, seed 1	79
Figure 6-64: Targeted and simulated Spectra, W280L, 3D, seed 2	79
Figure 6-65: Targeted and simulated Spectra, W280L, 3D, seed 3	79
Figure 6-66: Targeted and simulated Spectra, W280L, 3D, seed 4	79

Figure 6-67: Illustration of the along wind turbulence, wind speed in m/s according to the colorbar	80
Figure 6-68: W255C seed 4 acceleration magnitude measurements at intermediate nodes	83
Figure 6-69: Energy dissipation of all set trough dynamic wind loading	84
Figure 6-70: Points assessed on bridge girder, plotted on mode shape 1	85
Figure 6-71: Displacement magnitude XY plane W280C seed 1	86
Figure 6-72: Displacement magnitude XY plane W280C seed 2	86
Figure 6-73: Displacement magnitude XY plane W280C seed 3	87
Figure 6-74: Displacement magnitude XY plane W280C seed 4	87
Figure 6-75: Displacement magnitude XY plane W255C seed 1	88
Figure 6-76: Displacement magnitude XY plane W255C seed 2	88
Figure 6-77: Displacement magnitude XY plane W255C seed 3	89
Figure 6-78: Displacement magnitude XY plane W255C seed 4	89
Figure 6-79: W255C seed 4 calculated stress at points.....	90
Figure 6-80: W280C seed 4 calculated stress at points.....	90
Figure 6-81: P1	91
Figure 6-82: P2	91
Figure 6-83: P3	91
Figure 6-84: P4	91

Abbreviations

NPRA	Norwegian Public Roads Administration
NTP	National Transportation Plan
VLFS	Very large floating structures
DOF	Degrees of Freedom
FEM	Finite Element Method
FE	Finite Element
FEA	Finite Element Analysis
ULS	Ultimate Limit State

1. Introduction

1.1 Ferry Free E39

With its 1100 km length and approximately 21 hours travel time, the European route E39 connect several communities and cities in the west of Norway. The Norwegian National Public Roads Administration (Statens vegvesen) is set to improve the coastal highway E39, and are guided by political plans. The national transportation plan (Nasjonal transportplan) issued in 2017 states the overlying goal for the public roads for the period 2018-2029, outlining the project ferry free E39. Aiming to reduce the overall travel time by about half to 11 hours, achieved by substitution of ferries by underwater tunnels and bridges, along with several road section improvements.



Figure 1-1: European Route E39 [5]

The costs for construction and upgrading according the mentioned plan amounts to about 340 billion NOK [1]. Most of the costs of the ferry free E39 will be funded by road toll, and have calculated to contribute to both greater efficiency measured in both time and cost for goods transportation. Fjords with great width and depth along with harsh weather conditions offers solutions that will break world records in the field of bridge engineering. Construction have already begun across one fjord, at the southernmost ferry crossing. Across Boknafjorden the world's deepest and longest underwater tunnel with estimated cost to 16,8 billion NOK, will open in 2025-26 as part of the project named "Rogfast". Shortening the travel time between the greater cities Stavanger and Bergen by around 40 minutes. Further north, the next ferry across Bjørnafjorden would offer a time saving of 30 minutes by substitution ferry to bridge [1]. The 4500 meters wide fjord offers depth to almost 600 meters, these critical site condition make it almost impossible to achieve fixed foundations. Thus the crossing have been projected as a floating bridge, having an estimated cost of 17 billion NOK. Including the necessary road and tunnel construction on both ends, the total cost of the Bjørnafjorden crossing is right beneath 30 billion NOK. Construction of the two above mentioned crossings will contribute to a time saving of 70 minutes traveling from Stavanger to Bergen, connecting the two main metropolitan areas of the west of Norway. Several other fjord crossings are also deemed feasible, but some are placed in areas with few inhabitants compared to the sections

between Stavanger and Bergen. The areal impact of the project is illustrated on figure 2. according to NPRA, the region affected by the ferry free E39 project amounts to a third of Norway's population and approximately 60% of the export value, named the "golden coast" [2].

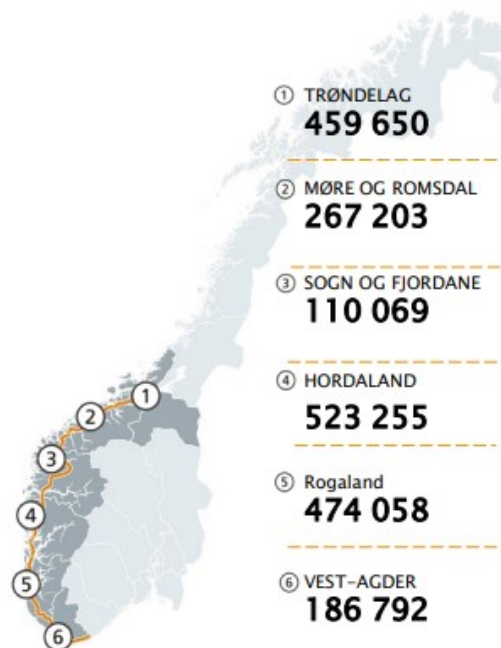


Figure 1-2: The «golden coast». Population along the planes Highway Route E39 [2]

The replacements of ferry crossings is widely discussed theme, several of the fjord crossings deem solutions that are quite groundbreaking and untraditional. Such that the question of relying the infrastructure on new and untried concepts is central. Some opinions on the Ferry Free E39 project point to that the cost could be reduced severely by replacing the existing ferries with new ferries relying on renewable energy. Such replacement would be more reliable than that of new bridge and crossings in term of new technology utilization, however the added value for the communities in form of great reduced travel time would not be achieve as swift. Ferry free E39 is in fact Norway's greatest infrastructure project, and most crossings are being studied to find the optimal design concept. For the area in both civil engineering and sustainability, the ongoing research initiated by the project would offer valuable knowledge available for several communities, also outside of Norway and north Europe. To accelerate the research efforts, the universities and the NPRA have also applied for funding for the European research projects [3].

Evidently, the Bjørnafjorden bridge is among one of the most expensive and demanding ferry replacements. Several design iterations have been done since the start of the project in 2009. Submerged floating bridge, cable stayed bridges with TLP foundation and floating bridge are among the designs which have been investigated. Several designs have since been rejected, and the floating bridge have been set as the current design. The main design consists of a ship fairway at the south end, provided by a cable stayed bridge giving free board length of 400 meters. Most of the bridge is supported by floating steel pontoons at 100 meters intervals. At the start of 2019 four different design configuration of the bridge exists. In 2019 NPRA appointed several consulting companies to further investigate the current designs in order to further pinpoint the final design.

1.2 Objective

The thesis work involves several key objectives. The main objective of this thesis is to construct a FEM of the bridge and apply dynamic loads due to a turbulent wind field. The thesis report is organized in the following Chapters:

- Chapter 1: Introduction to the project
- Chapter 2: Study of relevant fields
- Chapter 3: Applicable theory review
- Chapter 4: Review of design parameters
- Chapter 5: Modeling and work technique
- Chapter 6: Results
- Chapter 7: Conclusion and Discussion

1.3 Limitations

Modeling of the bridge and the dynamic load analysis both include several limitations. Limitations are sorted in two categories namely simplifications that have been done for the structural model itself and the simplification of applied loads.

Buffeting theory is only partly included in the formulation of aerodynamic damping and load appliance. Translation terms are simplified and the effect of rotation of the bridge girder is not included, which results in a model with low reliability in the results of torsional and vertical response.

Hydrodynamic is simplified, excitation forces are ignored essentially complying with a still water in which added mass, damping and restoring terms are effective. By which only the added mass effect is included as a constant mass inertia.

Simplifications of the models result in shortcoming accuracy of the dynamic analysis, in which only the horizontal response is assessed. Behavior of the horizontal response is largely determined by the first eigenmodes, which the model is able to accurately when compared to prior analysis carried out by consulting companies.

Modeling of a floating bridge subject to both aerodynamic and hydrodynamic effect is evidently a complex task, simplification should be in mind while assessing the dynamic analysis.

2 State of Art of the Floating Bridge Design

Very large floating structures have been constructed through the past decades, primarily utilized as floating airports, ports and storage facilities. Although floating bridges already exist, the behavior of such a large scale as the Bjørnafjorden bridge does not exist. Design and construction could thus be based on the experience from VLFS. The applicable experience is still quite limited, thus NPRA funds research among the three universities; University of Stavanger (UiS), Norwegian University of Science and Technology (NTNU) and Chalmers University of Technology. Over 50 researchers at the mentioned universities are connected to the ferry free E39 project.

2.1 Floating Structures

Large-scale floating structures are classified as either semi-submersibles or pontoons. Pontoons are characterized by their low depth-to-width ratios. Usually pontoons are deployed in a relative protected sea state condition such as in waters close to the coast or where breakwaters and other installations can be constructed to protect the structure from large waves and swells. Mooring lines comprising chains, ropes, sinkers, anchors or tethers may anchor the pontoon. Generally, pontoons are cost effective provided their low manufacturing cost and relatively easy to repair and maintain.

Semi-submersibles structural form is thicker than the pontoon and have since the 1970s been deployed in deep sea environment. Using column tubes or watertight ballast compartments, the Semi-submersibles are partly raised above sea level, compensating for the effects of larger wave-lengths and -heights. Floating oil drilling platforms are a typical example of this category of floating structures.

To achieve a floating bridge, both classes of floating structures have been assessed. Among other previous designs of the Bjørnafjorden bridge, one was based on a multi-span suspension bridge with towers supported on a tension-leg platform (TLP), effectively making a floating suspension bridge.

2.2 Existing Floating Bridges

Floating bridges have been prominent in the history of floating structures. Both connecting urban nodes across deep channels and permitting movement of soldier from shore to marine vessels during war. Historical floating bridges predominantly consisted of trafficable timber walkways built over an array of boats. With short lifespan and low rigidity these structures gradually declined, as the demand for more permanent solutions as cities grew.

One of the earliest floating bridges was constructed in 480 BC, during the invasion of Greece by the Persian king Xerxes. Comprising 300 small ships tied together, it was anchored by large ships at both ends in the Dardanelles Strait, Turkey.



Figure 2-1: Xerxes bridge [4]

Hobart Bridge

Being the first of its kind, the Hobart bridge in Australia completed in 1943. The unique design and construction consisted of 24 hollow concrete pontoons, connected to form a curved shape and anchored in the middle. At the western end, a large lifting section allowed ships to pass with a vertical clearance of 44 m. The featured arch shape provided the bridge necessary ability to withstand wind and wave forces. The bridge dismantled quite early in the mid-1950s due to increased traffic, soon replaced by the Tasman bridge [5].

Lacey V. Murrow Memorial Bridge

Being the first floating concrete bridge in the US, the Lacey V. Murrow Memorial bridge across Lake Washington, was completed in 1940. It has a combination of fixed and movable spans which would retract into fixed spans and give way to waterborne traffic. To date it is the second largest floating bridge in existence at 2020 m [6].



Figure 2-2: Homer M Hadley- (left), Lacey V. Murrow-Memorial Bridge (right) [7]

Evergreen Point Floating Bridge

The world's longest floating pontoon bridge at 4750 m was also constructed across Lake Washington, opened for traffic in 2016. Constructed of 77 floating concrete pontoons that are secured by 58 anchors to the lake bottom. Designed to withstand a 1,000 year earthquake and windstorms up to 89 mph. Along with the design life of 75 years the floating pontoon bridge is reminiscent of conventional bridges.

Bergsøysund Bridge

At the time of completion in 1992, the Bergsøysund bridge was the world's longest floating bridge measuring 931 m. The world's first end anchored bridge without side anchors, it is still one out of two existing pontoon bridges without anchors. The bridge is rested upon 7 hollow light weight concrete pontoons. To achieve sufficient stiffness, the deck is constructed above a steel frame [8].

Nordhordaland Bridge

The world's longest end anchored floating pontoon bridge was completed in 1994. It consists of 1243 m floating bridge along with a cable stayed part. Together measuring 1614 m, it is Norway's second longest bridge. The bridge is constructed with the same materials as the Bergsøysunds bridge, constructed by 10 floating hollow lightweight pontoons and a bridge girder constructed in steel [9].

2.3 Cable Stayed Bridge

With main spans covering a range from about 100 m to more than 1000 m, cable-stayed bridges are currently the most attractive bridge type in the world [10]. Cable stayed bridge design examine two main advantages over other bridge types:

- Cable stayed bridges could be built without auxiliary supports, since the flow of forces in construction phase is similar as the completed stage.
- Because the cable-stayed bridges do not undergo stress less deformations, they are stiffer than suspension bridges.

Thus the aerodynamic stability of cable-stayed bridges are distinctly greater than of suspension bridges, for a span of 500 m the critical wind speed for typical cable-stayed bridges reaches about 200 km/h, compared to suspension bridge which is stable up to 100 km/h .

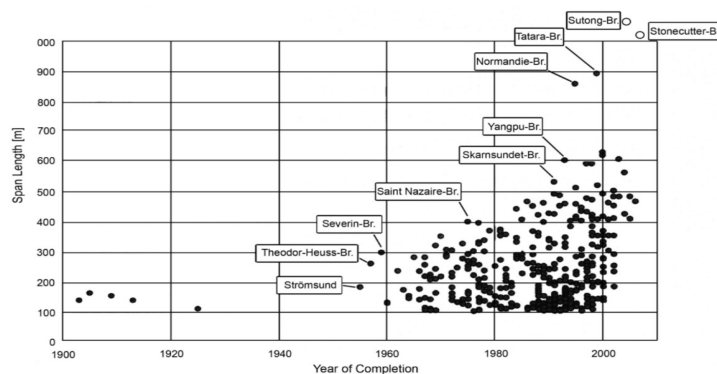


Figure 2-3: Development of Cable stayed bridges

Helgeland Bridge

Located at the west coast of Norway close to the polar circle, the Helgeland Bridge was completed in 1992. It has a main span of 425 m and a beam depth of only 1,2 m. Thus it has the record in slenderness with a ratio of 1:354 [10]. With the two towers measuring 138 m and 127 m in height and a total span of 1 065 m, the bridge provides a free height of 45 m under girder to allow for ship traffic. Due to the exposed location, the bridge withstood heavy storms during construction with wind speed up to 270 km/h.



Figure 2-4: Storm during construction of Helgeland bridge [10]

2.4 Bjørnafjorden Bridge Concepts

Evidently from the preceding chapters, no floating bridge with the necessary dimensions to cross Bjørnafjorden exist to date. However, the concept of floating bridges and especially cable stayed bridges is well known. Several design approaches still exist to date, all constructed by a floating bridge and a cable stayed part. Preceding's designs which have been studied but are no longer an option for the Bjørnafjorden Bridge includes the following.

Suspension bridge with two or more towers which are founded on floating tension leg platforms. Utilizing the well-known technology which is developed for offshore oil platforms. However, the design's initial cost assessment was deemed too high.

Floating Tube Bridge, in which the bridge is reminiscent of a tunnel floating in the water, floated by pontoons. A design which has not been utilized before either, and would not demand a high bridge to allow for ship traffic. However, the safety assessment utilizing such new technology over a great span, deems very high uncertainty along with even greater costs than a TLP suspension bridge [11].

Evidently, the cable stayed bridge connected to a floating bridge is the most reliable method taking preceding technology utilization in mind. The cost of construction have been deemed feasible and the design phases have been iterated several times. Four design approaches are still under further study, they all contain a cable stayed bridge in the south end and a floating bridge, and they are further described below:

- Alternative K11: Floating bridge which in the horizontal plane acts as an arch bridge, fixed at each end only.
- Alternative K12: Same concept as for K11, however an anchorage system is constructed which could be active during extreme conditions.
- Alternative K13: Floating bridge which is straight, equipped with anchorage which provide the horizontal stiffness.
- Alternative K14: Curved floating bridge and straight cable stayed bridge.

In this thesis, alternative K11 is assessed. During 2019 several consultants are working to further pinpoint the above designs.

3 Theory

3.1 Dynamic Analysis

Structural response is time dependent if the load exerted on the structure varies with time. A structure has a multitude of natural frequencies, and when the frequency of load is close to one of the natural frequency of the structure, a phenomena called resonance occurs. Close to resonance the response of the structure is severely greater than that of the static response subject to similar force magnitude. Thus a dynamic analysis based on the time variable load is fundamental to design the structure to ensure a long-lifetime and sufficient strength. Elementary parts of a vibrating system includes a mean for storing potential energy (spring or elasticity), a means for storing kinetic energy (mass or inertia), and a means by which energy is gradually lost (damper) [12]. The dynamic equation of motions is an extension of Newton's second law as following

$$f = ma \text{ becomes } r - k u - c \dot{u} = m \ddot{u} \quad \text{or} \quad m \ddot{u} + c \dot{u} + k u = r \quad \text{Eq. 3-1}$$

Where $r = (t)$, an externally applied load that varies with time. By convention, the displacement u is derivvated by time such that velocity is denoted \dot{u} and acceleration is denoted by \ddot{u} . In dynamic analysis the equilibrium of force is not only provided by the mass and acceleration product, $m\ddot{u}$, but also the two forces provided by damping and stiffness of the system. The damping coefficient c is in equation 3-1 defines the viscous related to the structural velocity. Stiffness k provides a force defined by the displacement, in structural analysis the stiffness is based on the geometry and material, calculated to an equivalent spring stiffness.

If all the basics components of the system behaves linearly, the resulting vibration is called linear vibration. In structural systems, specifically at high oscillations, the vibration is no longer linear and should be accounted for. The differential equations that govern the linear vibration are linear and nonlinear for nonlinear vibration. The principle of superposition holds for linear response, the mathematical technique for such problems are thus well defined. However for non-linear vibration the principle of superposition is not valid and the technique of analysis is more complex and less developed.

3.1.1 Eigenvalue

When a system is left to vibrate on its own, after the excitation by an initial displacement, velocity or force, the frequency with which it oscillates without external forces is called the natural frequency. The natural frequency w_n is of great importance to assess if the structure will experience resonance with external forces or movement. A structural system with n degrees of freedom will have n numbers of natural frequencies. For an undamped system, the natural frequency is defined by the relation of stiffness and mass.

$$w_n = \sqrt{k/m} \quad \text{Eq. 3-2}$$

3.1.2 Damping

Energy which dissipates in the form of heat and sound during each cycle is often of neglect size for a structural system. When energy is dissipated to heat and sound, it is defined by the mechanism known as damping. Consideration of the damping mechanism is however important for an accurate prediction of the vibration response. Especially close to resonance, when the damping is the controlling factor to the response magnitude. The vibrational energy in many practical systems is gradually converted to heat or sound, due to reduced energy in the system the response will gradually decrease accordingly.

3.1.2.1 Damping Mechanisms

Several damping mechanisms exist, the most commonly used being the viscous damping. By vibration in a fluid medium such as water or oil, the resistance offered by the fluid to the moving body causes energy to be dissipated. Viscous damping is based on several factors such as size, shape and the velocity of the vibrating body. For structural systems the damping is most often defined by Coulomb (dry-friction) damping or material (solid, hysteretic) damping. Damping force of the Coulomb damping is constant in magnitude but opposite in direction of the movement. The force is provided by friction between rubbing surfaces. When a material is deformed, energy is absorbed and dissipated by the material, termed material damping.

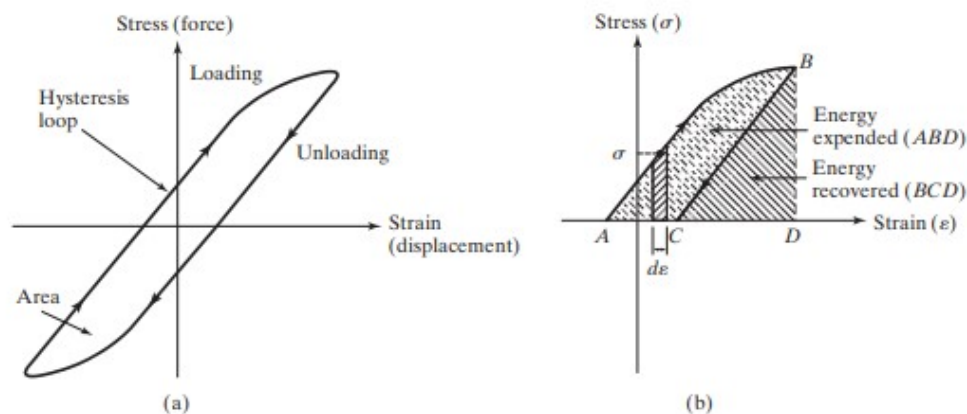


Figure 3-1: Material or Solid or Hysteretic Damping [12]

When a solid deforms, the internal planes slide and experience an internal friction. When a body experiences material damping is subject to vibration, the stress-strain diagram shows a hysteresis loop as illustrated on figure 5.

Damping is rarely viscous but to achieve a mathematical facilitated equation for analysis means, using an equivalent viscous damping is often utilized. By calculating the energy dissipated by the viscous damping, it is compared to the energy lost by other mechanisms [12].

3.1.2.2 Critical Damping and Damping Ratio

The critical damping c_c is defined as the value for the damping constant c in equation 3-1 for which the radical becomes zero [12].

$$\left(\frac{c_c}{2m}\right)^2 - \frac{k}{m} = 0 \quad \text{Eq. 3-3}$$

or

$$c_c = 2m\sqrt{\frac{k}{m}} = 2\sqrt{km} = 2mw_n \quad \text{Eq. 3-4}$$

The damping ratio ζ is then defined as the ratio of damping over the critical damping.

$$\zeta = c / c_c \quad \text{Eq. 3-5}$$

3.1.3 Dynamic Analysis with Several Degrees of Freedom

To be able to determine completely the positions of all parts of a system at any instance of time, a minimum number of degrees of freedom is required. Simple systems often requires only one DOF, but with increasing complex geometry the required DOF to be able to attain precise solutions increase. In mechanical systems, a DOF is often either one of the three translations or rotations, but could also be defined as a temperature. Besides geometric shape, the distinction between discrete and continuous systems must be established to correctly identify the required number of DOF. Practical systems can often be defined by a finite number of DOF. Systems of continuous elastic member have an infinite number DOF, which is found in most structural system. Systems with a finite number of DOF are called finite, those with infinite number DOF are called continuous. Solutions of a system with infinite number of DOF and thus an infinite number of equation is not possible, although it would give an exact result. Therefore most continuous systems are simplified as discrete, with a suiting number of DOF according to the computational power.

With the use of several degrees of freedom both complex geometric structures and discretization of continuous members. When applying the equation derived from Newton's second law for systems with many DOF, complicated algebraic manipulations is involved. To avoid such equations, the matrix representation for the system's mass, damping, stiffness and forces are utilized. Such that the equation introduced in 3.1 is applied for a multi-DOF system with the following equation.

$$[m] \ddot{\bar{x}} + [c] \dot{\bar{x}} + [k] \bar{x} = \bar{r} \quad \text{Eq. 3-6}$$

Where $[m]$, $[c]$ and $[k]$ are n by n matrices corresponding to the system with n DOF. The vectors \bar{x} , $\dot{\bar{x}}$, $\ddot{\bar{x}}$ and \bar{r} are all by dimension n by 1. Each coupled row in the equation is defined in a specified DOF.

3.1.4 Eigenmode

A system defined by several DOF consequently exhibits the equal number of eigenfrequencies. For system with several DOF, a corresponding eigenmode is found for each eigenfrequency. To understand the eigenmodes the eigenfrequency must first be examined. To find the natural frequencies of the system with multi-DOF, the system is set to vibrate without external force nor damping [13]. Such that equation 6 reduces to the following, expressed in matrix form.

$$[m]\ddot{\bar{x}} + [k]\bar{x} = 0 \quad \text{Eq. 3-7}$$

Further, the vector \bar{x} and its derivative could be expressed with the constant X , the function $T(t)$ which is a function of time and its derivative $\dot{T}(t)$ derived twice with respect to time.

$$\bar{x} = XT(t), \quad \dot{\bar{x}} = X\dot{T}(t) \quad \text{Eq. 3-8}$$

The vector X and its values which define the response distribution along the structural span at a given eigenperiod are thus independent of time t . Function T response time-variations for the model with frequency w_n^i and the time instance t . Therefore, the above equation definition mean that all coordinates have synchronous motions corresponding to an eigenfrequency. During motion, the shape does not change, only the amplitude. Configuration of the system, namely the vector X is known as the mode shape of the system. Assessing the eigenmodes of a structure gives a good understanding of how the systems response, and to which frequency they'll be triggered and prone to resonance [12].

3.1.5 Analysis Methods

There are mainly two different methods to analyze a system's dynamic response, the time-domain and frequency domain analysis method. Several factors decide which to utilize, the rudimental factor being if the time history of external load or response exists. If the time history of external load is available, an analysis based on incremental loading could be utilized based on a modeled system. Another factor is the available computational power, were the time domain demands much greater computational power.

3.1.5.1 Time-Domain Analysis

In the time-domain analysis, the external load is known and applied to the system. By incremental time analysis, the displacements at each node is examined through each step and the change of geometry is accounted for. Such that localized failures could be found during the given load history at a given time. Time-domain analysis is also applicable for nonlinear dynamic behavior. Several benefits exists for the time-domain analysis, but as complexity increases the computation required also increase rapidly.

3.1.5.2 Frequency-Domain Analysis

For the analysis of linear systems, the frequency domain could be utilized. The systems response is either calculated at each incremental frequency or is provided by a response spectrum, in both cases the output is often provided as either statistics or spectral density. Frequency domain could be considerable more efficient than time-domain, however: because of the constraints it is more limited in its applicability.

3.2 Finite Element Method

FEM is a method for numerical solution of field problems. Mathematically a field problem is described by differential equations or integral expression, often expressed in matrix form. With the readily available computational power today, FEM programs are mostly used to define FE models. FEM could be visualized as small finite pieces of a structure, called elements. Each element has a field quantity with simple spatial variation in form of polynomials. Element definition are often described by their degrees of freedom, which define the polynomials degree in the field quantity. With increasing DOF in an element, the field quantity polynomial degree increases. However, to retain an economical computational model, correct discretization of the model should be sought after. Finite element analysis and method could be summarized as the following [13]:

- FEA has several advantages over most other numerical analysis methods, including versatility and physical appeal.
- FEA is applicable to any field problem, for instance stress analysis, magnetic field and heat transfer.
- The body or region analyzed may have any shape, thus there is no geometric restriction.
- No restrictions on boundary conditions and loading.
- Material properties is not restricted to isotropy and may change between elements.
- A system could be defined by several different element types such as bar, beam, cable and plate.
- The mesh of a FE resembles the actual body or region to be analyzed.
- Greater use of field gradients and mesh grading could be used in regions were higher resolution is required.

3.2.1 System Definition

A FEM system consists mainly of several small pieces of structure, which are connected at points called nodes. Assembly of elements is called a FEM structure, a structure used in a sense to mean a defined body or region. The particular arrangement of elements is generally called a mesh. Thus numerically, a finite element mesh is represented by system of algebraic equations. The set of equation is defined in matrix form, were the unknown is the nodal quantities. In mechanical systems, the nodal quantities is often defined as translation, rotation or temperature. Generally the system definition is defined by the global stiffness matrix, the nodal displacement vector and the external loads vector [13].

$$[k]\{D\} = \{R\} \quad \text{Eq. 3-9}$$

3.2.2 Nonlinear Analysis

By the nature of geometric changes, material or nonlinearity due to changes of the boundary conditions, the application of non-linear analysis should be applied to the analysis. For the non-linear case, the assumption of a constant stiffness matrix is no longer valid. Global stiffness matrix is then based on the nodal displacements such that the new relation between stiffness and external load becomes [13]:

$$k_I(r) dr = dR \quad \text{Eq.3-10}$$

The stiffness defined by the incremental stiffness k_I . Equation 3-10 is in Abaqus solved by the use of Newton-Rapshon method as a numerical technique for solving the nonlinear equilibrium equations [14].

3.2.3 Newton-Rapshon Method

Newton-Rapshon method is an iterative method and could be applied to the structural nonlinear problem stated in Equation 3-10. Main motivation utilizing the Newton-Rapshon method is primarily the high convergence rates exhibited [15], compared to other alternative methods. Utilizing the method forms the following equation.

$$r_{n+1} = r_n - k_I^{-1}(r_n)(R_{int} - R) \quad \text{Eq. 3-11}$$

Equation 3-11 is further illustrated on Figure 3-2 for a 1 DOF system. Observed is the incremental stiffness updated for each iteration step. Although a great convergence rate is achieved, updating the incremental stiffness can be time consuming.

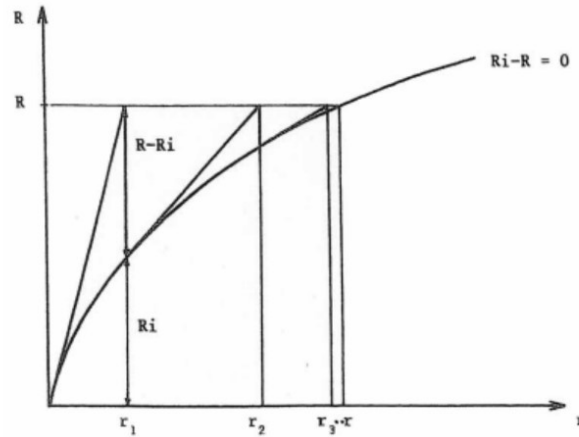


Figure 3-2: Newton-Raphson Method Illustrated [15]

A modified method of the method exists, obtained by the choice of how often the incremental stiffness is updated. The modified method leads to a computational efficient method where the convergence rate is somewhat slower [15]. In both cases the iteration is stopped when the sought accuracy is achieved. The accuracy criterion can be expressed by the equation:

$$|r_{n+1} - r_n| \ll \varepsilon \quad \text{Eq. 3-12}$$

The demanded maximum absolute difference ε , is the change of displacement from between two iterations.

3.2.4 Time Step

When implicit integration is used in Abaqus, the time stepping is based on the concept of half step residuals [14]. “The basic idea is that the time stepping operator defines the velocities and accelerations at the end of the step in terms of displacement at the end of the step and conditions at the beginning of the step. Equilibrium is then established at which ensures an equilibrium solution at the end of each time step and, thus, at the beginning and end of any individual time step. However, these equilibrium solutions do not guarantee equilibrium throughout the step. The time step control is based on measuring the equilibrium error (the force residuals) at some point during the time step, by using the integration operator, together with the solution obtained at, to interpolate within the time step. The evaluation is performed at the half step. If the maximum entry in this residual vector the maximum “half-step residual” is greater than a user-specified tolerance, the time step is considered to be too big and is reduced by an appropriate factor. If the maximum half-step residual is sufficiently below the user-specified tolerance, the time step can be increased by an appropriate factor for the next increment” [15].

3.2.5 Element Definitions

Essence of FEA is approximation by piecewise interpolation of a field quantity. As stated, usually polynomial interpolation is used. To define an accurate element type both will ensure correct results and swift computation of the system of equations. When utilizing FE models for dynamic analysis, accurate definition of mass and inertia in the mass matrix is of equal importance.

3.2.5.1 Euler-Bernoulli Beam Element

Utilized in the following analysis, is the element defined by the Euler-Bernoulli theory. Slender beams could sufficiently accurate be modeled by elementary beam theory, also known as Euler-Bernoulli beam theory. These elements does not allow for transverse shear deformation, plane sections remain plane and normal to the beam axis [14]. In a 3D representation, the simple illustration of translations and rotations are illustrated on figure below.

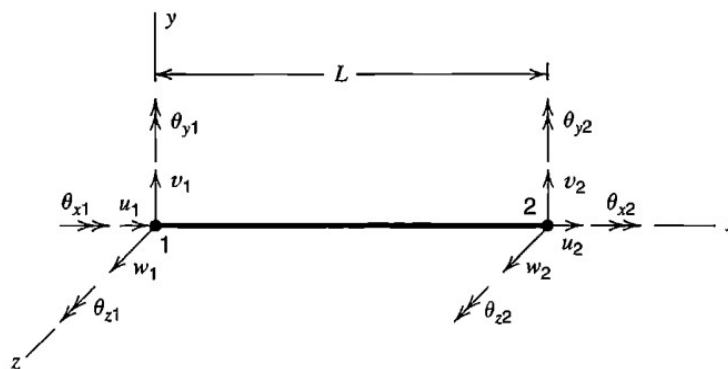


Figure 3-3: Deifinition of a Euler-Bernoulli beam along x-axis [13]

Interpolation of the Euler-Bernoulli beam elements use cubic interpolation functions, which makes them reasonably accurate for cases involving distributed loading along the beam [14]. Therefore they are well suited for dynamic vibration studies. In dynamic vibration studies, the d'Alembert (inertia) forces namely provide such distributed loading. The cubic beam elements are written for small-strain, large-rotation analysis.

3.2.5.2 Timoshenko Beam Element

Unlike the Euler-Bernoulli element definition, the Timoshenko beam allows for transverse shear deformation along with being applicable for both thick and slender beams. By utilizing linear interpolation, the computation is swifter compared to the Euler-Bernoulli element. Timoshenko beam elements may be subjected to large axial strains, and in combination with torsion the torsional strains are only accurate when axial strain is not large [14].

3.2.6 Mesh Convergence

By the discretization of both time and space quantities in the FEM, the convergence of system should be ensured. Mesh convergence is the computational mechanics that affect the accuracy of the results. It is related to how small the elements need to be to ensure that the results of the FEA are not affected by the size of the mesh, and thus a sufficient accurate model is established [13].

3.2.7 Dynamic Analysis in FEM

For a dynamic analysis, the equation is expanded to include the damping and mass matrix. The global stiffness matrix for the static case, is also applicable for the dynamic analysis. System definition of a FE structure in dynamic analysis thus resembles the equation 13 provided by expanding a dynamic analysis for several DOF systems.

$$[m]\{\ddot{D}\} + [c]\{\dot{D}\} + [k]\{D\} = \{R^{ext}\} \quad \text{Eq. 3-13}$$

The equation is a system of coupled, second-order ordinary differential equations in time. It constitutes a semi discretization, where nodal DOF are discrete functions in space but continuous of time. However, by applying numerical methods the solution by direct integration discretize in time.

3.2.7.1 Rayleigh Damping

Moving structure will experience damping. In finite element modelling, Rayleigh damping, is commonly adopted [12]:

$$[c] = \alpha[m] + \beta[k] \quad \text{Eq. 3-14}$$

The above formulation ensures that the eigen-vectors X are orthogonal with respect to the damping matrix $[c]$, similarly to their orthogonality with respect to the mass and stiffness matrices. Damping is thus represented by two terms. The first being provided by the mass-proportional damping coefficient α and the mass matrix $[m]$. Second term is the stiffness-proportional damping provided by the stiffness-proportional coefficient β and the stiffness matrix $[k]$.

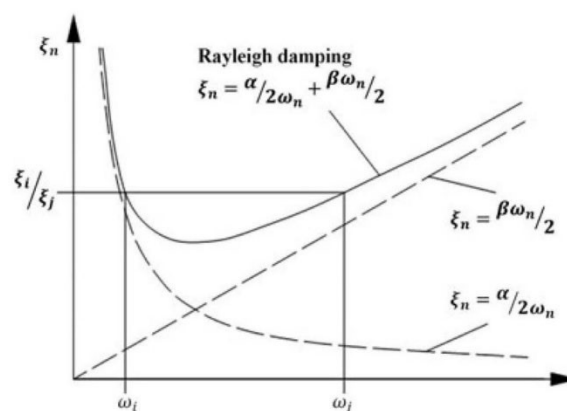


Figure 3-4: Rayleigh damping [12]

The two coefficients are determined based on the targeted values of damping ratios at two relevant eigenfrequencies. Evidently since several eigenvalues exist for a system with several degrees of freedom, the damping ratio also varies with different eigenmodes.

$$\zeta_n = \frac{1}{2\omega_n}\alpha + \frac{\omega_n}{2}\beta \quad \text{Eq. 3-15}$$

The values of α and β is selected based on the above stated function, which evidently varies with the varying critical damping factor ζ_n corresponding to n DOF. Because of the varying Rayleigh damping with the eigenvalue, and is therefore often selected according to engineering judgement.

Due to the nature of Rayleigh Damping, the damping ratio increases along frequency when beta is different from 0. Therefore judgement of which modes to study and their corresponding frequency in order to construct a representative damping coefficient applicable in the chosen frequency range. Large structures with several DOF response is defined mostly by the first modes, i.e. lowest frequency eigenvalues. Mass proportional damping is therefore chosen by the first mode of the system, found by iterative eigenanalysis. Stiffness proportional damping should be based on the frequency of loading, such that a frequency range between the two chosen eigenvalues should be controlling in the dynamic analysis carried out of the system.

3.2.7.2 Dynamic Implicit Method

Abaqus utilizes the d'Alembert's in the virtual work equation to solve FEA. The following theory is from the Abaqus theory manual [14]:

$$\int_V f \delta v dV = \int_V F \cdot \delta v dV - \int_V \rho \ddot{x} \cdot \delta v dV \quad \text{Eq. 3-16}$$

Where f is the body force at a point, F is the external body force and ρ is the material density. The d'Alembert term could be written as reference values, providing a more effectively part of the equation

$$\int_{V_0} \rho_0 \ddot{x} \cdot \delta v dV_0 \quad \text{Eq. 3-17}$$

The FE approximation of the integrals is in Abaqus utilized by the following

$$M^{NM} u''^M + I^N - P^N = 0 \quad \text{Eq. 3-18}$$

Where M^{NM} is the consistent mass matrix, I^N is the internal force vector and P^N is the internal force vector. The N implies an interpolation function basis where N^N is displacement independent.

A balance of d'Lambert forces weighted average of static forces is the following

$$M^{NM} u^{NM}|_{t+\Delta t} + (1 + \alpha)(I^N|_{t+\Delta t} - P^N|_{t+\Delta t}) - \alpha(I^N|_t - P^N|_t) + L^N|_{t+\Delta t} = 0 \quad \text{Eq. 3-19}$$

Where $L^N|_{t+\Delta t}$ is the sum of the Lagrange multiplier forces. The Newmark formula is used to obtain the displacements and velocity vectors.

$$x|_{t+\Delta t} = x|_t + \Delta t \dot{x}|_t + \Delta t^2 \left(\frac{1}{2} - \beta \right) \ddot{x}|_t + \beta \ddot{x}|_{t+\Delta t} \quad \text{Eq. 3-20}$$

$$\dot{x}|_{t+\Delta t} = \dot{x}|_t + \Delta t (1 - \gamma) \ddot{x}|_{t+\Delta t} \quad \text{Eq. 3-21}$$

Where α , β and γ is defined by:

$$\beta = \frac{1}{4}(1 - \alpha)^2, \quad \gamma = \frac{1}{2}\alpha \quad \text{and} \quad -\frac{1}{3} \leq \alpha \leq 0 \quad \text{Eq. 3-22}$$

Utilizing the method with automatic time increment might produce some noise if no damping is utilized. Numerical damping values around -0,05, ($\alpha = -0,05$) will efficiently remove noise and keep the response of interest unaffected. To keep the analysis to work efficiently, time steps should to some extent be automatically altered.

3.3 Hydrodynamic and Hydrostatic Effect

In regular wave conditions in which the linear wave theory applies, measuring waves in wave height and wave period, the hydrodynamic effect is dealt with as two cases [16]:

- Forces and moments on the body when the structures is restrained from oscillating, called wave excitation loads.
- Forces and moments on the body when the structures is forces to oscillate with the wave excitation frequency, identified as added mass, damping and restoring terms.

However, simplification for the pontoons hydrodynamic effects are done. Assuming pontoons not to be exposed to waves, only computing the behavior of the pontoon by the second case presented above. Further, by only concerning the first modes with low frequencies, the added mass term is assumed to be the main contribution of the hydrodynamic effect.

In marine engineering the movement of a vessel is commonly described as a six degrees of freedom system. Consisting of three translations DOF; surge, sway and heave along with the three rotations; roll, pitch and yaw. With the rotations being around the respective axes as illustrated in the figure below.

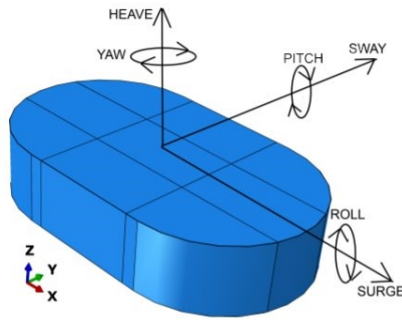


Figure 3-5: Marine orientations [16]

3.3.1 Added Mass

For the case of unsteady motion of bodies in a fluid or unsteady flow around objects, the force resulting from the fluid acting on the structure must be accounted for. This effect is added mass, and is a function of the area, velocity and density of fluid. For floating structures, the added mass is most often accounted for. However, for airplanes the effect is neglected given the low density of air. To account for the added mass, the mass or inertia matrix is expanded to include the effect of added mass. The effect is none when the system is static, and the effect changes with frequency. For simplicity, the added mass corresponding to large periods could be utilized in order to simplify the calculation, such that the added mass matrix is constant for all frequencies. Thus the inertia and mass matrix becomes the following utilizing the constant effect of added mass.

$$[m + m_{added\ mass}^{f \rightarrow \infty}]\{\ddot{D}\} \quad \text{Eq. 3-23}$$

3.3.2 Buoyancy Forces

For a floating structure, the position of the center of gravity and the center of buoyancy defines the Hydrostatic stability. To calculate the buoyancy forces are somewhat complex due to the changing buoyancy as the structure is displaces water. However a simplification could be done. For an equilibrium floating conditions, the hydrostatic calculations provided that the displacement and rotations are small could be provided in a simple form.

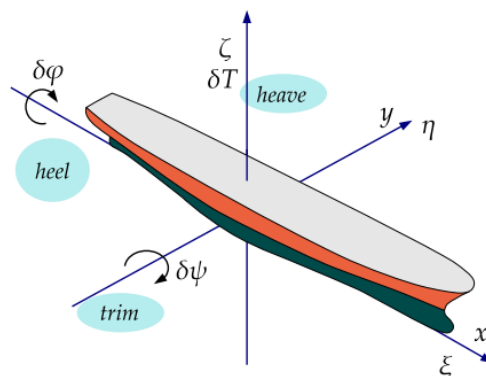


Figure 3-6: Ship changes in floating position [17]

The local directions of a floating specimens is provided by the figure above. Buoyancy forces acting on the specimen due to water displacement is provided by the following matrices:

$$\begin{bmatrix} \delta R_z \\ \delta R_x \\ \delta R_y \end{bmatrix} = \begin{bmatrix} \gamma A & \gamma M_{\omega\xi} & \gamma M_{\omega\eta} \\ \gamma M_{\omega\xi} & G\left(\frac{I_x}{\nabla} + z_v - z_G\right) & G\frac{I_{x\eta}}{\nabla} \\ \gamma M_{\omega\eta} & G\frac{I_{x\eta}}{\nabla} & G\left(\frac{I_y}{\nabla} + z_v - z_G\right) \end{bmatrix} \begin{bmatrix} \delta T \\ \delta\psi \\ \delta\varphi \end{bmatrix} \quad \text{Eq. 3-24}$$

In which Archimedes' Law holds, the buoyancy force B is provided:

$$G = \rho g \nabla = \gamma \nabla = B \quad \text{Eq. 3-25}$$

Small changes assume that the sinus and tangent of the rotations are approximated by the rotation itself, implying that cosines of the same rotation amounts to 1. Thus for symmetrical shaped floating specimen, in which the local coordinate system is identical to the global coordinate axes, the three forces above are further simplified:

$$\delta R_z = \gamma A \delta T \quad \text{Eq. 3-26}$$

$$\delta R_x = G \left(\frac{I_{xs}}{\nabla} + z_v - z_G \right) \delta\psi \quad \text{Eq. 3-27}$$

$$\delta R_y = G \left(\frac{I_{ys}}{\nabla} + z_v - z_G \right) \delta\varphi \quad \text{Eq. 3-28}$$

3.4 Aerodynamic Effects

Wind load on structures is determined by a combined effect of mean wind speed and the time-varying wind velocity field (turbulence). Both components generate forces on the structure depending on its aerodynamic characteristics. Different types of interaction between the natural wind and a structure can also take place, such as vortex induced vibrations or galloping, which both concern large responses across the main wind direction. Cable supported bridges may also vibrate when vertical and torsional movements are coupled, a phenomenon known as classical flutter occurs at high wind velocities. Bridge aerodynamics is an important element of long-span bridges to cope with wind loads. The total wind force can be stated in term of a time-averaged wind force and a gust force, and the statistical determination of the wind profile and the varying flow termed wind turbulence must be established in order to assess the load.

3.4.1 Wind Profile

Wind profile is calculated based on the statistically obtained reference wind speed, which in the Eurocode 1 [18] is e.g. provided as the 10 minutes mean wind speed with the annual probability of exceedance of 0,02, at 10 m height above the ground for the so-called terrain category II. Wind velocity and its corresponding direction near the ground surface changes with height. Wind velocity reduces close to the ground as the ground surface tend to reduce the wind velocity, however this effect reduces with height and is effective up to a height of 1000 meters which is known as the atmospheric boundary layer [19]. There are two effective

ways to describe the wind profile, namely the “Power Law” and the “Logarithmic Law”. According to Eurocode 1 [20] “Logarithmic Law” is utilized to describe the wind profile given the 1 hour wind velocity at 10 meters height with the required return period.

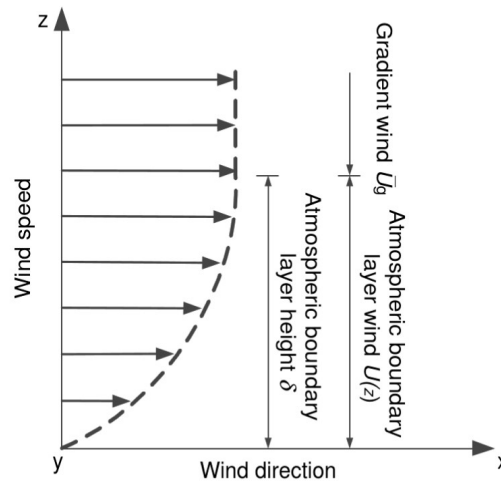


Figure 3-7: Wind profile in the atmospheric boundary level [19]

By assuming the wind direction is along the x-axis, y-axis the horizontal direction and z-axis it the vertical direction defined positive upwards, the wind velocity-vector at point $f = (x_f, y_f, z_f)$ at time t is expressed by the following [19]:

$$\bar{U}(x_f, y_f, z_f) + u(x_f, y_f, z_f, t) \quad \text{Eq. 3-29}$$

$$v(x_f, y_f, z_f, t) \quad \text{Eq. 3-30}$$

$$w(x_f, y_f, z_f, t) \quad \text{Eq. 3-31}$$

In which \bar{U} is the mean wind velocity in the wind direction and u , v and w are the turbulence component.

3.4.1.1 Logarithmic Law

In strong winds conditions, the most accurate mathematical expression for the mean wind profile is the “Logarithmic Law” [21]. Originally derived for the turbulent layer on a flat plate, have found to be valid in unmodified form for strong winds in the atmospheric boundary layer near the ground. In consideration of wind shear in strong winds above the ground, a non-dimensional wind shear can be considered:

$$\frac{d\bar{U}}{dz} \frac{z}{u_*} = \frac{1}{k} \quad \text{Eq. 3-32}$$

In which the friction velocity, u_* , is defined:

$$u_* = \sqrt{\tau_0 / \rho_a} \quad \text{Eq. 3-33}$$

Where τ_0 is the surface shear stress denoting the retarding force per unit area exerted by the ground surface on the flow, and ρ_a is the air density. The non-dimensional wind shear is a constant. The integration of Equation 3-33 then yields the usual form of the logarithmic law for the mean wind velocity $\bar{U}(z)$ at any height z .

$$\bar{U}(z) = \frac{1}{k} u_* \ln(z/z_0) \quad \text{Eq. 3-34}$$

Where z_0 is the surface roughness length, which is an effective height of surface roughness elements; k is von Karman's constant and has been found experimentally to have a value of about 0.4.

3.4.1.2 Power Law

Although the logarithmic law has a sound theoretical basis, it cannot be used to evaluate mean wind velocity at a height z below the zero-plane displacement z_h . The power law is an empirical formula for mean wind velocity profile, for mean wind velocity $\bar{U}(z)$ at any height z , the power law can be written as:

$$\bar{U}(z) = \bar{U}_1 \left(\frac{z}{z_1}\right)^\alpha \quad \text{Eq. 3-35}$$

Where \bar{U}_1 the wind velocity at any reference height z_1 , α is the power law exponent which depends on surface roughness.

3.4.2 Turbulence

To describe a turbulent flow, statistical methods must be applied. The three turbulence components are treated as stationary random processes mathematically and are described by the means of their standard deviations, time scales and integral length scales, power spectral density functions and that define the frequency distribution, and normalized cospectra that specify the wind spatial correlation.

Mathematically, the standard deviation σ is described as the square root of the variance. For the turbulence components in longitudinal direction u , the variance can be written as:

$$\sigma_u^2 = \frac{1}{T} \int_0^T u(t)^2 dt \quad \text{Eq. 3-36}$$

Where $u(t)$ are the turbulence component as a function of time t while T denotes the averaging time. Variance and standard deviation for the lateral direction and vertical direction is found utilizing the same mathematical relations.

Turbulence intensities is often used to describe the intensity of turbulence. It is defined as the ratio of the standard deviation of each turbulence component to the mean wind velocity of the same averaging time. Thus, the turbulence intensity $I_u(z)$ can be expressed as:

$$I_u(z) = \frac{\sigma_n(z)}{\bar{U}(z)} \quad \text{where } n = u, v, w \quad \text{Eq. 3-37}$$

Following the same relation, turbulence intensities for both lateral direction and vertical direction is also found. Normally, the turbulence intensities increase with surface roughness and decreases with height. It also varies with the duration (averaging time) used in determining the mean wind velocity and the standard deviation.

3.4.2.1 Buffeting

Several aero-elastic phenomenon are applicable for structures subject to static and turbulent wind fields. Buffeting falls in the category of wind—induces vibration due to wind turbulence that are created by the fluctuating and random forces. Buffeting in bridges may cause serviceability discomfort due to high and unpredicted movements. Buffeting and flutter can also have a coupling effect at high velocities, however for this thesis only buffeting is assessed.

Wind velocities are by nature fluctuating and thus random force is generated. The variable force frequency is variable, and may cause resonance when close to the bridge natural frequency. Response of the bridge is mainly defined by the turbulence intensity, natural frequency of the bridge and the shape of the structure. Flutter is a forms of an aerodynamic instability in long-span bridges. According to N400 [21] the critical wind speed for such an instability should be at least 1,6 times the design wind speed, used in the ULS case [19].

3.4.3 Wind Action Application

3.4.3.1 Wind Load

Wind load is divided into three force components for an element. Lift, drag and twisting moment where the lift and drag force are perpendicular and parallel to the wind direction, respectively. The moment acts around the axis of a line like element. Interaction of wind and structure is defined by aerodynamic coefficients, velocity pressure and reference dimensions. In a general simplified form the force contributions are known as the following [22].

$$q_D = q(z) H C_D \quad \text{Eq. 3-38}$$

$$q_L = q(z) B C_L \quad \text{Eq. 3-39}$$

$$q_M = q(z) B^2 C_M \quad \text{Eq. 3-40}$$

In which the velocity pressure is defined:

$$q(z) = \frac{1}{2} \rho v_r |v_r| \quad \text{Eq. 3-41}$$

Where q_D , q_L and q_M denotes the drag, lift and moment force per unit length respectively. C_D , C_L and C_M are the shape coefficients. H is the reference cross-wind dimension and B is the reference along wind dimension. Parameter q is the wind velocity pressure determined by the relative velocity $|v_r|$ of the structure and wind field and the density of air ρ .

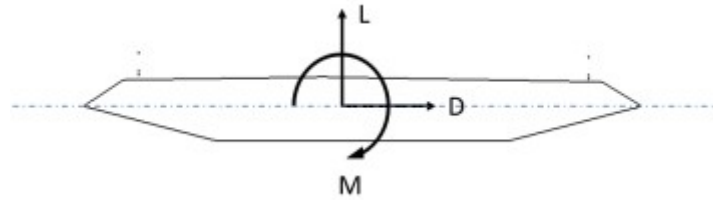


Figure 3-8: Sign convention

In a simplified static state, the above equation holds, However, as the aim is to reproduce turbulent wind fields, the turbulent flow is varying through space and in time.

Shape coefficient is defined for certain angle of attack, and could be implemented to vary through an analysis based on the orientation of the applicable element. Lift forces also have a tremendous increase given the rotation of the bridge girder. By assumption of a plane element and small variance of the angle of attack, both mentioned effects are assumed to provide negligible variance in the resultant force, and thus are not included in the thesis which considers a selected number of wind load components.

3.4.3.2 Wind Dampers

Damping effect, a velocity proportional force of the movement of the structure is extracted from the Davenport wind load formulation. In which the damping effect is proportional to the velocity of the structure and the wind velocity in the following form [22]:

$$C_{ae}^i = \rho C_i A_i U \quad \text{Eq. 3-42}$$

In which ρ is the density of air, C_i the drag factor for the applied direction of the damper, A_i the applicable area and U the mean wind velocity.

3.4.4 Statistical Concepts of Turbulent Wind Field

Turbulence is a random process, but could be modelled as a stochastic process. In order to obtain a sound method to model a turbulent wind field in both space and time axis, some parameters must be assessed. The following theory is gathered from [19].

The mean wind velocity is described by the expected value:

$$\mu_x = E[X(t)] \quad \text{Eq. 3-43}$$

The standard deviation σ is the root of the variance, which describes the functions spread in expected value.

$$\sigma^2 = E[(X(t) - \mu_x)^2] \quad \text{Eq. 3-44}$$

The standard deviation of the wind describes the turbulence intensity of the wind.

$$I_u(z) = \frac{\sigma_v}{U(z)} \quad \text{Eq. 3-45}$$

Further, in the perpendicular direction to the wind, the turbulence intensities are defined:

$$I_v(z) = \frac{3}{4}I_u(z) \quad \text{Eq. 3-46}$$

$$I_w(z) = \frac{1}{4}I_u(z) \quad \text{Eq. 3-47}$$

In order to quantify the inter-dependency of two random variables, e.g. turbulence components in two different points in space, and time, the covariance of two processes are expressed:

$$\kappa_{XY}(\tau) = E[(X(T) - \mu_X)(Y(T + \tau) - \mu_Y)] \quad \text{Eq. 3-48}$$

Where τ is the time-shift. When the random turbulence is observed, a statically measures within periods of time and space could be deemed stationary. Meaning even though the turbulence is random, it exhibit almost homogenous average characteristics. When describing the cross covariance of turbulence, the mean value is already subtracted. Thus, the cross covariance over the distance, r , and time, τ , for turbulent wind is expressed by the following:

$$R_m(r, \tau) = E[m(x, \tau)(m(x + r, t + \tau))] \quad \text{Eq. 3-49}$$

Cross covariance is utilized to compute the correlation of the processes, both the correlation and cross corelation is expressed respectively as:

$$\rho_X(\tau) = \frac{\kappa_Y(\tau)}{\sigma_X^2} \quad \text{Eq. 3-50}$$

$$\rho_{XY}(\tau) = \frac{\kappa_{XY}(\tau)}{\sigma_X\sigma_Y} \quad \text{Eq. 3-51}$$

To describe the turbulence trough time and space, time and length-integral scales are defined. Both sizes integrates the dimensionless correlation trough time or space. Time and length integrals are defined respectively:

$$T(z) = \int_0^{\infty} \rho_i(z, \tau) d\tau \quad \text{Eq. 3-52}$$

$$L_i = \int_0^{\infty} \rho_i(z, \tau) dr \quad \text{Eq. 3-53}$$

Length integrals are prescribed in Eurocode 1 [20], in which the length integrals are computed by the reference length scale equal to 100 m and reference height z_1 equal to 10 m.

$$L_u = \begin{cases} L_1 \left(\frac{z}{z_1} \right)^{0,3}, & \text{if } z > z_{min} \\ L_1 \left(\frac{z_{min}}{z_1} \right)^{0,3}, & \text{if } 0 \leq z \leq z_{min} \end{cases} \quad \text{Eq. 3-54}$$

Auto spectra S_N describes how the velocity variations (i.e. process variance) are distributed over different frequencies. The covariance in the frequency plane based on the integral of the covariance R_N .

$$S_N(z, n) = 4 \int_0^{\infty} R_N(\tau) \cos(2\pi n\tau) d\tau \quad \text{Eq. 3-55}$$

Several methods exists to describe the wind spectra based on the discussed parameters, according to Eurocode 1 [20] the Kaimalspectra is utilized in which the Auto spectra S_U is computed:

$$S_u^* = \frac{n S_u(z, n)}{\sigma_u^2(z)} = \frac{6,8 f_L}{(1 + 10,2 f_L)^{\frac{5}{3}}} \quad \text{Eq. 3-56}$$

$$f_L = \frac{n L_u^*(z)}{U(z)} \quad \text{Eq. 3-57}$$

Cross-spectra defines the correlation between two points measured in the frequency plane. The cross correlation of frequency plane is complex and could be expressed two ways:

$$S_{uu}(P_1, P_2, n) = 2 \int_{-\infty}^{\infty} R_u(r, \tau) e^{-i2\pi n\tau} d\tau \quad \text{Eq. 3-58}$$

$$S_{uu}(P_1, P_2, n) = |S_{uu}(P_1, P_2, n)| e^{i\phi_{uu}(n)} \quad \text{Eq. 3-59}$$

In which $\phi_{uu}(n)$ is the phase spectra.

Coherence is the magnitude of the cross-spectra normalized by the spectra of the involved single variables.

$$Coh(r, n) = \frac{|S_{uu}(P_1, P_2, n)|^2}{S_{uu}(P_1, n)S_{uu}(P_2, n)} \quad Eq. 3-60$$

Since cross-spectra is a complex functions, it is divided into a real and imaginary part called the co-spectrum and the quadrature spectrum respectively. The co-spectrum and the quadrature spectrum together make up the root square version of Eq. 3-60 defines the so-called root-coherence function:

$$\psi(r, n) = \frac{\sqrt{Coh^2(r, n) + Quad^2(r, n)}}{\sqrt{S_{uu}(P_1, n)S_{uu}(P_2, n)}} \quad Eq. 3-61$$

In the atmospheric boundary level the difference between the real and imaginary part is small, and could thus be assumed to be of same size as the Co-coherence spectrum. The root coherence function is then simplified:

$$\psi(r, n) = \frac{Coh(r, n)}{\sqrt{S_{uu}(P_1, n)S_{uu}(P_2, n)}} \quad Eq. 3-62$$

3.4.4.1 Coherence by Single Parameter

According to bridge design handbook N400 [23], Coh-coherence is approximated by an exponential decay function based on Davenport [24], referred to as the “Davenport coherence model”. In which the Coh-coherence is computed:

$$Coh(r, n) \approx \exp\left(\frac{-C_i^j f d_y}{\bar{u}}\right) \quad Eq. 3-63$$

Where \bar{u} is the horizontal mean wind velocity, $i = u, v, w$, $j = y, z$ and C_j^i is the decay coefficient. Usually the coh-coherence is computed with $j = y, z$, which would be applicable for 3 dimensional simulation, i.e. two spatial and time dimension. Due to the curvature of the bridge, one should rather calculate the wind co-coherence in three space dimensions and the fourth time dimension, and correspondingly simulate the turbulent flow field. As opposed to utilizing Taylor’s frozen turbulence hypothesis, a more accurate turbulent wind field is expected when applying the third space dimension.

3.4.4.2 Coherence by Two Parameters

Wind coherence have been studied details in the 1960s and 1970s, in which field measurements were conducted and the “Davenport coherence model” was established [25]. Few new field measurements have been conducted since, even though the accuracy of the coherence model presented above have been proved inadequate for several problems [25]. An issue with the coherence model utilizing a single decay coefficient is the overestimation of decay parameter [26]. Etienne, Jacobsen and Reuder thus define a coherence function with two parameter [25]:

$$Coh(r, n) \approx \exp\left(-\sqrt{\left(\frac{C_i^j f d_y}{\bar{u}}\right)^2 + \left(\frac{d_z}{l_2}\right)}\right) \quad Eq. 3-64$$

Which in addition to the parameters defined in the single parameter coherence model, $l_2 = \bar{u}/C_2^i$ is introduced, proportional to a typical length scale of turbulence, calculated by decay coefficient C_2^i and introducing the vertical velocity fluctuation based on $d_z = |z_2 - z_1|$.

3.4.5 Simulation of Turbulent Wind

Spectral representation of the assumed stochastic process could decompose to harmonic components with stochastic amplitude and phase [27]:

$$X(t) = \int_{k=1}^N A_k \cos(\omega_k t + \phi_k), \quad k = 1, 2, 3 \dots \quad Eq. 3-65$$

In which ϕ_k is independent stochastic variables between 0 and 2π , $\omega_k = \left(k - \frac{1}{2}\right) \Delta\omega$, in which $\Delta\omega$ is a measure of the resolution in frequency, t is the time and A_k is deterministic constants. In a practical aspect, the wind field could be assessed by several wind “waves” in which the amplitude is extracted from the statically measured or computed sizes as discussed in 3.4.4. Each harmonic representation is set into a random phase, which deems that even though a simulation based on the same statically measures, great differences could be observed due to the random effect introduced by the phase shift.

Above representation of stochastically processes could be defined to converge to a Gaussian process when the number of harmonically process increase significantly towards infinity. Thus the process could be deemed ergodic in mean value and auto-correlation function. This implies that the process could simulate a Gaussian process with given spectral density or autocorrelation function, provided that N is sufficient high. Autocorrelation function could thus be expressed by first defining A_k by the following functions:

$$A_k^2 = 2S_X^0(\omega_k) \Delta\omega \quad Eq. 3-66$$

$$R_X(\tau) = \sum_{k=1}^N \frac{1}{2} A_k^2 \cos(\omega_k \tau) \quad Eq. 3-67$$

Which are combined to achieve the following expression:

$$R_X(\tau) = \sum_{k=1}^N S_X^0(\omega_k) \Delta\omega \cos(\omega_k \tau) \quad \text{Eq. 3-68}$$

A limitation of the harmonic representation in the simulation is that the function will be periodic with the period $T_0 = \frac{2\pi}{\Delta\omega}$. Thus recommended the minima representation of harmonic components are $N \geq \omega_{up} \frac{T}{\pi}$. In which T is the length of simulation and $\omega_{up} = \max(\omega)$. Another limitation is the frequency resolution, in which $\Delta\omega$ should be sufficiently smaller than the minimal bandwidth of the smallest top. Utilization of a variable $\Delta\omega$ is also possible in order to achieve a more economical computation.

4 Bridge Design

4.1 General

Design of the end anchored bridge is split into several major structural elements.

- Bridge girder
- Cable-stayed tower and legs
- Cables
- pontoons
- Pontoon columns
- North support
- South support

Several of the structural elements are not relevant for the modelling of the bridge and will thus not be reviewed. This includes the north and south support, which is fundamental to ensure the end stiffness. They are simply modelled as fixed support in this thesis.

Through the whole span, the bridge girder follows the same curvature in the horizontal plane. Chosen during the latest design iteration, utilizing curvature of the bridge girder in the horizontal plane is more efficient to handle the horizontal loads by transferring to the boundaries by axial force rather than moment. Curvature was found to increase the buckling capacity, and that a curvature towards west would provide necessary strength towards the direction most exposed to horizontal wind and wave loads.

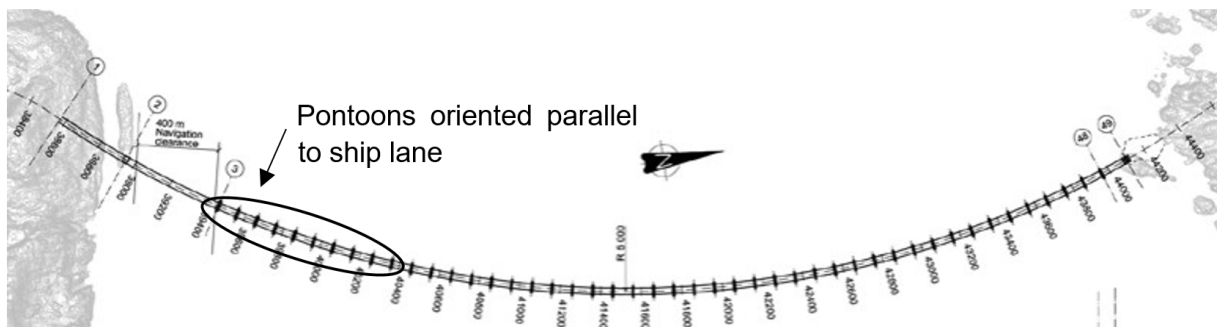


Figure 4-1: Plan view and the horizontal curvature [28]

Due to high demand to accessibility to the inner fjords by seaway, a navigation clearance measuring 400 m x 45 m is the reason to construct the cable stay. In order to minimize vertical curvature to 5%, a high bridge is constructed between the cable stayed bridge and the low bridge.

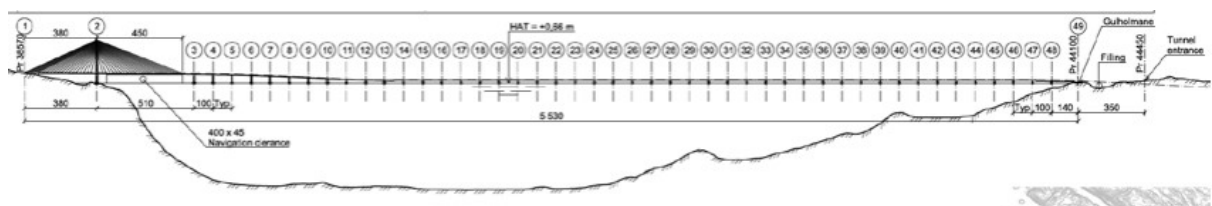


Figure 4-2: Vertical view of bridge elevation [28]

For convention, the road is divided in profiles according to a road system defined by NPRA. Numbering starts at south, with profile number 38500 and ends at 44500 at north. Each profile is placed in 100 meters intervals along the arch length, such that profile 38600 is 100 meters north of 38500. Along the profile numbering, the vertical curvatures are given in the table below.

Table 4-1: Vertical curvature

Profile nr.	Curvature
38500-38600	R = 5 000 m
38600-39400	-0,5 %
39400-40020	R = 14 500 m
40020-40250	-4,97 %
40250-40500	R = 5 000 m
40500-43800	0 %
43800-44300	R = 6 456 m
44300-44500	-5,0 %

Table 4-2: Bridge elevation

Profile nr.	Height
38500	56,065 m
39000	53,565 m
39500	50,570 m
40000	35,308 m
40500	16,165 m
44000	14,969 m

4.2 Bridge Girder

The structural part of the bridge girder consists of a steel box, constructed from panels which are stiffened by trapezoidal stiffeners and supported on transverse beams. There exist three different bridge girder cross sections designated to four spans; the low and cable stayed bridge, high bridge and at the end support.

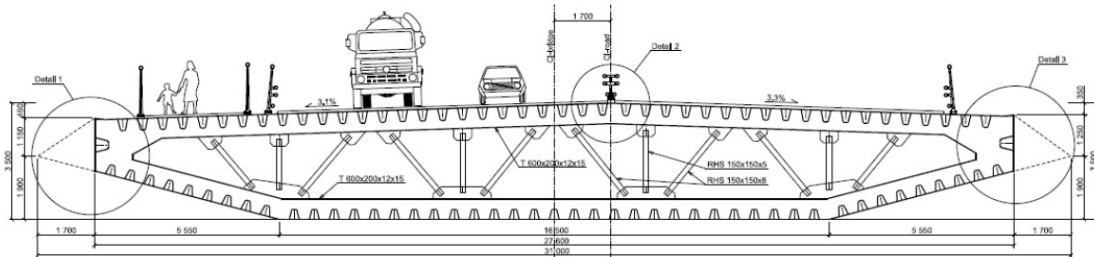


Figure 4-3: Bridge Girder Cross Section [28]

The thickness of the plates constructing the section is between 12 mm and 35 mm. Girder sections properties are further based on calculation according to (Summary Report), summarized in the tables below. Following the nomenclature of the reports, the weak and strong axis refer to the horizontal and vertical directions respectively.

Table 4-3: General bridge girder parameters

Vertical Center of Gravity	1,93 m
Width	31 m
Width excluding wind guides	27,6 m
Height	3,5 m
C_D	0,529
C_L	0,133
C_M	0,042

Table 4-4: Girder material properties

E-modulus	210 000 MPa
G-modulus	80 769 MPa

Table 4-5: Girder Cross Section properties

Parameter	Type 1	Type 2	Type 3
Location	Cable Stayed Low Bridge	High Bridge	North End South End
A	1,43 m ²	1,68 m ²	2,634 m ²
I _{strong}	115,62 m ⁴	132,47 m ⁴	181,1 m ⁴
I _{weak}	2,68 m ⁴	3,2 m ⁴	5,049 m ⁴
I _t	6,10 m ⁴	7,32 m ⁴	10,86 m ⁴
Mass ¹⁾	17 836 kg/m	19 798 kg/m	27 287 kg/m

1) Mass includes transverse beams, railings and asphalt.

Additional mass is present in the girder due to the reinforcements at several section as the table below provide.

Table 4-6: Additional steel weight due to reinforcements

Arch length	Weight
Axis 3	101 000 kg
Axis 4-6	71 000 kg
Axis 7-12	59 000 kg
Axis 13-47	55 000 kg
Axis 48	71 000 kg

4.3 Cable Stayed Tower

The 228 meters high tower is the only concrete structure besides both the north and south support. Constructed with hollow reinforced concrete elements, the tower has a varying cross section from the foundation to the top of the inclined tower. The diamond shaped tower allows for horizontal movement of the bridge girder with a clearance of 6 meters on each side. The bridge deck is only suspended by the stay cables, there exists no connection between bridge girder and the concrete tower cross beam. Prior reports found that normal reinforcing steel quantities would provide sufficient strength in the vertical legs and tower, and pre-tensioned concrete should be utilized in the cross beam.

Table 4-7: Cable stay tower material properties

E-modulus	36 000 MPa
G-modulus	15 117 MPa
Density	2550 kg/m ³

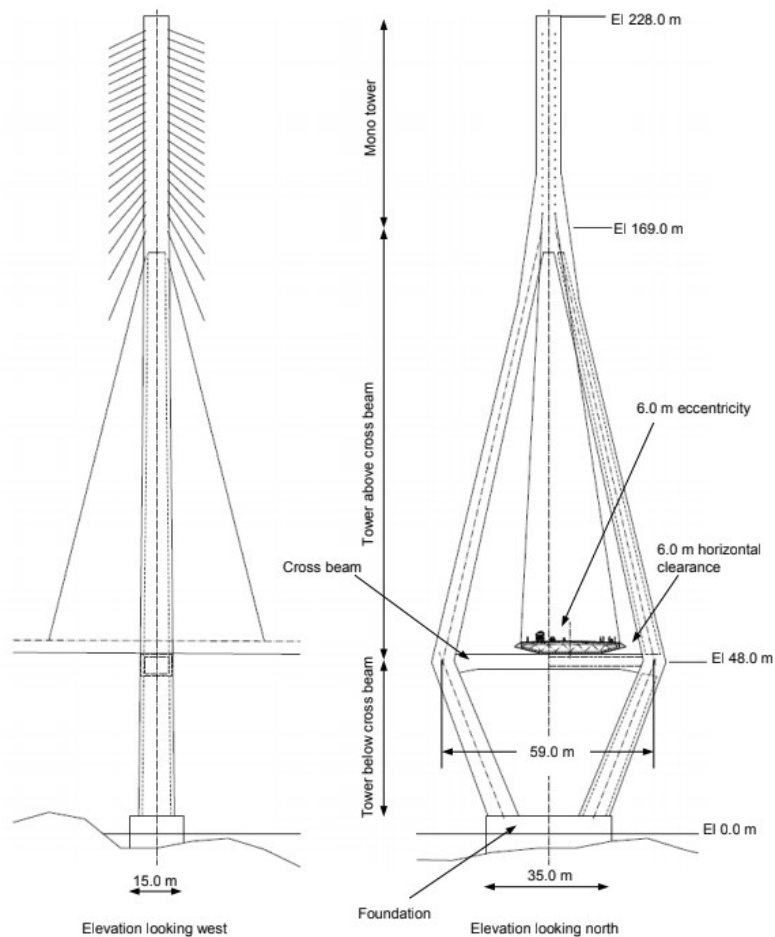


Figure 4-4: Cable Stay Tower [28]

Table 4-8: Geometric specification of tower concrete elements

Parameter	Tower legs	Cross beam
Thickness strong axis	1 200 mm	1 500 mm
Thickness weak axis	1 000 mm	700 mm
Height strong axis	12 000 mm (at foundation) Constant decline between foundation and mono tower 6 000 mm (at mono tower)	9 500 mm (constant)
Height weak axis	6 000 mm (at foundation, decline towards cross beam) 4 000 mm (constant between cross beam and mono tower) 6 000 mm (at mono tower)	4 000 mm (constant)
C _D	1.0	1.0

4.4 Cables

The 88 cables are evenly distributed, there are 22 cable pairs connecting the bridge girder and cable tower on the north side and 14 cable pairs connecting the bridge girder and cable tower on the south side. Remaining 8 cable pairs connects the tower to a cable anchor on the south side of the tower. Anchoring points to the bridge girder and cables is set every 20 meters, except for the span through the bridge girder, the span between anchoring points are set to 60 meters.

Table 4-9: Cable Properties

E-modulus	195 000 MPa
Density	7600 kg/m ³
Strand area	150 mm ²
C _D	0.8

Cables are designed so that they reach a tension level corresponding to 28% of the capacity for permanent loads, with a breaking strength of 1860 MPa, the cable stress is designed to be 520,8 MPa during service life. Calculation of the variable number of strands in each individual cable pair and thus the cable force provided by pre-stress is provided in 5.4.5.

4.5 pontoons

The 46 floaters have been designed as conventional plated steel ship type structures. Due to different loading of the floaters caused by the difference of height and girder sections, four different pontoon design exists. Each pontoon type has a designated position and is divided in the four sections; low bridge, ramp, high bridge and ship navigational channel.

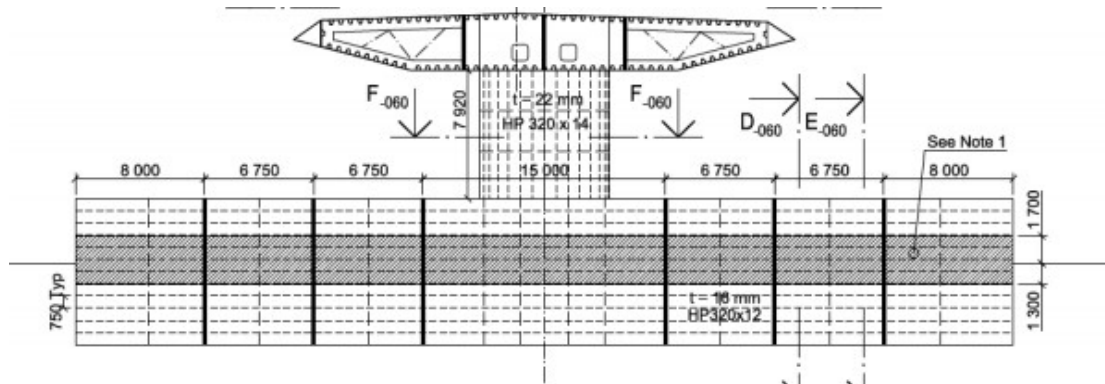


Figure 4-5: Pontoon type 4 side view [28]

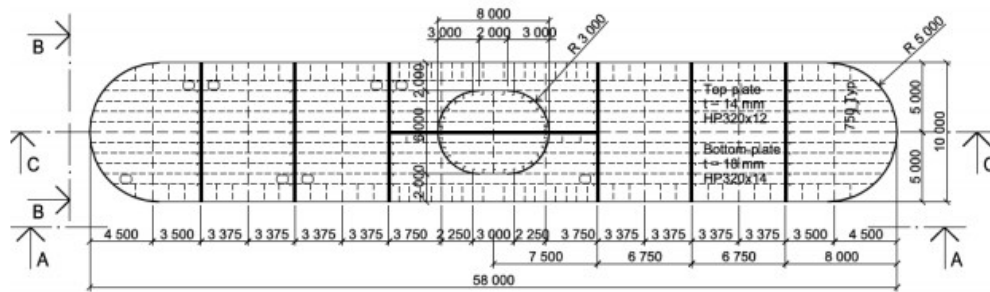


Figure 4-6: Pontoon type 4 top view [28]

Pontoons are designed with watertight compartments to prevent the pontoons from accidental flooding and utilize Super duplex stainless steel in the splash zone to provide sufficient corrosion resistance. In sizing of the pontoons, satisfactory dynamic behavior have been central to provide sufficient stiffness in roll, pitch and heave motion. A numbering system of the pontoons is introduced, starting at the first pontoon closest to the ship navigational channel with numbering from 1 to 46. pontoons are generally perpendicular to the bridge girder, the 10 first pontoons (Pontoon nr. 1:10) deviate by being oriented parallel to the ship lane, illustrated on figure 4-1.

Table 4-10: Pontoon properties

Parameter	Type 1	Type 2	Type 3	Type 4
Position	1	2:4	5:10	11:46
L	58 m	58 m	58 m	58 m
B	16 m	14 m	12 m	10 m
H	9 m	9 m	9 m	9 m
Mass	1180 tonnes	1040 tonnes	900 tonnes	750 tonnes
C _D	1,05	1,05	1,05	1,05

4.6 Pontoon Columns

At each pontoon, a column is placed at the centroid supporting the bridge girder. In order to minimize wind loading, a round shape have been chosen. Ship impact have been the governing design criterion for the columns. Thus due the changing moment arm as a result of variable bridge heights, column plate thickness and stiffeners are changing through the high to low bridge.

Table 4-11: Pontoon column properties

Parameter	Type 1	Type 2	Type 3	Type 4
Position	1	2:4	5:10	11:46
L	12 m	10 m	10 m	8 m
R	6 m	5 m	4 m	3 m
I _y	37,47 m ⁴	14,369 m ⁴	10,23 m ⁴	5,53 m ⁴
I _t	74,94 m ⁴	28,738 m ⁴	20,46 m ⁴	11,06 m ⁴
Mass	17 317 kg/m	9 429 kg/m	7 956 kg/m	7 200 kg/m

5 Modelling

5.1 General Definitions

Abaqus does not carry any units when computing. Such that consistent use of units is elementary. Units utilized in the following model is given:

Table 5-1: Units of measure utilized in model

Length	m
Mass	kg
Force	N
Pressure	Pa
Temperature	°C

Global coordinate system is defined with x-axis towards north, y-axis towards west and z-axis upwards. Origin is places at the south landing.

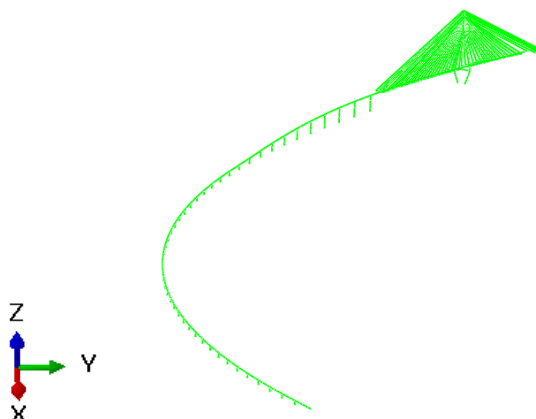


Figure 5-1: Bjørnafjorden FE Model in Abaqus CAE

5.2 MATLAB

MATLAB is a powerful matrix computation program, utilized with a user interface graphics, which makes navigation of variables an ease. MATLAB is utilized to model the whole bridge, due to the geometrical and sectional changes throughout the model along with the wind loading, makes MATLAB a natural choice. There is little to no communication between Abaqus and MATLAB, such that the input files constructing the model and analysis is provided in input files to Abaqus. The input files are clean text type files which follows Abaqus syntax.

Modelling in MATLAB initiated by defining the global geometry of the bridge, the curvature both in horizontal and vertical plane was defined for the bridge girder along with section identification to later assign section properties. The cable tower was constructed in its local coordinates and then places in the same global coordinate system as the floating bridge. Utilizing MATLAB to generate the geometry makes a swift process, and parameters are easy to change if wanted. Main advantage is utilization of the nodes to later model the wind and the ability to swift calculate section properties since they are varying for several elements.

After geometrical definition, all nodes are defined in matrices, along with elements which are based on the nodes. Identification of the section such as the pontoon type or the dimensions of the concrete tower is saved in separate columns for easy definition of the section properties when the input is written. A large amount of data are included in the script, which resulted in about 3000 lines of code.

5.3 Abaqus

Abaqus FEA is a software suite for FEA and computer aided design. Abaqus offers a user friendly CAD to design FEM, however to the intricate geometry, MATLAB was chosen instead. Utilizing the input file syntax is a clever way to achieve an accurate model. Input syntax even provide some options which are not available utilizing Abaqus CAE.

Abaqus models consist of nodes which are defined by three coordinates along with a designated node number. Elements are constructed by nodes, and all elements are have distinct element number. Element sets and node sets makes it easy to navigate in the model by finding for instance the bending moment throughout the bridge girder.

All parameters are in Abaqus essentially possible to change, however governing parameters secure a correct calculation of the problem, such that the model is ill-defined, warnings and error messages are provided which makes troubleshooting easy.

5.4 Bridge Components

This section describes how the elements constructing the bridge was modelled and what input syntax was utilized. The general parameters are as provided in 4.

5.4.1 Girder

The girders are defined as 20 m long elements, amounting to 270 elements constructing the 5400 m long bridge measured in arch length. Element utilized are both Euler-Bernoulli and Timoshenko beam elements in order to compare the difference of linear or cubic interpolation of the element. Interpolation is the calculation of deflected shape in between the nodes which construct the element, derivation of the deflection it utilized to compute strain which Abaqus computes the stress. If the element in linear the interpolation of the deflected shape is linear, thus the strain is not continuous through the element, a cubic interpolation is more suited for complex deflected shape while also providing a continuous strain. However by utilizing a fine mesh the difference between element formulations is not observed. Element length amounts to 0,72 of the element width, which is well below the recommended ratio provided in the Abaqus manual of 8. There exists 3 different girder cross sections which is taken to account in the model of the bridge. All elements section properties are defined as a general beam section. Due to the changing curvature of the bridge, individual definition of local coordinate system is utilized, in which the principal axis of the beam direction is defined when assigning the beam section properties.

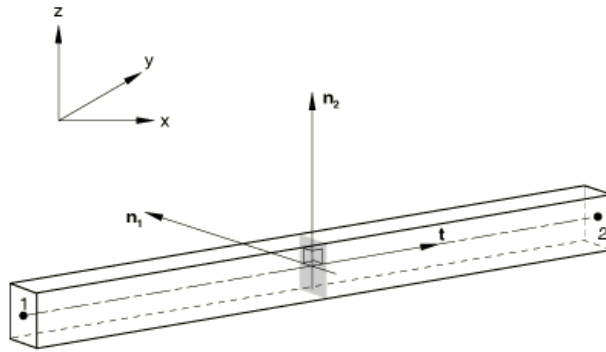


Figure 5-2: Abaqus definition of beam local axis [14]

Further, the input syntax utilized and properties defined is given below:

```
*ELEMENT,TYPE=<element type>,ELSET=<element set>
<element #>,<node1 #>,<node2 #>

*BEAM GENERAL SECTION,DENSITY=<ρ>, ELSET=<element set>
<area>,<I11>,<I12>,<I22>,<J>
<x1>,<y1>,<z1>
<young's modulus>
```

Element type is either defined by B31 for Timoshenko formulation or B33 for Euler-Bernoulli formulation. Material properties and section properties are well defined in 4.2. Coordinates (x_1 , y_1 , z_1) define the beam principal direction n_1 . To account for the Rayleigh damping, the section properties are also defined by a line with damping properties:

```
*DAMPING,ALPHA=<alpha>,BETA=<beta>
```

In which the alpha and beta parameters are found as discussed in section 3. Iterative analyse are done in Abaqus, then both parameters are defined in MATLAB based on the first and second circular frequencies.

To assess the section stresses, the arbitrary section is provided with section points in which stress is calculated. The points are defined at the girders both sides along with top and bottom by definition of the two coordinates in the local coordinate system in the following way:

```
*SECTION POINTS
<x1>,<y1>,<x2>,<y2...>
```

Each pair of coordinates (x,y) is defined along the axis (n_1,n_2) illustrated in figure 18.

To account for the effect of added weight on the girder cross section due to reinforcements, mass elements are defined accordingly at the applicable points in the following manner, in which the mass is applied in a single node:

```
*ELEMENT,TYPE=MASS,ELSET=<element set>
<element #>,<node #>
*MASS, ELSET=<element set>
<mass>
```

5.4.2 pontoons

Pontoons are defined by simple nodes, three nodes are utilized to properly simulate the pontoon properties. First the center of buoyancy, center of gravity and the point in which the column is connected are defined. The nodes are connected by arbitrary stiff beam elements regarded as connectors. At the COG, a mass point is defined with the corresponding mass for the type of pontoon in the following way:

```
*ELEMENT,TYPE=MASS,ELSET=<element set>
<element #>,<node #>
*MASS, ELSET=<element set>,ALHA=<alpha>
<mass>
```

Rayleigh damping is also provided, however the element does not contribute to any stiffness, only the alpha parameter is applied to the mass element.

Further, the pontoons buoyancy is based on 5 m draft, such that the center of buoyancy is centered 2,5 m below the surface. To account for the buoyancy forces provided by the pontoons, a concentrated load in the Z direction is applied at the center of buoyancy. Buoyancy forces are calculated based on the draft, which is 5 meters for all pontoons, and the individual water plane area. Concentrated loads are provided in the following way.

```
*CLOAD
<node #>,<direction>,<magnitude>
```

In which the magnitude is calculated based on the heave stiffness of the pontoon.

At the center of buoyancy three springs are applied to act as the water plane stiffness in heave, sway and roll directions. Simple calculation of the rotary stiffness are carried out assuming that the rotations are small.

Heave, roll and pitch stiffness are defined in the following way:

$$K_{heave}^i = A^i \rho g$$

$$K_{roll}^i = \frac{LB^3}{12} \rho g$$

$$K_{pitch}^i = \frac{BL^3}{12} \rho g$$

In which i denotes the pontoon identification, A , L and B denotes the geometrical measures area, length and width respectively, ρ is the water density and g is the gravitational constant.

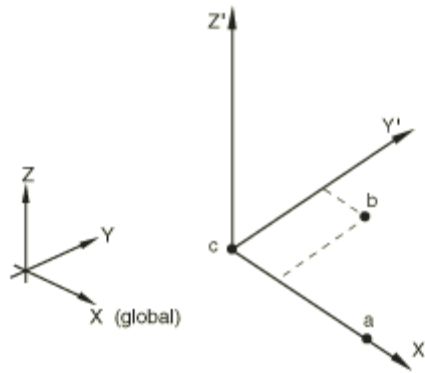


Figure 5-3: Local rectangular coordinate system [14]

Due to the different orientations of the pontoons, each pontoon has assigned their own coordinate system in which the springs are defined. Each pontoon is defined in the following way:

```
*ORIENTATION,NAME=<orientation name>
<xa>,<ya>,<za>,<xb>,<yb>,<zb>,<xc>,<yc>,<zc>
```

Where the coordinates refer to the three points (a,b,c) illustrated on the figure above. Coordinates are provided on the global coordinate system, such that Abaqus recognises the local coordinate system by point a which is positioned at the local x-axis, point b on the local x-y plane and point c at the local origin. Springs are defined according to the local coordinate system, and are individual calculated due to the changing geometry of the pontoons.

```
*ELEMENT,TYPE=SPRING1,ELSET=<element set>
<element #>,<node #>
*SPRING,ELSET=<element set>,ORIENTATION=<orientation name>
<direction>
<stiffness>
```

Where the direction is provided as one of the 6 directions, and the stiffness is provided as either N/m or N/degree.

To account for the effect of added mass, a mass element is defined at the center of buoyancy which are not put under gravity loading, but provide mass inertia. Due to the varying magnitude in each direction of the pontoon added mass, an anisotropic mass element is defined in which the effective mass in each direction is defined. Due to simplification, only the constant added mass is applied, however the added mass effect are frequency dependent.

```
*ELEMENT,TYPE=MASS,ELSET=<element set>
<element #>,<node #>
*MASS,ELSET=<element set>,TYPE=ANISOTROPIC,ORIENTATION=<orientation>
<Mx>,<My>,<Mz>,
```

In which each mass M_x, M_y and M_z is provided by prior analysis, from the reports [28]. Added mass is provided as a variable along the periods of oscillation, in the model the added mass corresponding to a very large period is utilized. Which will provide the correct added mass for the first global modes which are the most fundamental modes.

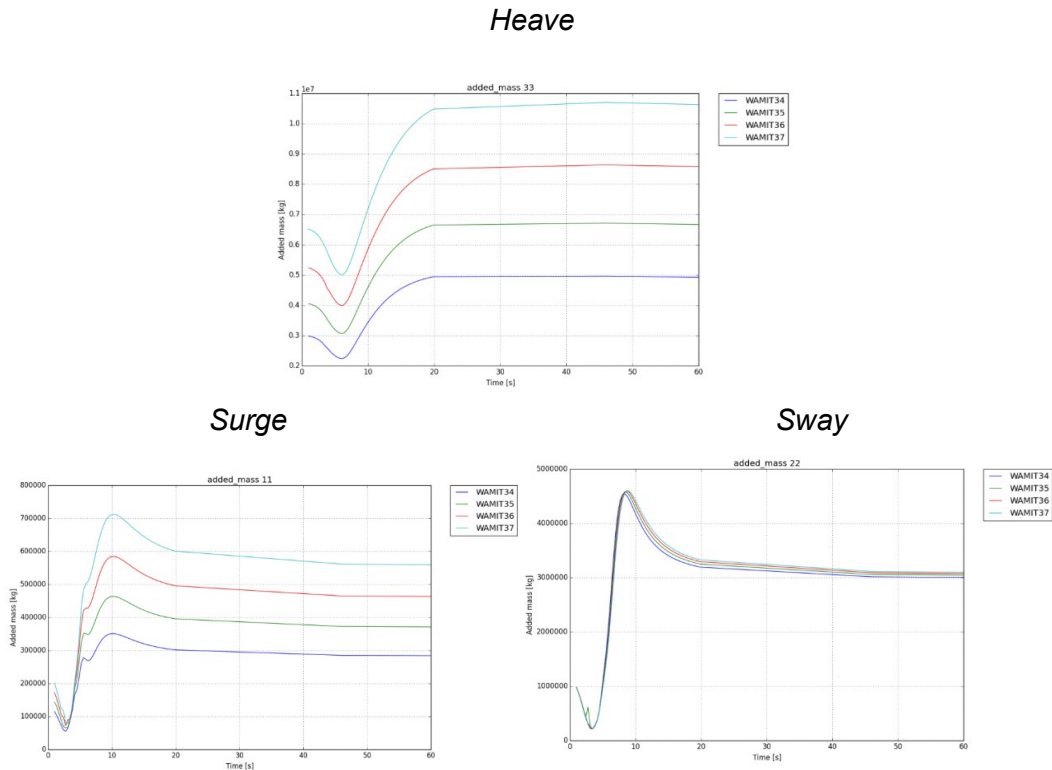


Figure 5-4: Variable added mass [28]

Added mass coefficients are based upon simulation results which is presented in the report, in the model present the added mass is simplified as constant. In above figure reference WAMIT34, WAMIT35, WAMIT36 and WAMIT37 correspond to pontoon type 1,2,3 and 4 respectively. Added mass has a huge mass inertia effect when compared to the pontoon mass, table below provides the added mass which is utilized in the FEM.

	Type 4	Type 3	Type 2	Type 1
Weight [kg]	750 000	900 000	1 040 000	1 180 000
Added mass fraction of weight:				
Heave	667 %	733 %	817 %	915 %
Surge	40 %	43 %	44 %	47 %
Sway	400 %	333 %	288 %	254 %

Table 5-2: Constant added mass utilized

5.4.3 Pontoon Columns

Pontoons columns are connected at the connection point at the pontoon and at the girder. An arbitrary point is defined at the girder in order to properly define where the column interact with the girder. In order to achieve the effect, rigid beams utilized as connectors are utilized, connecting the girder COG to the connection point. Each column is divided in approximately 5 m length, and defined as either Euler-Bernoulli or Timoshenko elements likewise the girder elements in order to assess the difference. Each pontoon is thus defined by two nodes and assigned section properties by general beam parameters:

```
*ELEMENT,TYPE=<element type>,ELSET=<element set>
<element #>,<node1 #>,<node2 #>

*BEAM GENERAL SECTION,DENSITY=<ρ>, ELSET=<element set>
<area>,<I11>,<I12>,<I22>,<J>
<x1>,<y1>,<z1>
<young's modulus>
```

The different column section properties are assigned according to section 4. Along with section properties, Rayleigh damping is applied:

```
*DAMPING,ALPHA=<alpha>,BETA=<beta>
```

5.4.4 Cable Tower

The concrete legs and cable tower has a rather complicated section definition, due to the gradually changing dimensions. Thus the elements are set to be short in order to have an acceptable linearization of the non-linear section property changes. Tower elements are defined by two nodes in the same procedure defined for the girder and pontoons; two node elements with general beam section properties and Rayleigh damping.

```
*ELEMENT,TYPE=<element type>,ELSET=<element set>
<element #>,<node1 #>,<node2 #>
*BEAM GENERAL SECTION,DENSITY=<ρ>, ELSET=<element set>
<area>,<I11>,<I12>,<I22>,<J>
<x1>,<y1>,<z1>
<young's modulus>
*DAMPING,ALPHA=<alpha>,BETA=<beta>
```

However, the cross section properties are calculated at both ends, and then a calculation of the effective combined property is done in the following way, based on Abaqus recommendation for tapered sections:

$$A^{eff} = \frac{A^I + \sqrt{A^I A^J} + A^J}{3} \quad \text{Eq. 5-1}$$

$$I_{11}^{eff} = \frac{I_{11}^I + \sqrt[4]{(I_{11}^I)^3 I_{11}^J} + \sqrt{I_{11}^I I_{11}^J} + \sqrt[4]{(I_{11}^J)^3 I_{11}^I} + I_{11}^J}{5} \quad \text{Eq. 5-2}$$

$$I_{22}^{eff} = \frac{I_{22}^I + \sqrt[4]{(I_{22}^I)^3 I_{22}^J} + \sqrt{I_{22}^I I_{22}^J} + \sqrt[4]{(I_{22}^J)^3 I_{22}^I} + I_{22}^J}{5} \quad \text{Eq. 5-3}$$

In which the subscripts I, J refer to each end node.

5.4.5 Cables

Cables are connected utilizing rigid beam elements as connectors, due to the line representation of the girder and cable tower, connector elements are connecting the center of the girder or tower and to the point which the cables are connected. Definition of the cables are done in the same way as the other two node elements utilizing B31 or B33 for comparing means.

```
*ELEMENT,TYPE=<element type>,ELSET=<element set>
<element #>,<node1 #>,<node2 #>
*BEAM GENERAL SECTION,DENSITY=<ρ>, ELSET=<element set>
<area>,<I11>,<I12>,<I22>,<J>
<x1>,<y1>,<z1>
<young's modulus>,<thermal coefficient>
*DAMPING,ALPHA=<alpha>,BETA=<beta>
```

However, the bending stiffness is in reality insignificant, a bending stiffness has to be provided in order to omit singularities when computing results based on the constructed stiffness matrix. Bending stiffness is set to be equal to the moment of inertia based on the cable diameter and is thus insignificant, also utilized for the temperature loading, the thermal coefficient is provided.

Pre-stressing of cables requires advanced calculation in order to attain pre-stress which will ensure that the bridge girder does not deflect or carry moment. Several determination processes exists [29]. However, a simple approach was adopted to calculate the cable pre stress; utilizing the simple static calculation of the equivalent load on each cable and the decomposition of cable force give the cable angle. Forces in the cables are thus based on both the load provided by the girder and the cable itself, such that the cable weight is included. Required vertical component are then transposes to the cable force direction and the total cable force is found. Based on the service stress of cables the cable area is calculated and rounded to

Another effect on prevalent on the cables are the change of the effective E-modulus due to the curvature of the cable [30]. A simple approach exists in which the sags of the cable is assumed to have a simplified form, in which a calculation method also utilized by preceding reports are utilized: the horizontal cable projection and tangent modulus in the following way.

$$E_{eff} = \frac{1}{1 + Ka^2} E \quad \text{Eq. 5-4}$$

$$K = \frac{\gamma^2}{12\sigma_1^3} E \quad \text{Eq. 5-5}$$

In which K is the tangent modulus, based on the horizontal cable projection a , the density of steel γ and the stress for permanent state σ_1 .

Further, the cables are put under temperature loading, in which the temperature is lowered by 10°C. The temperature coefficient is thus calculated so that all cables are induced a stress equal to the stress for permanent state. Input is provided in the following matter:

```
*TEMPERATURE
<element #>,<temperature>
```

The temperature is initially set at 10 °C and lowered to 0 °C during the first step.

5.4.6 Boundary Conditions

There are mainly five points in which boundary conditions are set. It is assumed that both fixed ends are bound in all directions, cable tower along with the cable anchorage south of the cable tower are also bound in all directions. Boundary conditions are defined by nodes, in which the magnitude of the boundary are set, in this case it is only set to 0.

```
*BOUDNARY
<node #>,<1 direction>,<2 direction>,<boundary>
```

Such that for all boundaries, the 1st and 2nd direction are set to 1 and 2 respectively, such that all directions in between are also bound. Boundary conditions utilized are equal to 0 prior utilized in the reports [28]. Summation of the applied boundaries are supplied in the table:

Table 5-3: Boundary conditions

Point	Boundary
Bridge girder end south	$(x,y,z,r_x, r_y, r_z) = 0$
Bridge girder end north	$(x,y,z,r_x, r_y, r_z) = 0$
Cable anchors	$(x,y,z,r_x, r_y, r_z) = 0$
Tower foundation	$(x,y,z,r_x, r_y, r_z) = 0$

5.4.7 Dampers

Both aerodynamic damping and hydrodynamic is effective on the bridge girder and pontoons. The damping force is proportional to the velocity of the structure, thus it is applied as dashpot connected to the nodes which construct the mentioned elements. The aerodynamic dampers

are applied to the bridge girder and the pontoon columns, assuming that the velocity of the cable tower is insignificant to have an effect. Dashpots are assigned with coefficients which refer to the global coordinates. Element "dashpot1" is utilized in Abaqus which applies a dashpot between a node and the fixed ground.

```
*ELEMENT,TYPE=DASHPOT1,ELSET=<element set>
<element #>,<node #>
*DASHPOT,ELSET=<element set>
<direction>
<coefficient>
```

Aerodynamic dampers have a calculated coefficient according to section 3.4.3.2. Hydrodynamic potential damping is neglected as it is of little effect in the first modes, the quadratic damping is based on the current speed, which in a conservative approach is assumed to be zero.

5.4.8 Connectors

To account for the geometric simplicity of the line like representation of the elements, several connectors are utilized in order to obtain the same geometrical effect provided by the eccentric connection points between several elements. Connector elements are defined as very stiff beams.

Defined as rigid beam, the table below represents the section properties of the connector elements:

Table 5-4: Connector element properties

Area	3 m ²
I ₁₁ , I ₂₂	100 m ⁴
J	100 m ⁴
E-modulus	210 GPa
Density	1 kg/m ³

5.5 Turbulent Wind Field

Construction of the wind field is done utilizing a MATLAB script provided by a postdoc at the University of Stavanger Etienne Cheynet. Mean wind velocities are first calculated at each element, based on the vertical position. It was chosen to utilize a high return period of 100 years in order to examine the extreme conditions. Following the Eurocode 1 [20], extrapolation of mean wind speed to greater heights than 10 m is done utilizing the logarithmic law. MATLAB script provided takes into account the 3 dimensional structure when calculating the wind field, such that the provided wind field is in fact 4 dimensional, three coordinates and time dimension. Most utilized method to take into account the third translation is to utilize Taylor Frozen turbulence hypothesis [31] in which the turbulent field is assumed not to evolve in the along-wind direction so that the turbulence at a point further downstream in the mean wind direction can be taken as a delayed version of the turbulence in a certain point upstream, without any change in the flow field between the two points. However, in reality the turbulent field does change while traveling through the third axis, which is taken into account in the turbulent wind field script provided by Etienne Cheynet.

Table 5-5: Wind velocity and return periods calculated according to Eurocode 1 [20]

Return period [years]	Wind Velocity [m/s], 1 h mean 10 m height
1	21,4
10	25,8
50	28,5
100	29,6
10 000	35,9

Along with the design wind speed given above, the location of the bridge in the fjord results in the use of reduction factors in specified sectors found based on wind simulations [32]. The direction of the wind is defined by the cardinal direction which is measured clockwise rotation from north, such that 90° is from east, 270° is from west. Bridge girder is rotated 10°, such that wind from 280° and 100° will hit the middle of the bridge span perpendicular.

Table 5-6: Directional reduction coefficients

Sector	Reduction Coefficient
0°-75°	0,7
75°-225°	0,85
225°-255°	0,9
255°-285°	1,0
285°-345°	1,0
345°-360°	0,7

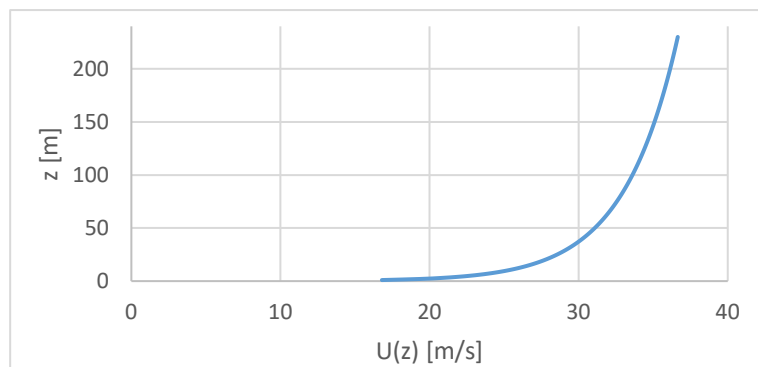


Figure 5-5: Calculated mean wind velocity

Following the handbook N400 [23] co-coherence is calculate based on a single decay coefficient. The coherence model captures the vertical coherence of the horizontal velocity components well, however by utilizing a two-parameter decay function the effect of reduced eddy based on the vertical coordinates is included [25]. Two parameter decay coefficients utilized in the coherence function is provided by Etienne Cheynet in table below:

Table 5-7: Decay coefficients utilized

j	C_j^i	C_1	C_2
u	C_u^x	1	
	C_u^y	7	0,01
	C_u^z	6	0,01
v	C_v^x	1	
	C_v^y	7	0,01
	C_v^z	10	
w	C_w^x	1	
	C_w^y	5	0,05
	C_w^z	3	0,1

In which the second decay coefficients C_2 provide a variation in the coherence model. Such that with increasing vertical components of two points, the coherence of the turbulence is reduced provided by the magnitude of C_2 . Further explanation of the term is assessed utilizing a simplified model, recall form section 3.4.4.2, the coherence $Coh(r,n) \approx \exp\left(-\sqrt{(C_1 d_y * k_1)^2 + (C_2 d_z * k_2)}\right)$, for simplicity it is asses here by the decay coefficients and the distance components. Such that as the magnitude of C_1 and d_y increase, the coherence decrease accordingly. Similar for the second term as C_2 and d_z increase, coherence decrease.

5.6 Wind Load Application

Wind loads are provided on all elements groups; bridge girder, cable tower, cables and pontoon columns. Each element load components are decomposed to the global coordinate system. Each element have a calculated drag, lift and/or moment force coefficient for each applicable direction. Loads are applied as line loads, such that only the force per length is provided, the total force is calculated by Abaqus. Two types of inputs are provided based on the wind loads. First a static wind load in order to achieve a swift convergence of the system, then a variable loading is defined. The sampling frequency is set to 4 Hz, such that an the variable value for each load is defined in tabular form for each increment conforming to 0,25 seconds.

```
*DLOAD,AMPLITUDE=<amplitude name>
<element>,<direction>,<magnitude>
```

Resultant force is the product of magnitude and amplitude at the time instance, such that the magnitude is defined as 1 Newton, such that the variable amplitude is in fact the total force magnitude. The direction is the provided as either the global directions. For each element an individual amplitude and load definition is provided.

Time instance	Load magnitude	Amplitude	Resultant force
0 s	1 N	0	0 N
0,25 s	1 N	120	120 N
0,50 s	1 N	90	90 N
0,75 s	1 N	130	130 N

Table 5-8: Example utilization of tabular amplitude speciation for load variation

5.6.1 Calculation of Wind Load

Calculation of the wind load have been tremendously simplified, by not accounting for the variation of lift coefficients due to angle of attack and assuming a constant drag coefficient when the yaw of wind field is changed. Resulting in a large reduction of lift component which increase significantly with the angle of attack. Element groups and applied forces are summarized:

- Bridge girder: Normal drag, lift and moment force.
- Pontoon columns: Normal and transverse drag force.
- Cable stay tower: Normal and transverse drag force.
- Cables: Normal and transverse drag force.

In which normal drag force is the force provided by the decomposed wind component with a perpendicular angle of attack of the main axis, the transverse drag force is the force provided by the decomposed wind components perpendicular to the normal. Lift force is calculated by the vertical turbulence component. Moment force is calculated based on the wind component with a perpendicular angle of attack of the rotation axis, which for the bridge girder coincide with the main axis.

The resultant wind component U, u, v and w are combined to three wind components for each element provided individual wind components and element orientation. In the following equations, (x, y, z) are the element local axis.

$$V_y = U_y + u_y + v_y \quad \text{Eq. 5-6}$$

$$V_x = U_x + u_x + v_x \quad \text{Eq. 5-7}$$

$$V_z = w \quad \text{Eq. 5-8}$$

Were the subscripts x, y and z denote which axis the component is decomposed to and V is the magnitude of all wind components.

The three decomposed wind components are then applied in the computation of wind load, in which the element drag, lift and/or moment coefficient are utilized. Coordinate system (x,y,z) are the element local axis.

$$F_y = \frac{1}{2} \rho C_D D_z V_y |V_y| \quad \text{Eq. 5-9}$$

$$F_x = \frac{1}{2} \rho C_D D_y V_x |V_x| \quad \text{Eq. 5-10}$$

$$F_z = \frac{1}{2} \rho C_L D_y V_y |V_y| \quad \text{Eq. 5-11}$$

$$M_x = \frac{1}{2} \rho C_M D_y^2 V_x |V_x| \quad \text{Eq. 5-12}$$

Were the forces and moments are calculated as line loads per unit length, in which D_y and D_z are the diameter of the perpendicular axis in which the force is calculated.

5.7 Step Definition

Several steps are utilized in order to obtain the applicable results in a swift manner. All steps utilize the option of toggling non-linear geometry changes on, such that the changes in geometry of the structures are included in the incremental calculation of stiffness and mass matrix. For the static steps the time length is not of importance, as a reduced time length would only imply a smaller increment size.

Step 1: Static

Length: 1 second

The first steps introduces the model to gravitational forces, such that the initial deflection of the bridge is established. The temperature of the cables are changes by a temperature increment such that an initial stress is introduced and cable pre-stress forces are efficiently applied. Several iterations of the static loading of the structures are done in order to check the model for faults.

Step 2: Frequency

Length: N/A

Second step utilize a frequency step which is not in the time line, the frequency step obtains the first 100 eigenfrequencies of the system based only on the mass and stiffness matrix.

Step 3: Complex frequency

Length: N/A

Sole purpose of the third step is to obtain the effective damping ratios for each eigenfrequency, which in Abaqus is only obtainable by executing complex frequency step.

Step 4: Static

Length: 1 second

To make the model ready for the dynamic wind loads, a static load of the first wind load step is applied in order to obtain a swift convergence of the system, a 1 second time length was found to be necessary length in order to achieve convergence.

Step 5: Dynamic

Length: 3600 second

Final step is when the dynamic load is applied, a maximum incremental size of 0,25 seconds are defined such that the loads at each increments amounting to 4 Hz is included in the computation. Due to the long firs periods of the system, a long step time is utilized in order to obtain enough cycles.

5.8 Summary of Analysis

Due to the curvature of the bridge, wind is applied in several directions. To obtain some statsical ground for the results, a minima of 4 analysis were done for each analysis.

Table 5-9: Summary of analysis sets

Set ID	Wind Cardinal direction	Number of analysis	Return Period	Wind Field Model	Wind Reduction Coefficient
W255C	255°	4	100 years	4D	1,0
W280C	280°	4	100 years	4D	1,0
W345C	345°	4	100 years	4D	1,0
W100C	100°	4	100 years	4D	0,85
W280L	280°	4	100 years	3D	1,0

In each set a number of 4 analysis are carried out, each with individual wind field simulations which deviate by changing the seed which is attributed by the use of random phases in the harmonic representation.

6 Results

6.1 Model Verification

In this section a verification of the model, studying the static deflection due to gravitational load and the impact of element formulation is done.

6.1.1 Initial Deflection

First the model is examined after the first static step, which applies the gravitational loads to the model.

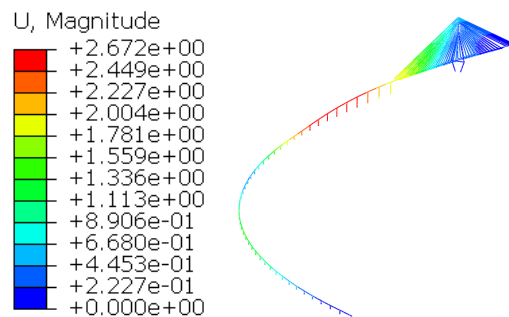


Figure 6-1: Initial deflection magnitude in meters all directions after applying gravitational load

A relatively large displacement in the x-y plane is observed in the section between high bridge and low bridge. The cause is most likely the pre-tensioned cables in the cable stayed bridge causes a moment due to the eccentricity. Eccentricity is caused by the curved shape of the girder in the x-y plane.

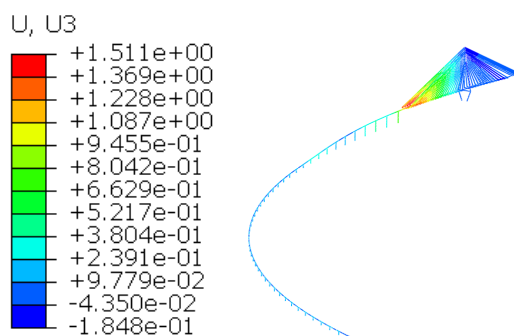


Figure 6-2: Initial deflection Z direction in meters after applying gravitational load

Upward curvature is seen in the cable stayed bridge causing an upward displacement of 1,4 m. An upward displacement on the floating bridge part is observed to be smaller; about 0,5 m. The pre-tensioned cables caused a small curvature since the girder is only carrying dead load. Pre-tensioning is designed to carry loads caused by i.e. traffic and would under traffic loading be straight. pontoons does not have the designed 5 m draft but 4,5 m due to only being put under dead load. If the girder were to be loaded with traffic load, the pontoons would have the designed 5 m draft and have smaller displacement in the z-direction.

6.1.2 Bridge Girder Forces

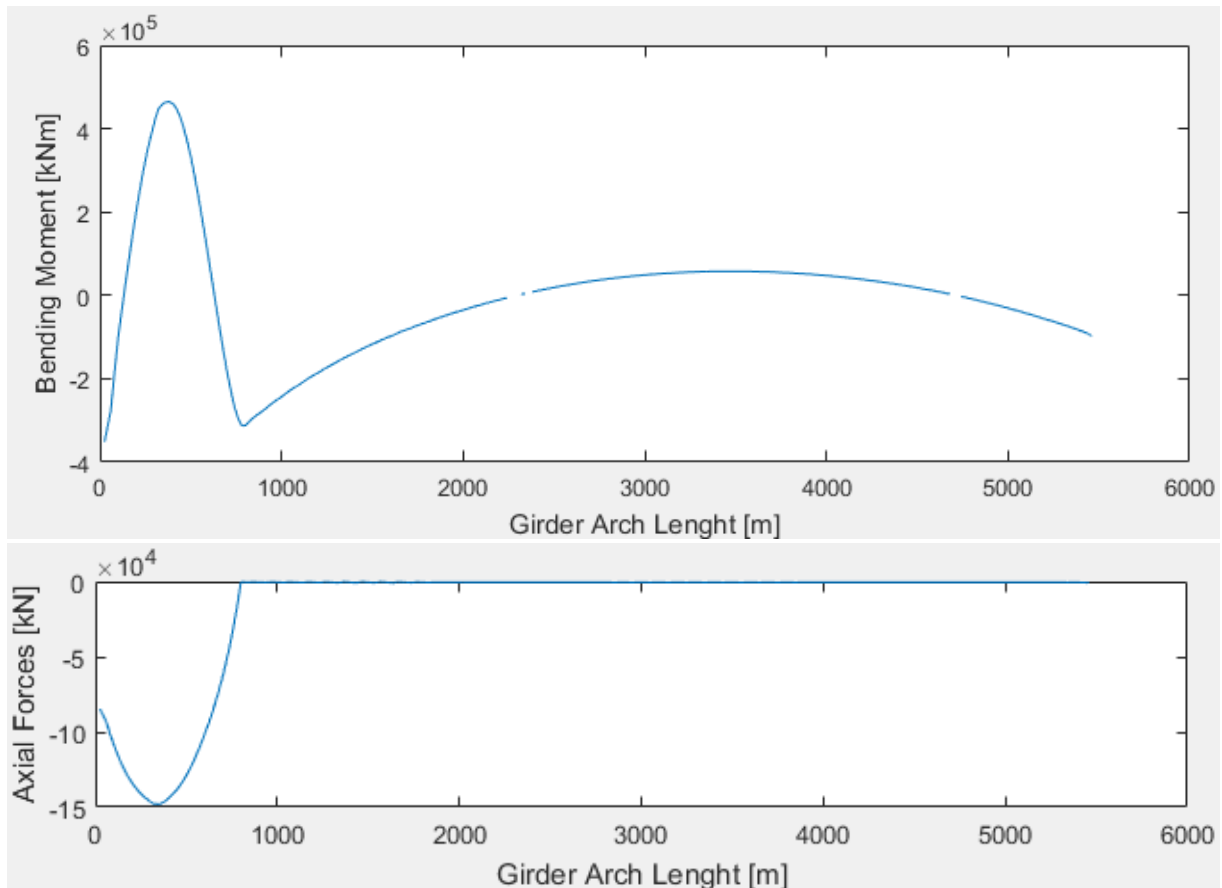


Figure 6-3: Moment about the strong axis and axial forces

In above figures, the X-axis is the bridge girder arch length. A high moment around the strong axis is noted at the middle of cable stayed bridge, which is produced by the pre-stressed cables and the eccentricity due to the bridge curvature in the horizontal plane. Compressive forces are also present in the bridge girder in the cable stayed part, due to the pre-stress of the cables.

6.1.3 Reaction Forces

Table 6-1: Reaction Forces

	Girder South	Girder North	Tower Leg 1	Tower Leg 2
F_z [kN]	6 270	15 200	245 000	237 000
M_1 [kNm]	66 000	63 000	376 000	255 000
M_2 [kNm]	103 000	330 000	199 000	305 000
M_3 [kNm]	404 000	95 000	4 337	3 600

6.1.4 Stress in Girder

Four points are defined in the bridge girder in order to obtain the stress in bridge girder, positions and naming of the points are given below:

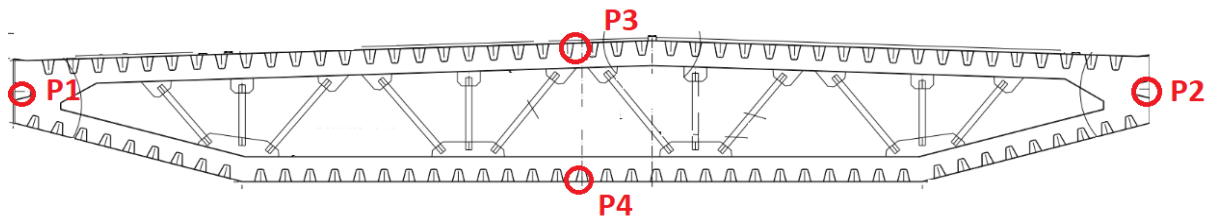


Figure 6-4: Stress points

Stresses at points after applying gravitational load

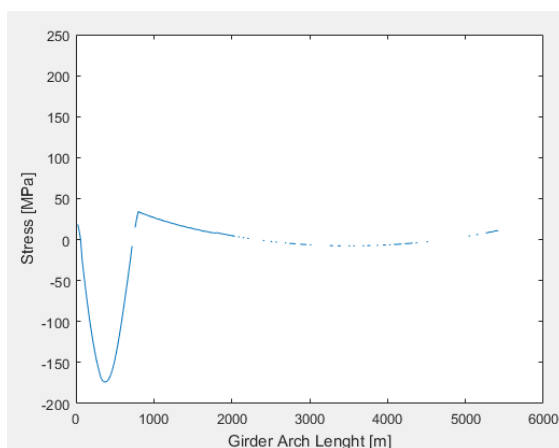


Figure 6-5: P1

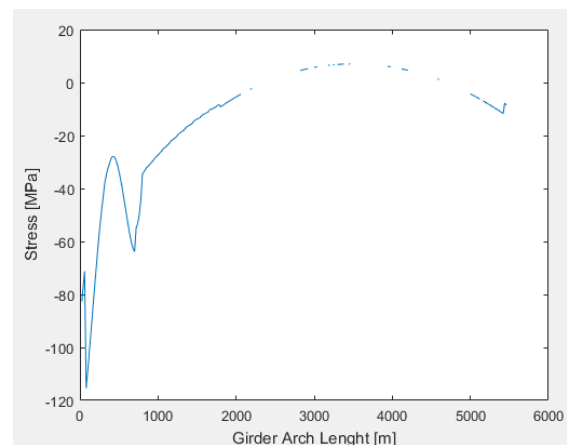


Figure 6-6: P2

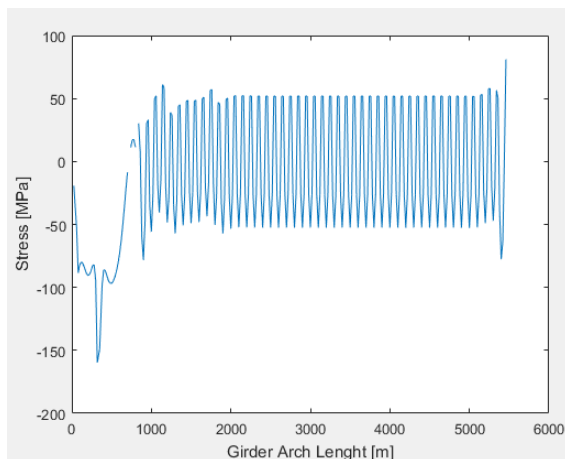


Figure 6-7: P3

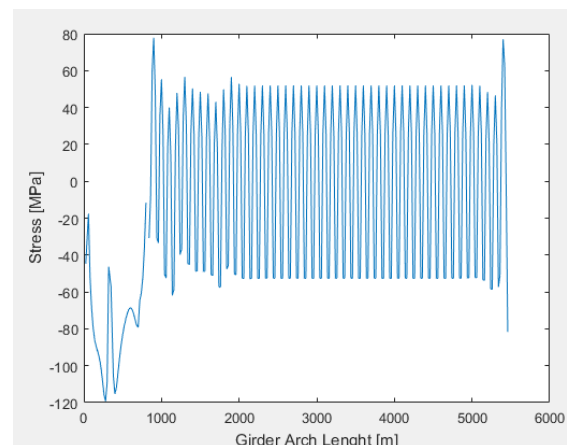


Figure 6-8: P4

Stresses at the points resembles the shape of the section moment and axial forces presented in 6.1.2. Stresses at P1 and P2 corresponds to the moment divided by the inertia and multiplying with the distance from the center to the point. Thus the stress is only a factor multiplied with the section force in the simple form. Stresses at P3 and P4 are provided by both axial force and the moment about the weak axis.

6.2 Eigen Frequency and Eigenmode Analysis

Eigen-value and eigen-vector Analysis is done in order to find eigenfrequencies, mode shapes and effective damping ratio.

6.2.1 Eigenfrequencies

Eigenfrequencies analysis is carried out for two model, with and without the effect of added mass. Results show that the first modes have very large eigenperiods and that added mass has a significant effect on the first eigenperiods. Effect of added mass and the comparison of eigenfrequencies with prior reports are carried out. Later the achieved damping is presented and the effect of aerodynamic and Rayleigh damping is examined.

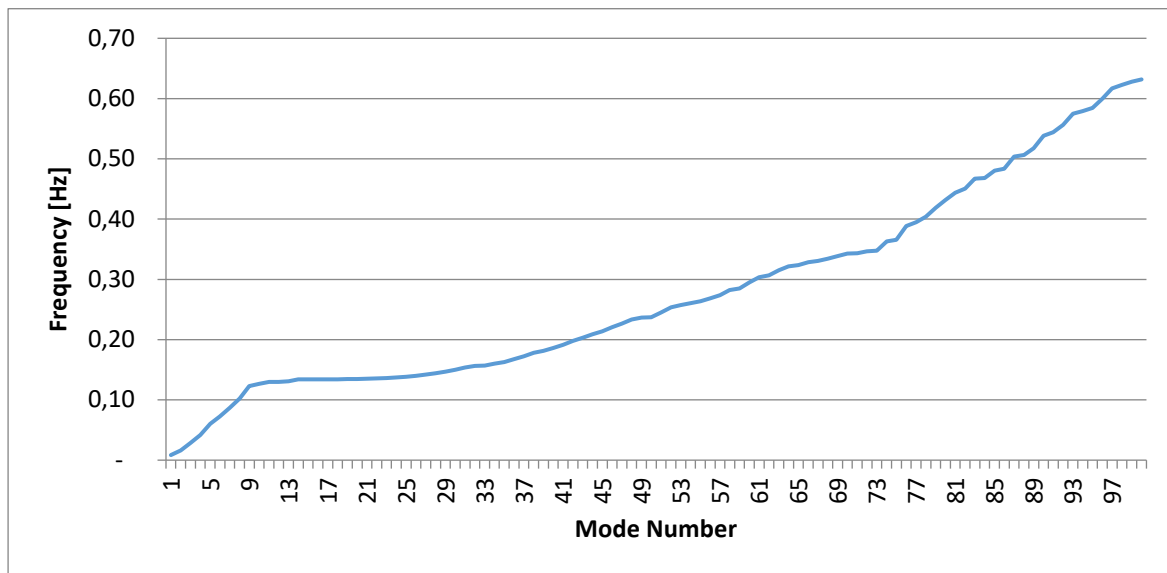


Figure 6-9: First 100 eigenfrequencies including added mass

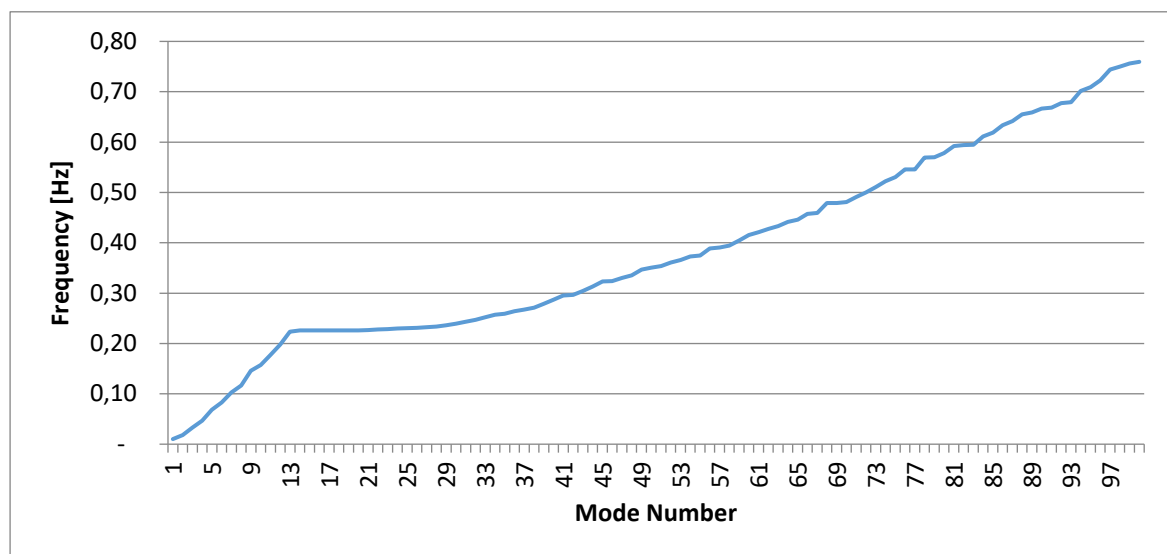


Figure 6-10: First 100 eigenfrequencies without added mass

6.2.1.1 Eigenfrequencies Including Added Mass

Below table presents a table of the first 50 eigenfrequencies for the model including the effect of added mass. First 100 eigenfrequencies are quite low, well below 1 Hz, emphasizing the large flexibility of the bridge, expected for such slender structure.

Table 6-2: First 50 eigenfrequencies with added mass

Mode	Frequency [Hz]	Period [s]	Mode	Frequency [Hz]	Period [s]
1	0,01	118,44	26	0,14	7,14
2	0,02	61,26	27	0,14	7,04
3	0,03	35,07	28	0,14	6,93
4	0,04	24,10	29	0,15	6,80
5	0,06	16,64	30	0,15	6,66
6	0,07	13,84	31	0,15	6,51
7	0,09	11,54	32	0,16	6,40
8	0,10	9,81	33	0,16	6,36
9	0,12	8,13	34	0,16	6,24
10	0,13	7,91	35	0,16	6,14
11	0,13	7,71	36	0,17	5,97
12	0,13	7,71	37	0,17	5,80
13	0,13	7,63	38	0,18	5,62
14	0,13	7,46	39	0,18	5,51
15	0,13	7,46	40	0,19	5,38
16	0,13	7,46	41	0,19	5,22
17	0,13	7,45	42	0,20	5,05
18	0,13	7,45	43	0,20	4,92
19	0,13	7,44	44	0,21	4,79
20	0,13	7,43	45	0,21	4,68
21	0,13	7,41	46	0,22	4,54
22	0,14	7,39	47	0,23	4,41
23	0,14	7,34	48	0,23	4,28
24	0,14	7,29	49	0,24	4,23
25	0,14	7,22	50	0,24	4,21

6.2.1.2 Eigenfrequencies excluding the effect of added mass

Table 6-3: First 50 eigenfrequencies excluding added mass

Mode	Frequency [Hz]	Period [s]	Mode	Frequency [Hz]	Period [s]
1	0,01	98,71	26	0,23	4,32
2	0,02	54,50	27	0,23	4,31
3	0,03	30,74	28	0,23	4,28
4	0,05	21,35	29	0,24	4,23
5	0,07	14,65	30	0,24	4,18
6	0,08	12,06	31	0,24	4,11
7	0,10	9,77	32	0,25	4,05
8	0,12	8,56	33	0,25	3,97
9	0,15	6,84	34	0,26	3,89
10	0,16	6,36	35	0,26	3,86
11	0,18	5,64	36	0,26	3,79
12	0,20	5,05	37	0,27	3,74
13	0,22	4,47	38	0,27	3,69
14	0,23	4,43	39	0,28	3,59
15	0,23	4,43	40	0,29	3,48
16	0,23	4,43	41	0,30	3,39
17	0,23	4,43	42	0,30	3,37
18	0,23	4,43	43	0,30	3,29
19	0,23	4,43	44	0,31	3,19
20	0,23	4,42	45	0,32	3,09
21	0,23	4,41	46	0,32	3,09
22	0,23	4,39	47	0,33	3,03
23	0,23	4,37	48	0,34	2,98
24	0,23	4,36	49	0,35	2,88
25	0,23	4,33	50	0,35	2,85

6.2.2 Mode Shapes

To assess the mode and its main response components could be widely discussed, as almost all directions are involved in each mode, however by visual inspection, some directional shapes are seen to be significant. Mode shape and its main participation of movement governs the response of the structure, and a general idea of which load will trigger what kind of movement is gathered. In order to further investigate the participation of several DOF, a scaled participation plot gathered from Abaqus is obtained. In which the participation factor for each direction is given as a percentage of the maximum participation factor of the first 20 modes, explained further below.

In general the first modes with the longest periods are governed by horizontal movement, followed by increasing participation of torsional, observed to be contributing from mode 3. The first vertical mode is observed at mode 9. Vertical participation of the cable stayed bridge girder is observed in mode 10. No local tower modes involving concrete cable stay tower is observed in the examined eigenmodes, or any up till mode 50. Cable tower motion is assumed to be neglected in the dynamic response analysis of the bridge due to the high stiffness compared to the slender bridge girder.

Horizontal modes

Dominating the first modes is the horizontal movement of the girder, in modes 1 through 9 great participation of horizontal modes are observed. In the first modes corresponding to eigenperiods in between 118 till 17 seconds. Long periods require a long analysis time period in order to allow for sufficient cycles, such that the effect of resonance could be examined. Due to the nature of environmental loads, the first modes are most likely triggered by wind load, which mainly provides the horizontal loads.

Vertical modes

Vertical participation of the bridge girder is not observed until mode, corresponding to a period with 8 seconds duration. Participation of vertical movement is since present in all modes, but stands most out in mode 16, in which almost the whole bridge is in phase with the vertical movement. Mode 14 and 15 corresponding to eigenperiods of 7,5 seconds provides mainly vertical participation, in which the first mentioned is almost symmetric, and the second is asymmetric.

Torsional modes

Torsion participation is observed from mode 3 corresponding to eigenperiods of 35 seconds. Participation of the torsional movement increases as the eigenperiods length decrease, and is coupled with all other movement in all of the first 20 modes except modes 14 and 15.

Displacement
XYZ Coordinates
High
Low

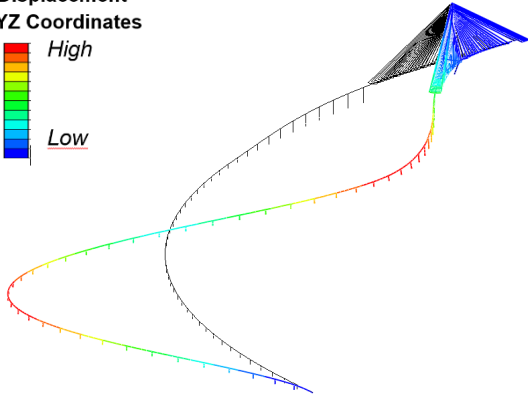


Figure 6-11: Mode 1

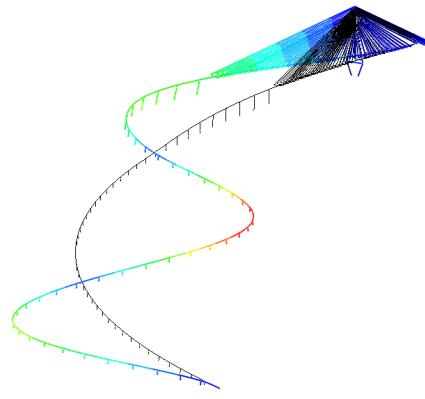


Figure 6-12: Mode 2

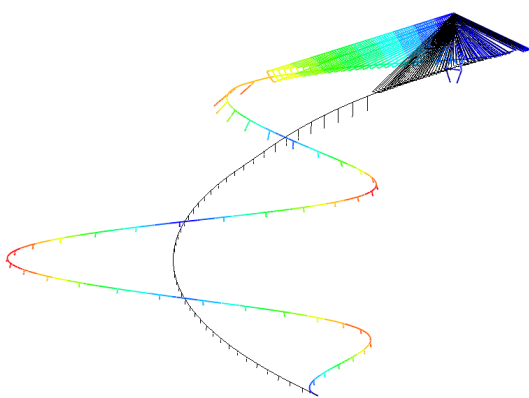


Figure 6-13: Mode 3

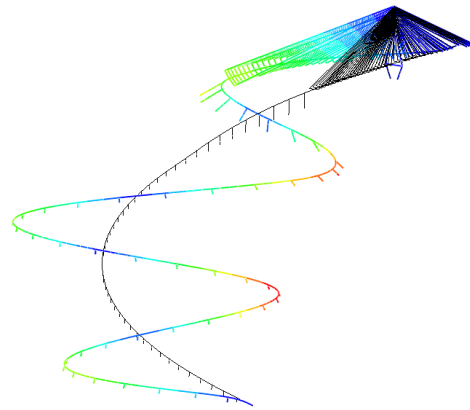


Figure 6-14: Mode 4

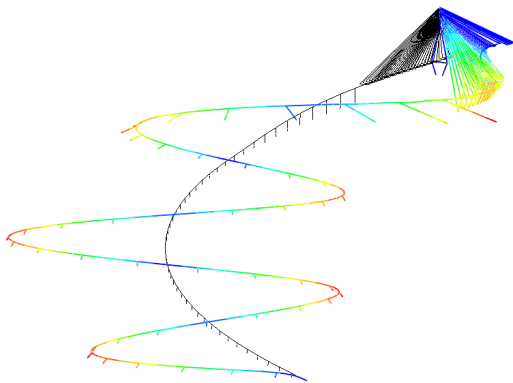


Figure 6-15: Mode 5

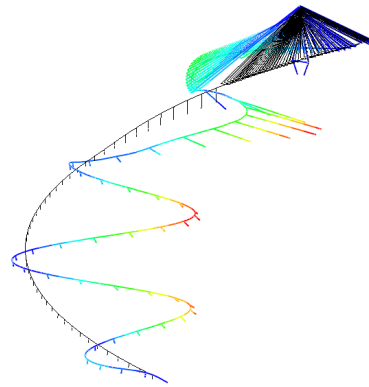


Figure 6-16: Mode 6

Displacement
XYZ Coordinates
High
Low

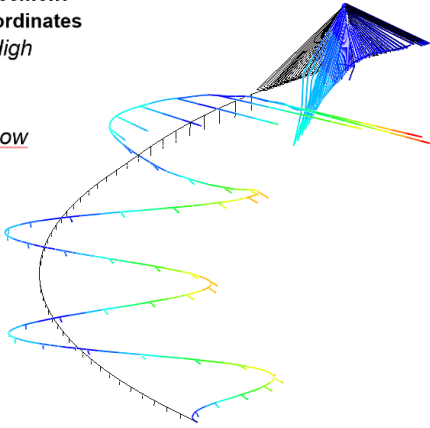


Figure 6-17: Mode 7

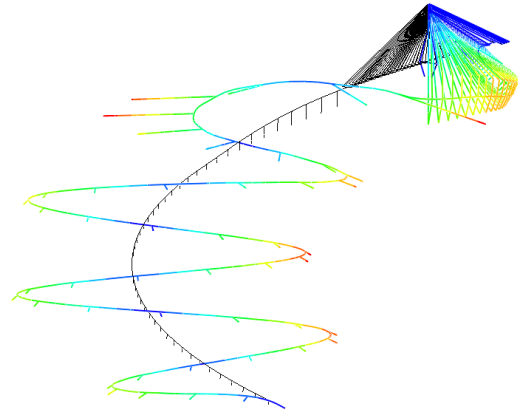


Figure 6-18: Mode 8

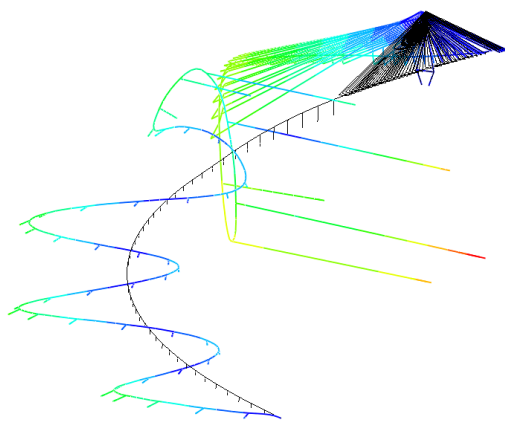


Figure 6-19: Mode 9

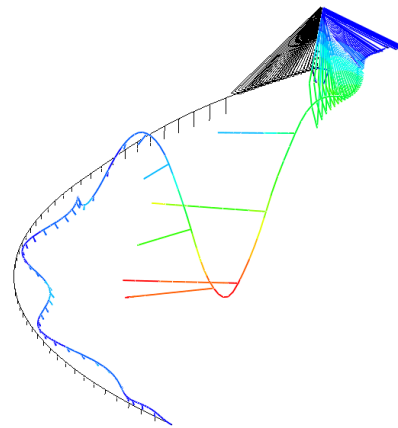


Figure 6-20: Mode 10

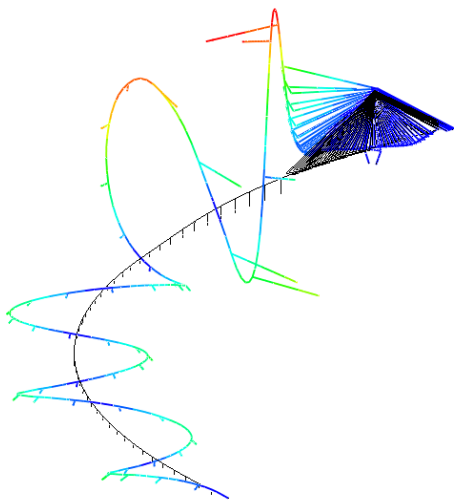


Figure 6-21: Mode 11

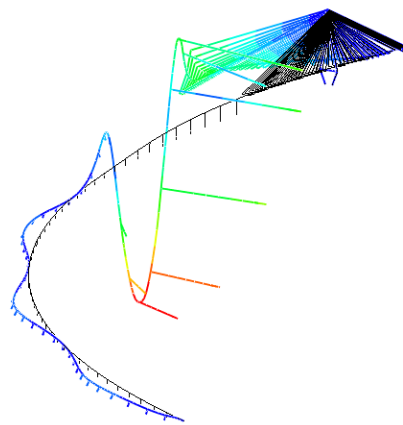


Figure 6-22: Mode 12

Displacement
XYZ Coordinates
High
Low

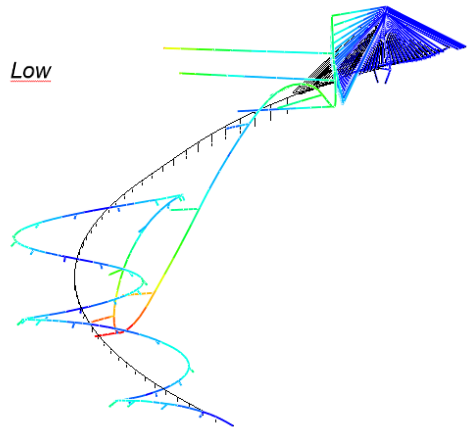


Figure 6-23: Mode 13

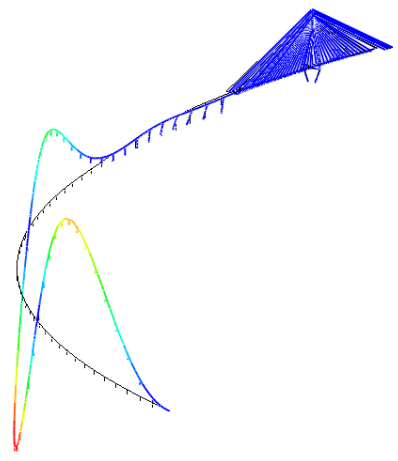


Figure 6-24: Mode 14

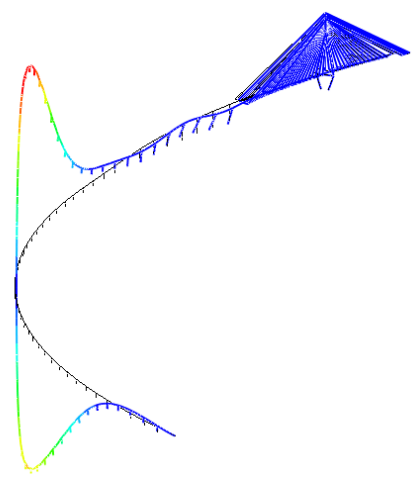


Figure 6-25: Mode 15

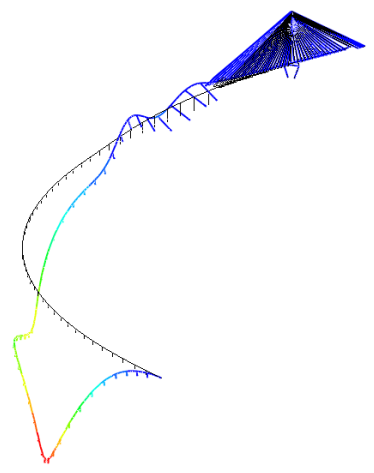


Figure 6-26: Mode 16

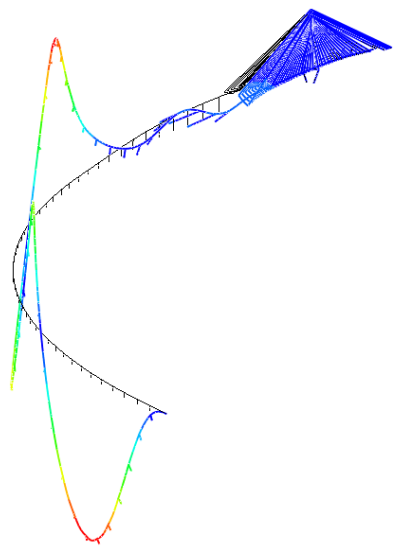


Figure 6-27: Mode 17

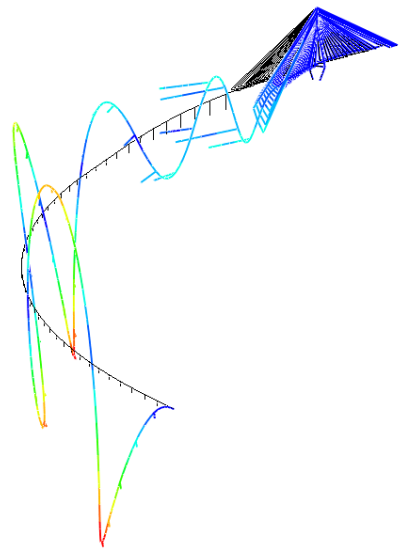


Figure 6-28: Mode 18

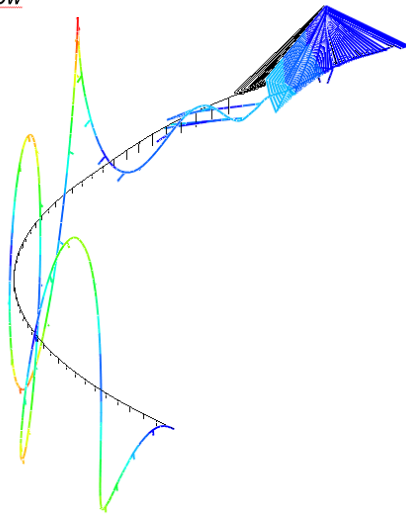
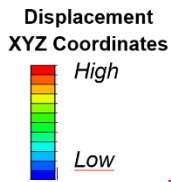


Figure 6-29: Mode 19

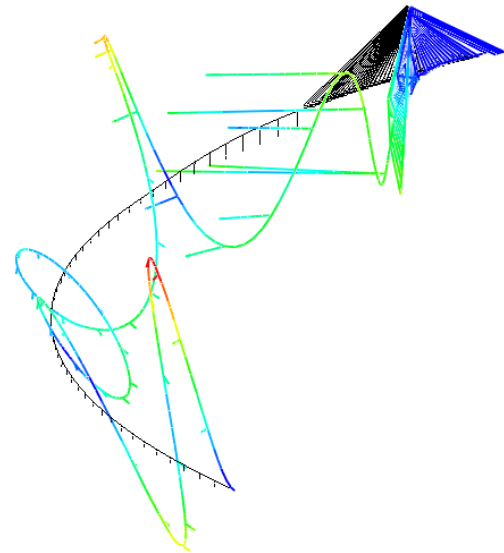


Figure 6-30: Mode 20

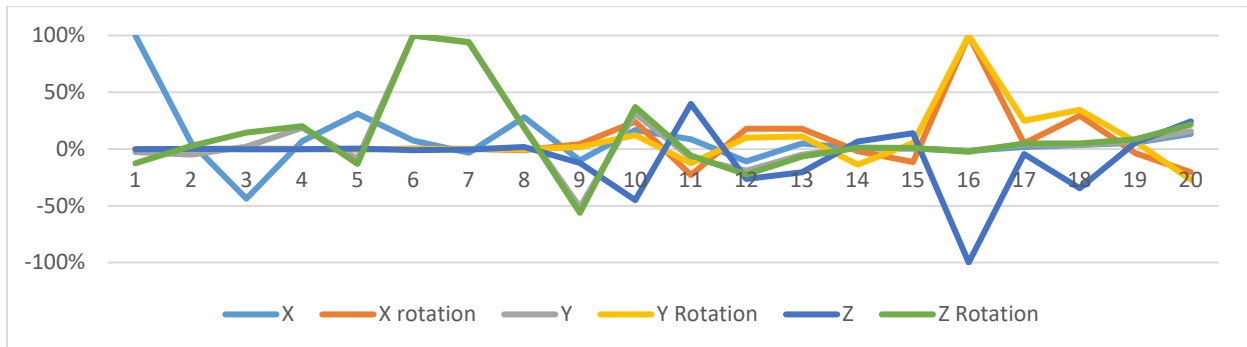


Figure 6-31: Scaled participation factors for the first 20 eigenmodes

In the figure above, participation factors calculated in Abaqus are scaled by dividing upon the maximum value of the respective DOF maximum participation factor achieved during the first 20 modes. Utilized to achieve a better understanding of the active DOF of the complex eigenmodes presented above.

$$\Gamma_{\alpha i} = \frac{1}{m_{\alpha}} \phi_{\alpha}^N M^{NM} T_i^M$$

Modal participation factors $\Gamma_{\alpha i}$ are calculated for each mode α and global direction i . Calculation of the modal participation factors is based on the normalized mass m_{α} , eigenvector for each mode ϕ_{α}^N , the mass matrix M and the magnitude of the motion in each DOF T . DOF In which subscripts N and M refer to degrees of freedom of the finite element model [14].

6.2.3 Comparing Eigenfrequencies

6.2.3.1 Added Mass Effect on Eigenfrequencies

The first modes are dominated by horizontal movement, such that the pontoons is largely participating. Thus the added mass is assumed to have a significant effect on the eigenperiods. Added mass effect is seen to increase the first eigenperiods, compared to the eigenperiods of the model without added mass, the first mode is 20 second longer with the added mass. Added mass is based on the area and shape of the pontoons, such that by changing the pontoon size or shape would imply changing the first eigenmodes significant. As the eigenperiods reduce, the effect of added mass is reduced significantly. It may be caused by lower horizontal participation of the higher modes along with other governing parameters such as stiffness in both vertical horizontal and torsional movement.

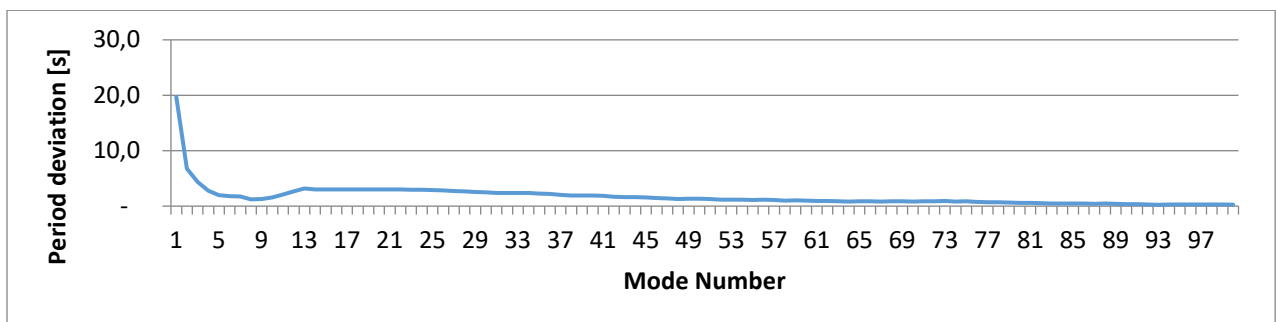


Figure 6-32: Increase in eigen-periods due to added mass effect for the first 100 modes

6.2.3.2 Comparing Prior Results

Comparing the results with prior results provided in the report for both added mass effect and without [28], a relatively small deviation is observed in the first modes. In the figure below, both added mass and no added mass graphs are compared to prior results on similar models which takes into account both added mass and without added mass. Similarity of the trends in both graphs are observed, however it is seen that the added mass eigenfrequencies computed deviate the least in the first modes, the eigenfrequencies of the model without added mass deviate the least in modes in between modes 12 through 35.

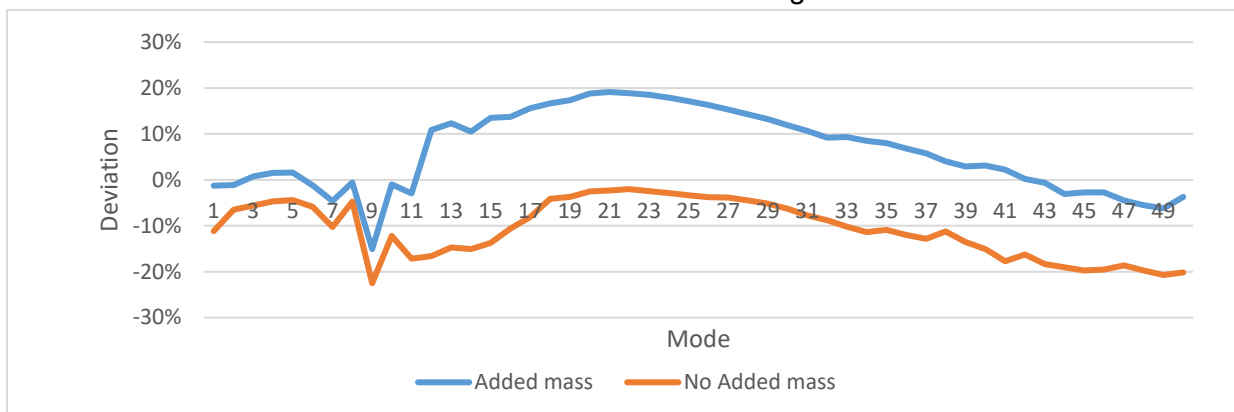


Figure 6-33: Deviation of obtained eigenperiods from those in the previous reports

6.2.3.3 Horizontal Curvature

To assess the effect of curvature on the eigenperiods, several eigenanalysis were carried out utilizing different horizontal curvature, it is found to impart only a small deviation of eigenfrequencies when changing the horizontal curvature.

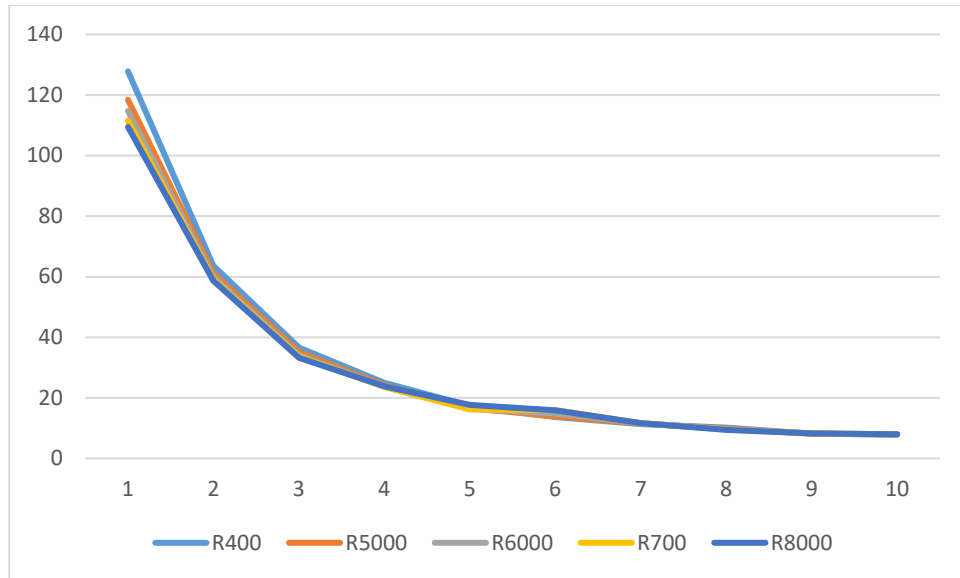


Figure 6-34: Eigenperiod comparison of different horizontal bridge girder curvatures

6.2.4 Damping

Considering the expected contribution of the modes to the response excited by the environmental loading, the first frequency to compute Rayleigh damping was chosen to be the first mode, the second frequency correspond to mode 100. Resulting in the circular frequency 0.0532 rad/s and 3.9678 rad/s. Targeting an effective damping ratio of 0,5% provided by Rayleigh damping at the first modes and at mode 50, the Rayleigh damping parameters are computed at mode 1 and mode 100. Even though the target effective Rayleigh damping ratio is 0,5% at mode 50, a greater effective damping ratio than expected was achieved, thus the second mode for calculating the Rayleigh damping parameters was chosen to mode 100.

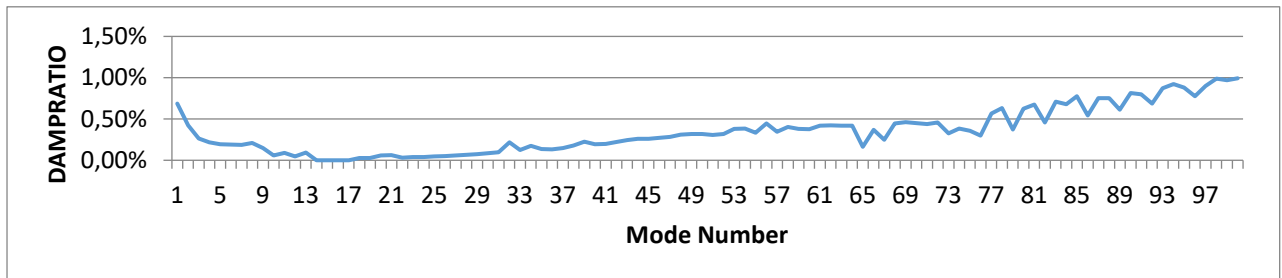


Figure 6-35: Effective damping ratio provided by structural damping, applied as Rayleigh damping.

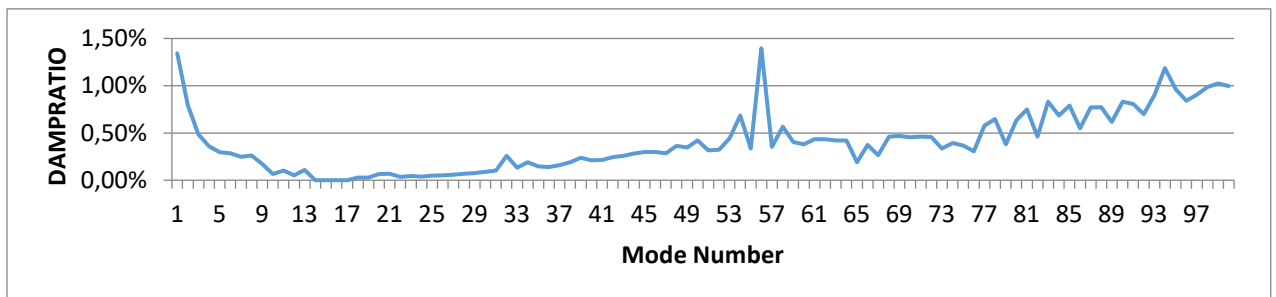


Figure 6-36: Total effective damping ratio

Damping Ratio Peak at Mode 56

An abnormal amount of damping ratio is achieved at mode 56 caused by the aerodynamic damping, which in the model is simulated as dashpots connected to the girder. Around mode 60 is when the cable tower is included in the mode shape, expecting a lower effective damping ratio since no damping is applied at the cable tower. If observing the mode in which the abnormal damping ratio is achieved, the girder is observed to be both included in form of horizontal, vertical and rotational displacement. Resulting in activation of the aerodynamic dampers in the whole girder length.

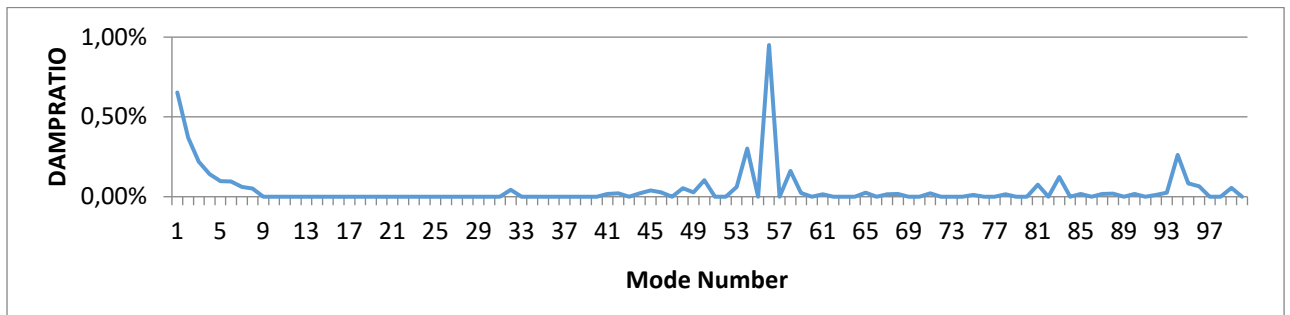


Figure 6-37: Effective damping ratio provided by aerodynamic dampers

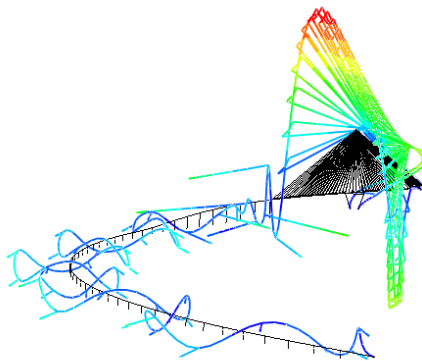


Figure 6-38: Mode 56 with abnormal effective damping ratio

6.2.5 Element Formulation

6.2.5.1 Euler Bernoulli vs Timoshenko

Illustrated on the below graphs, it can be observed that the adopted element formulation have negligible effect on the eigenperiods and frequencies of the structure in the first modes with eigenmodes below 100. As the frequency increase the deviation increase significantly. The Euler-Bernoulli formulated model, termed B33, is achieving about 15%-25% greater eigenfrequencies at modes around 500, corresponding to a eigenfrequencies of about 10 Hz. Higher eigenfrequencies is in general caused by greater stiffness or lower mass. Mass formulation utilized for the two elements are the same, such that the difference is most likely caused the stiffness formulation of the elements. In Timoshenko elements, termed B31, shear deformation is allowed. Thus the formulation of the B31 elements results in a less stiffer beam, thus the results showcase that the lower frequency obtained by the B31 elements are an effect of the shear flexible formulation.

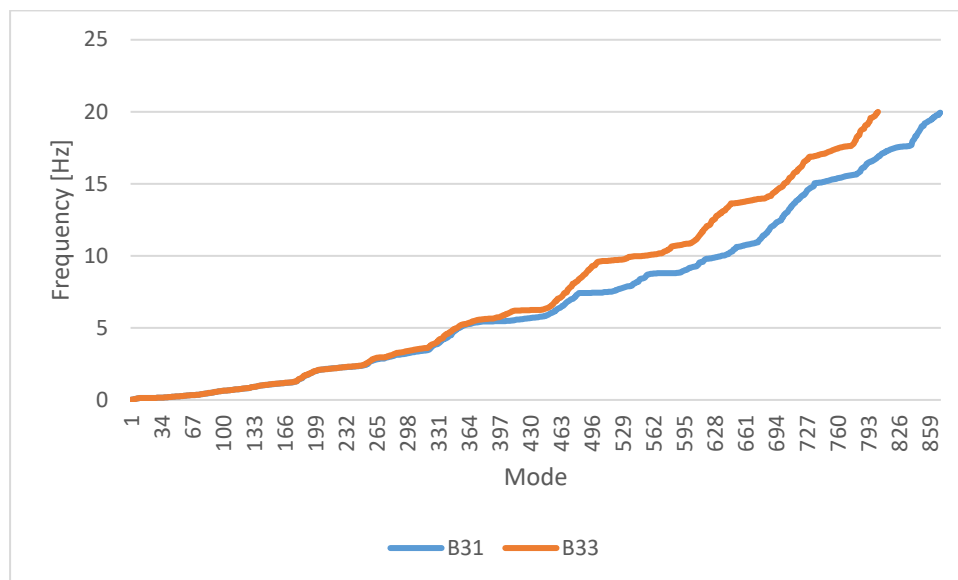


Figure 6-39: Comparing eigenfrequencies of B31 and B33 based structure

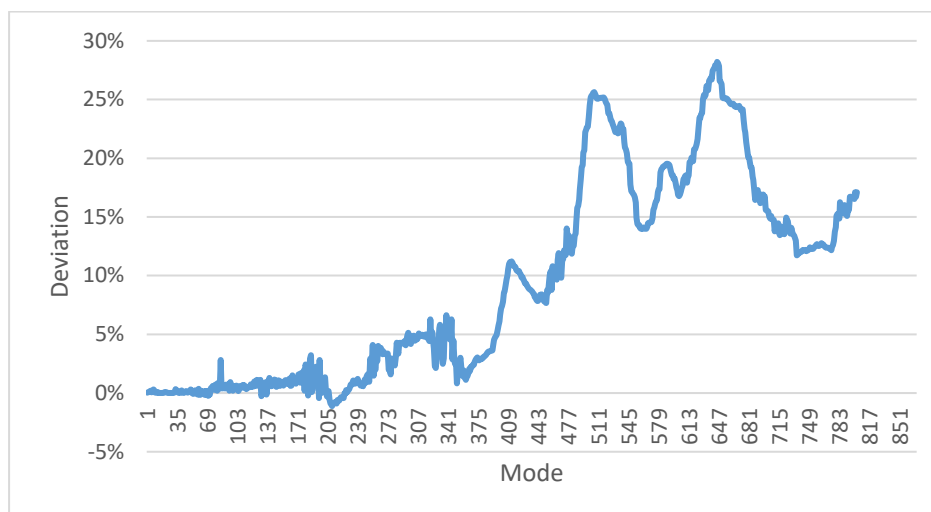


Figure 6-40: Deviation in eigenfrequency as function of eigenperiods B31 vs B33 element formulation

6.2.5.2 Element Size

An analysis of two separate element size formulation of the bridge girder was done, in which the eigenperiods are analyzed. FE models are expected to converge as the element size is adequate small. A simple check is carried out to find when the element size converges and which element size is appropriate for the bridge girder.

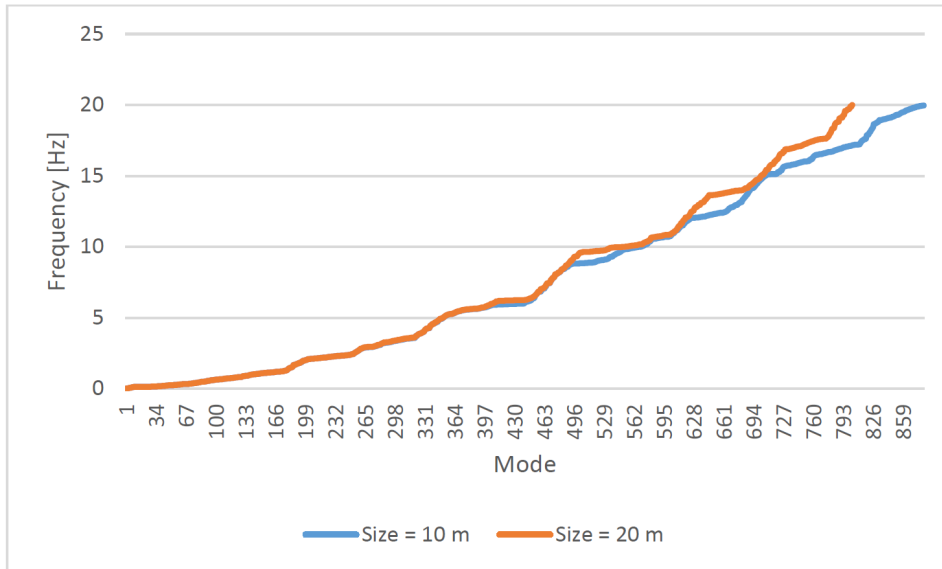


Figure 6-41: Comparing Eigenfrequencies of girder element size

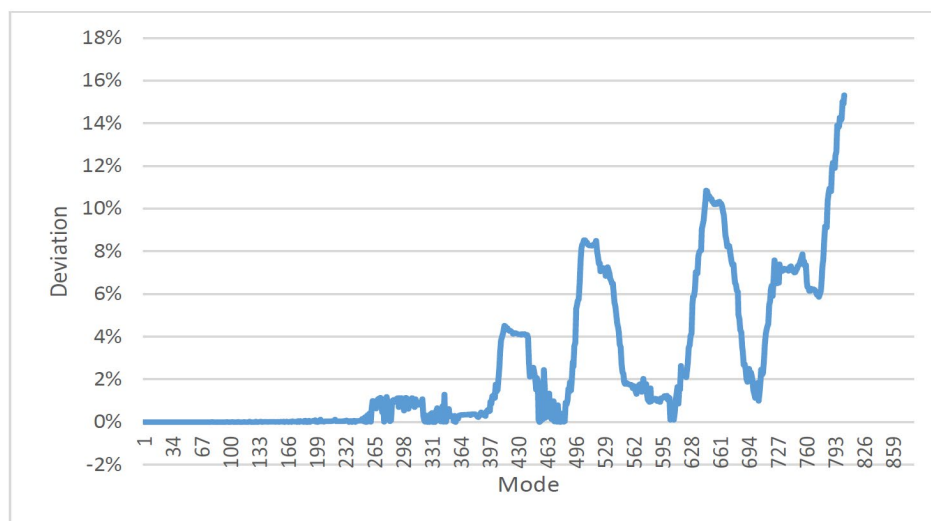


Figure 6-42: Deviation of girder element size 20 vs 10 m

In the figures above, a great variation of the deviation is observed in the higher modes. By observing the mode shape, the least deviate modes are mostly governed by the cable stayed tower and cables which are formulated with the same element size in both figures above. Modes with the greatest deviation includes large displacements of the bridge girder.

6.3 Wind Field

6.3.1 Turbulence Components

In the figures below, turbulence components are provided based on simulating the wind conditions with the 100 year return period and wind direction 280° , corresponding to wind from west. As expected the turbulence components vary in magnitude $u > v > w$. Caused by the turbulence intensity, in which $I_u > I_v > I_w$.

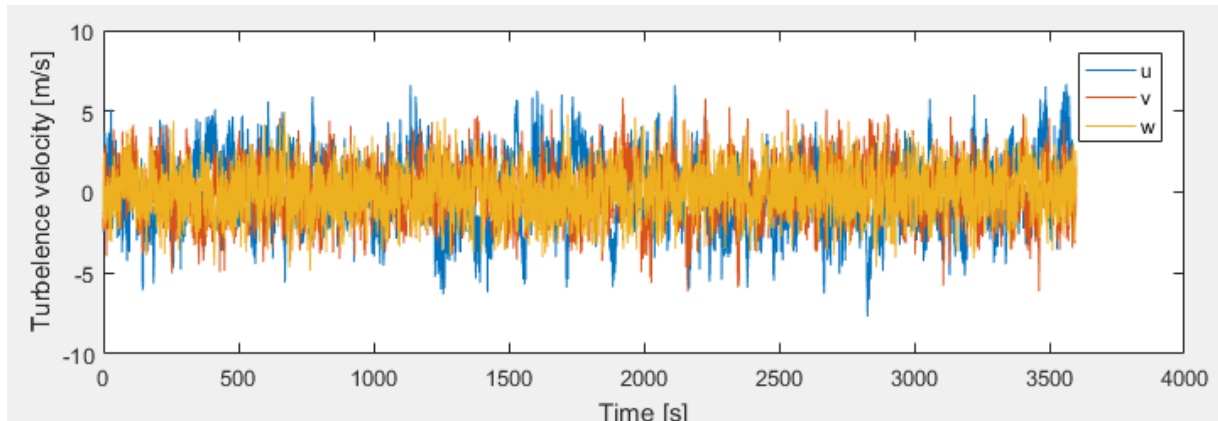


Figure 6-43: Turbulence components at height 228 m

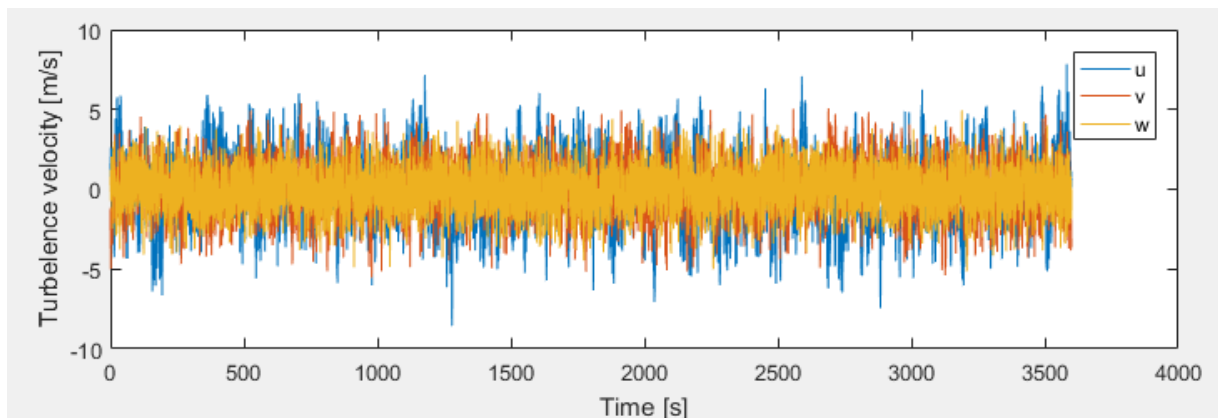


Figure 6-44: Turbulence components at height 50 m

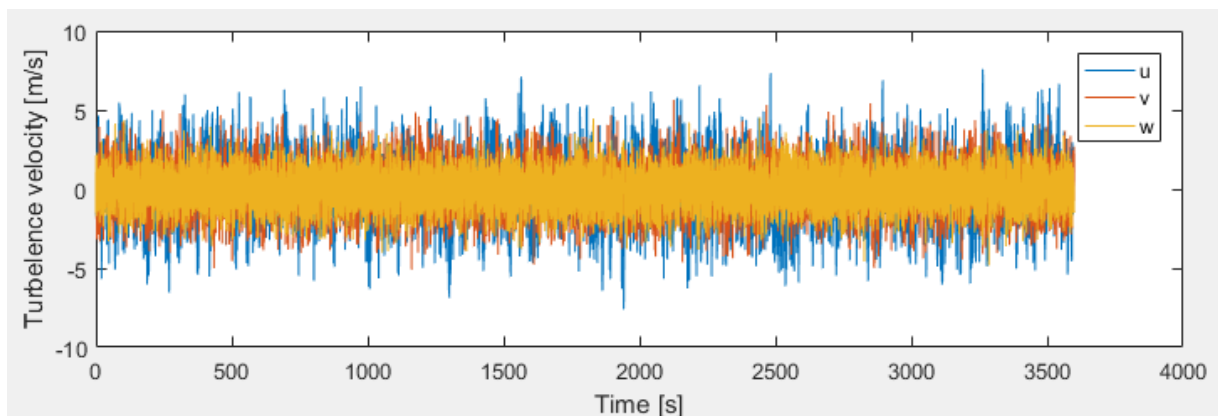


Figure 6-45: Turbulence components at height 16 m

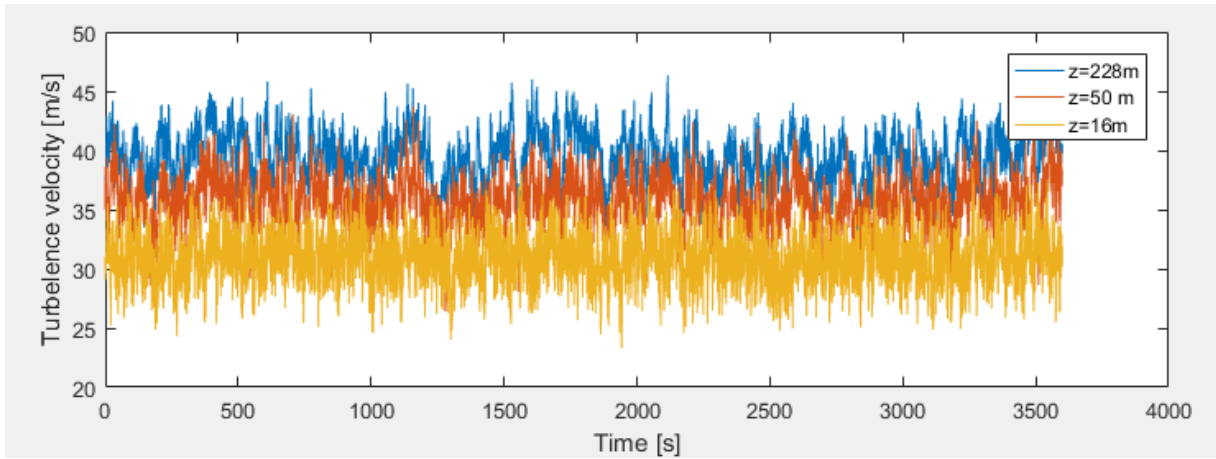


Figure 6-46: Total along wind velocity at three heights

6.3.2 Frequency Spectra

6.3.2.1 Obtained Turbulence Spectra

In the following figures, the obtained turbulence spectra is illustrated for all sets. Dotted line is based on the Kaimal spectrum, and the continuous graph is the obtained spectra. For several sets some random variation in the lower frequencies is observed compared to the Kaimal spectrum. Low frequencies of around 10^{-2} Hz are important for analyzing the bridge due to the long eigenperiods which is the main mode which govern the dynamic response, which should be kept in mind when viewing the results from the dynamic loading.

Comparing the 3D W280L set and 4D 255C, indication of greater accuracy for the 3 dimensional frequency coherence spectrum is seen at the lower frequencies, it might be explained by the emphasis on fewer dimensions.

Wind applied from the east achieves a greater spectra than the target at the low frequencies several times, observed in sets W100C. Comparing to the wind from west in set W280C is seem to be an effect of applying the wind perpendicular to the bridge. Wind applied perpendicular to the wind has the greatest simulated area when combining both space dimensions. Wind simulated from the west and east should thus have been simulated with a greater sampling frequencies such that the spectra obtained would have been more accurate.

W255C
 Direction: 255
 Wind field: 4D

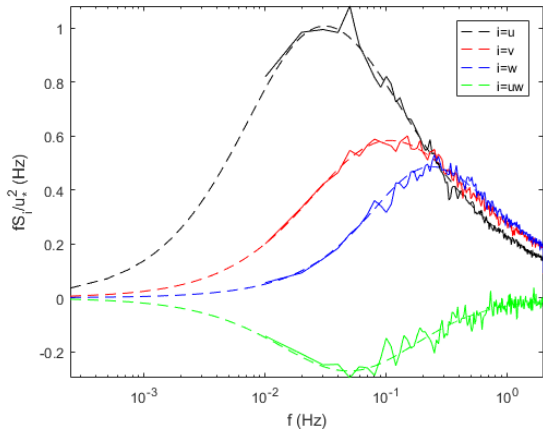


Figure 6-47: Targeted and simulated Spectra, W255C, seed 1

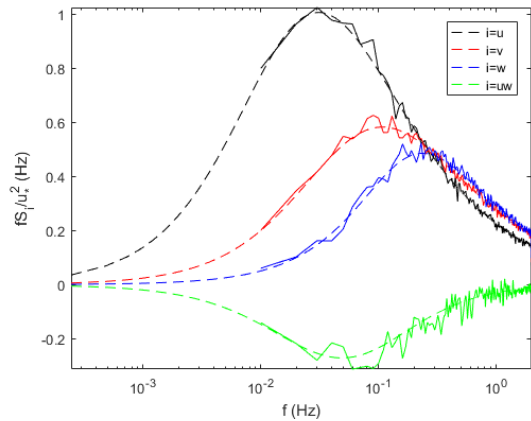


Figure 6-48: Targeted and simulated Spectra, W255C, seed 2

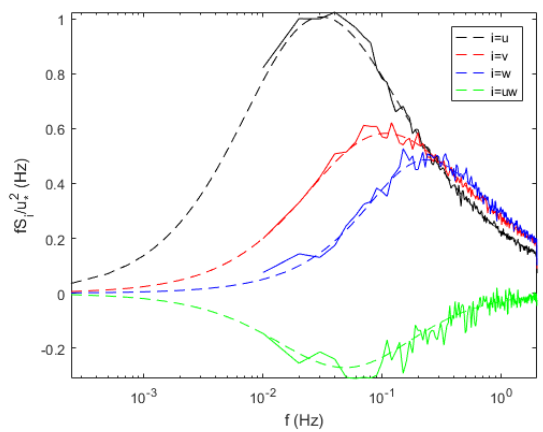


Figure 6-49: Targeted and simulated Spectra, W255C, seed 3

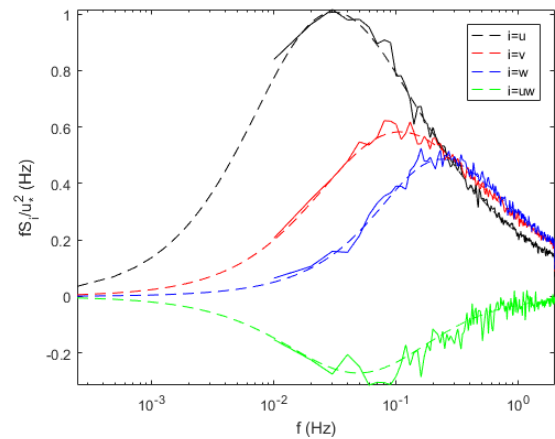


Figure 6-50: Targeted and simulated Spectra, W255C, seed 4

W280C
 Direction: 280
 Wind field: 4D

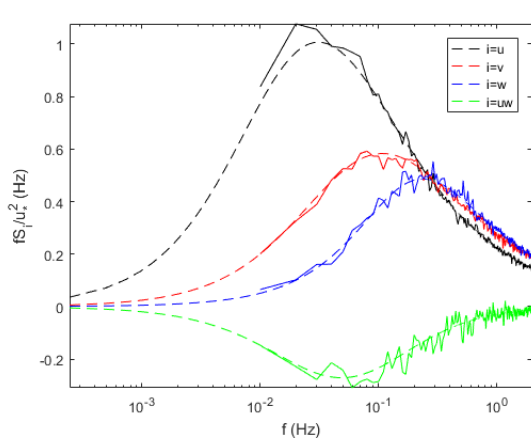


Figure 6-51: Targeted and simulated Spectra, W280C, seed 1

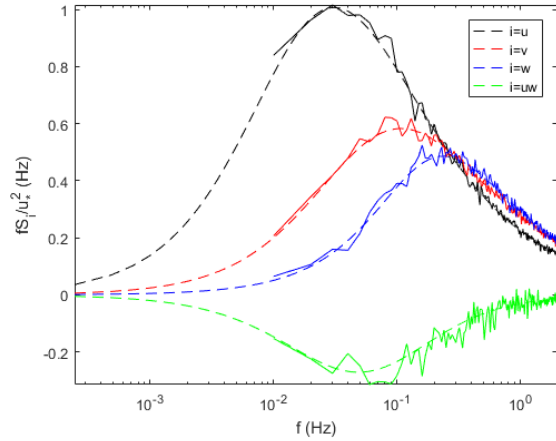


Figure 6-52: Targeted and simulated Spectra, W280C, seed 2

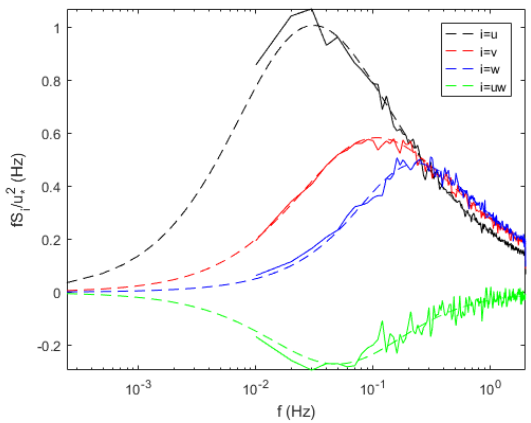


Figure 6-53: Targeted and simulated Spectra, W280C, seed 3

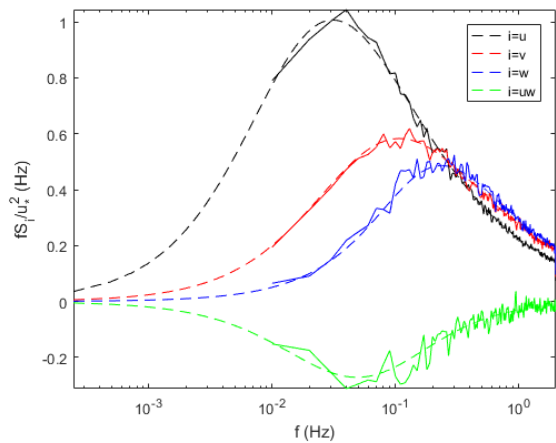


Figure 6-54: Targeted and simulated Spectra, W280C, seed 4

W345C

Wind direction: 345

Wind field: 4D

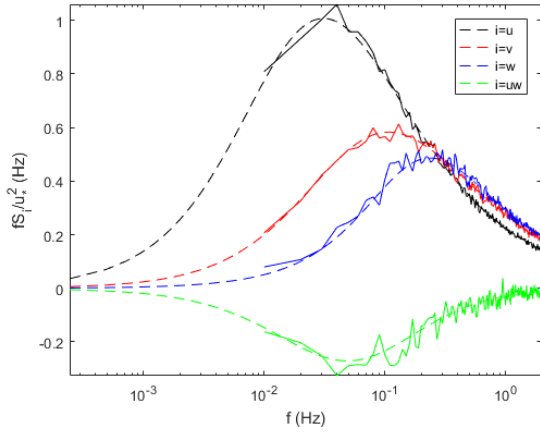


Figure 6-55: Targeted and simulated Spectra, W345C, seed 1

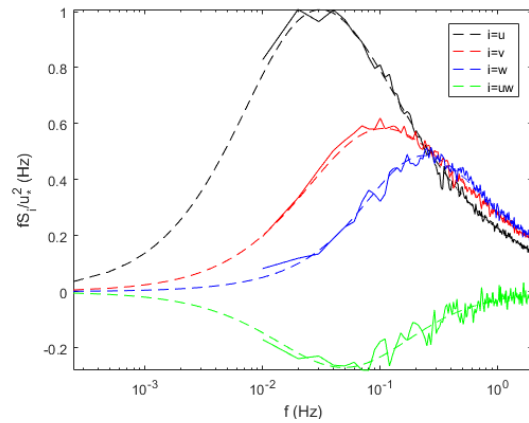


Figure 6-56: Targeted and simulated Spectra, W345C, seed 2

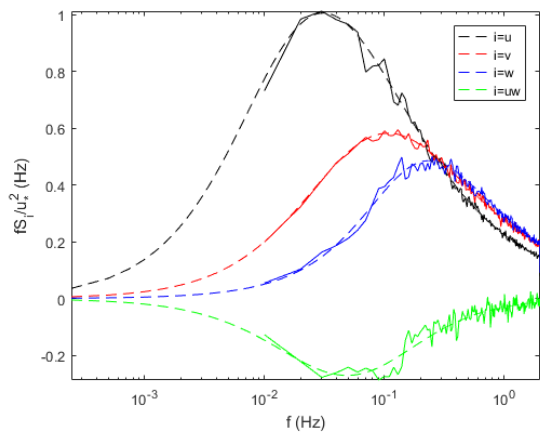


Figure 6-57: Targeted and simulated Spectra, W345C, seed 3

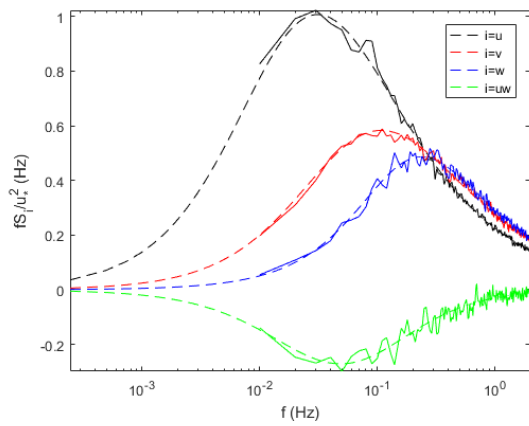


Figure 6-58: Targeted and simulated Spectra, W345C, seed 4

W100C
 Wind direction: 100
 Wind field: 4D

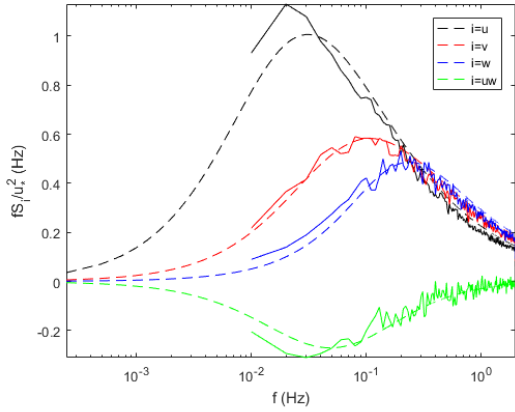


Figure 6-59: Targeted and simulated Spectra, W100C, seed 1

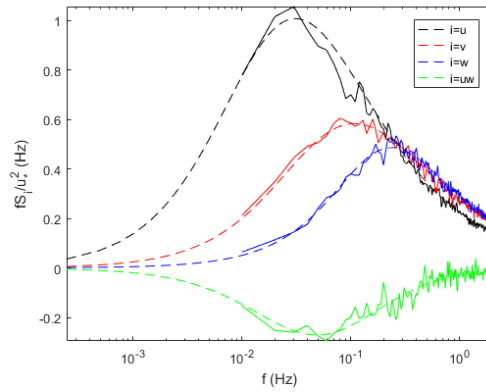


Figure 6-60: Targeted and simulated Spectra, W100C, seed 2

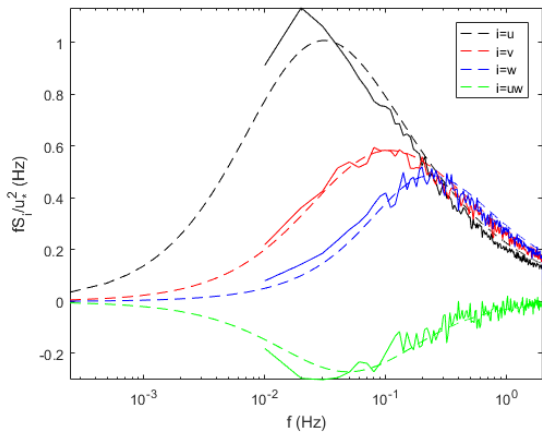


Figure 6-61: Targeted and simulated Spectra, W100C, seed 3

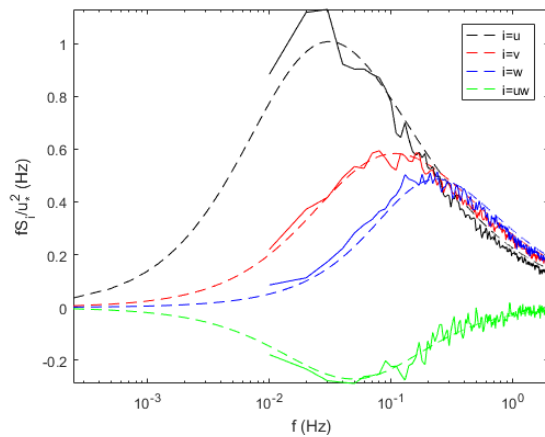


Figure 6-62: Targeted and simulated Spectra, W100C, seed 4

W280L
 Wind direction: 280
 Wind field: 3D

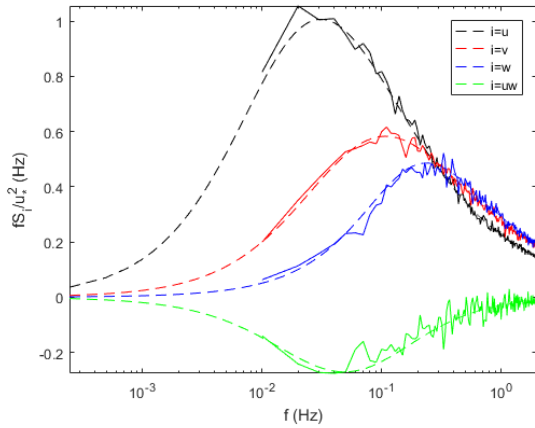


Figure 6-63: Targeted and simulated Spectra, W280L, 3D, seed 1

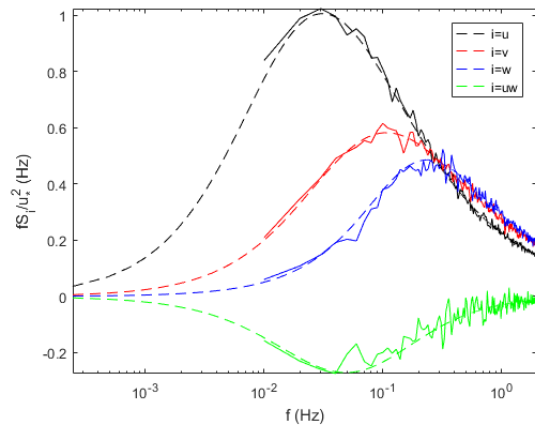


Figure 6-64: Targeted and simulated Spectra, W280L, 3D, seed 2

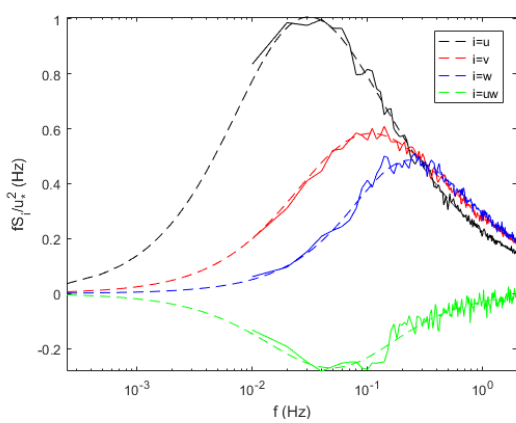


Figure 6-65: Targeted and simulated Spectra, W280L, 3D, seed 3

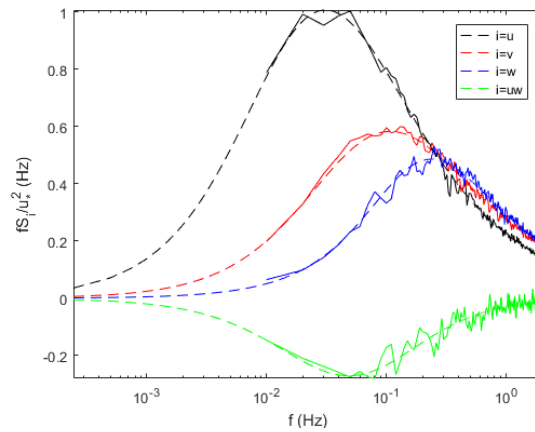


Figure 6-66: Targeted and simulated Spectra, W280L, 3D, seed 4

6.3.2.2 Visualization of Turbulence

To assess a turbulent wind field is difficult given coherent wind field the large amounts of data. In the figure below the turbulence component u is illustrated by time axis at each node in which the turbulence is computed, magnitude of turbulence u is given in m/s according to the color bar. Nodes are provided in four categories; girder, cables, pontoon and cable tower. Nodes 1-271 are the girder nodes, in which the height is varying providing a varying pattern across the nodes. However, the first nodes are positioned at height 50 m, the last nodes are positioned at height 16 m. As the height reduces, the pattern seems to increase randomness - reminiscent of white noise.

Nodes 370-489 are the pontoon columns nodes which are placed in heights between 5 – 16 m. Nodes 490-580 is the tower nodes positioned at height 5 m to 228 m.

Comparing the two intervals in the figure below, larger coherent yellow and blue areas are observed as the height of nodes increase, which is in line with the coherence model in which the coherence is increased with height both in time and space axis.

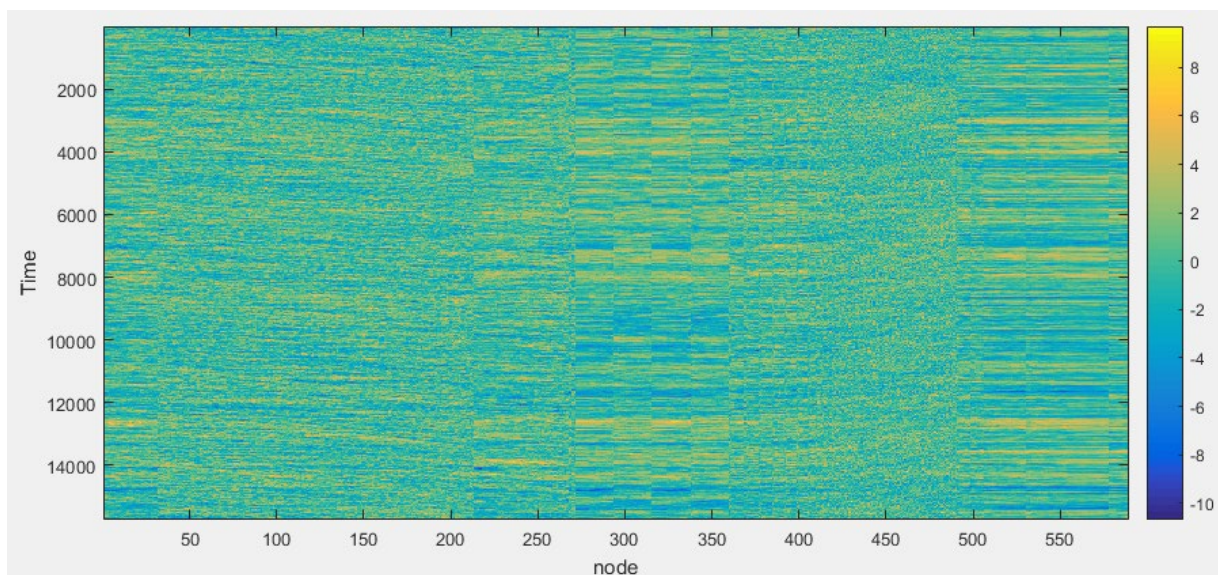


Figure 6-67: Illustration of the along wind turbulence, wind speed in m/s according to the colorbar

6.4 Bridge Girder Response

6.4.1 Dynamic Loading

Below represent the bridge girder responses measured in displacement, acceleration, bending moments and axial force. The measurements is the maximum observed values through the dynamic time history analysis. Each measurement is provided with the standard deviation calculated of the data set, in each set 4 analysis is carried out based on different turbulence seeds, providing a sense of the variability and uncertainty in tthe results. Measurements are only carried out from 500 seconds in the analysis in order to not obtain faulty measures caused by the transition between static and dynamic analysis.

Table 6-4: Magnitude of maximum values of bridge girder responses

<i>Set ID</i>	<i>Displacement [m]</i>	<i>Std. Dev [m]</i>	<i>Acceleration [m/s²]</i>	<i>Std. Dev [m/s²]</i>
<i>W255C</i>	46,1	0,3	2,07	0,09
<i>W280C</i>	37,4	4,1	2,23	0,14
<i>W345C</i>	4,9	0,1	0,78	0,71
<i>W100C</i>	43,5	6,4	1,48	0,14
<i>W280L</i>	39,5	1,8	1,99	0,03

Displacement

Evidently from the table presented, the magnitude of displacement of the bridge girder varies with the wind direction. The southwest wind coming from 255°, with a yaw angle of 65° on the normal of the middle tangent of the bridge girder provide the greatest displacement. To account for the horizontal loading of the bridge girder, the bridge has an arched shaped which increase the stiffness for evenly distributed load. However the geometric shape results in lower capacity for unsymmetrical loading. Each load component of the wind acting in the horizontal plane is decomposed to a normal force at each bridge girder element. As an effect, wind with an yaw angle of 0° to the bridge normal at the middle of the bridge will result in even distributed loads across the bridge girder, opposed to bridge with an inclined angle which will result in a higher loads at either the south or north end to the curvature and thus changing orientation of each bridge girder element.

Even though the load applied in W280C which is based on wind load perpendicular to the bridge has the largest magnitude of load, it is also the most evenly distributed as opposed to the load applied in W255C which results in a distributed load with greatest magnitude at the north end. W255C applied load in which the displacement of the structure resembles mode shape 1 of the bridge structure, in which the bridge has the least resistance to displacement.

Set W100C in which the wind is applied from the east also have significant displacement, even though the mean wind velocities are reduced with a reduction factor. Bridge girder is evidently suspect to large displacement caused by winds applied from the east, which may be caused by the perpendicular load to the girder in which a high load magnitude is applied at the middle of the span, which causes a deflection similar to mode shape 2. W100C composed of 100° wind direction also provides great displacement. Evidently the wind hitting the wind perpendicular to the deck midspan provides the greatest forces. However, the W100C sets is loaded with wind utilizing a 0,85 reduction factor, such that the wind field is in fact constructed of lower mean wind velocity. Forces applied from east also provides

compressive axial forces in the girder, such that a small effect on the large displacement might be caused by a buckling behavior.

Acceleration

Acceleration of the bridge girder varies greatly with the wind field direction. Wind coming from North-West produce the least acceleration of the girder. Wind coming from all other directions is almost in the same interval around 2 m/s², no design codes exists of the acceleration of bridge girder, the measured acceleration is very high. It seems that the acceleration is coinciding with the total forces which is applied to the bridge girder. In which case W280C has the greatest force magnitude due to being perpendicular to the bridge, second is the W255C set in which the load is applied almost perpendicular to the bridge. W100C also applies wind perpendicular to the bridge, however it is simulated utilizing a reduction factor on the mean wind velocity. W345C applies the least force to the bridge, due to the high angle of the bridge.

Table 6-5: Maximum forces and moments observed

Set ID	Moment [MNm]				Axial Force [MN]	
	Strong Axis	Std. Dev.	Weak Axis	Std. Dev.	Main Axis	Std. Dev.
W255C	6300,00	187,08	361,25	187,08	175,00	4,12
W280C	6000,00	70,71	312,50	70,71	169,75	5,45
W345C	-3170,00	11,18	-462,00	11,18	-138,75	1,92
W100C	-6600,00	339,12	-347,50	339,12	-302,50	14,79
W280L	5850,00	111,80	350,00	111,80	172,25	4,77

Section Forces and Moments

During the 1hr dynamic loading step section forces and moment I measured and the maximum values are provided in the table. Strong axis moment are noted closed to the supports. Large deviations are noted, the probability of achieving the given forces might therefore be very low, since the variation between the sets are high.

3D vs 4D Turbulence Simulation

Comparing the 3D W280L set with 4D 280C set it is evidently a small deviation of all measures. Variation of the measures may simply be explained by the large standard deviation and the few numbered seeds, resulting in large uncertainty of the numbers.

Time History Acceleration

To illustrate the acceleration and which part is most suspect to high acceleration the figure above illustrates the variation of acceleration through the bridge girder. In all analysis mostly the high bridge is observed to have the greatest acceleration. In the span which the bridge transition from cable stayed to pontoon stayed the bridge does not have the mass inertia effect provided by the added mass of pontoons.

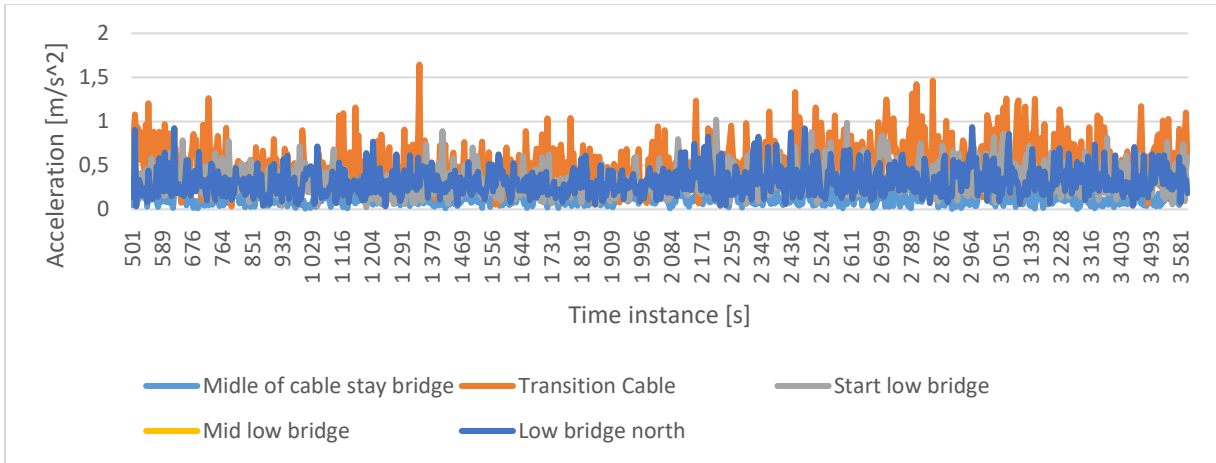


Figure 6-68: W255C seed 4 acceleration magnitude measurements at intermediate nodes

Energy Dissipation

Energy is dissipated with the velocity of the structure, calculated in Abaqus as the product of velocity and damping coefficient. Thus high dissipation indicate that the bridge has a high velocity, if the energy dissipation suddenly increase, the velocity is increasing at the structure might experience resonance. Evidently, the energy dissipation is quite similar for all sets excluding the set in which the wind is coming from NW.

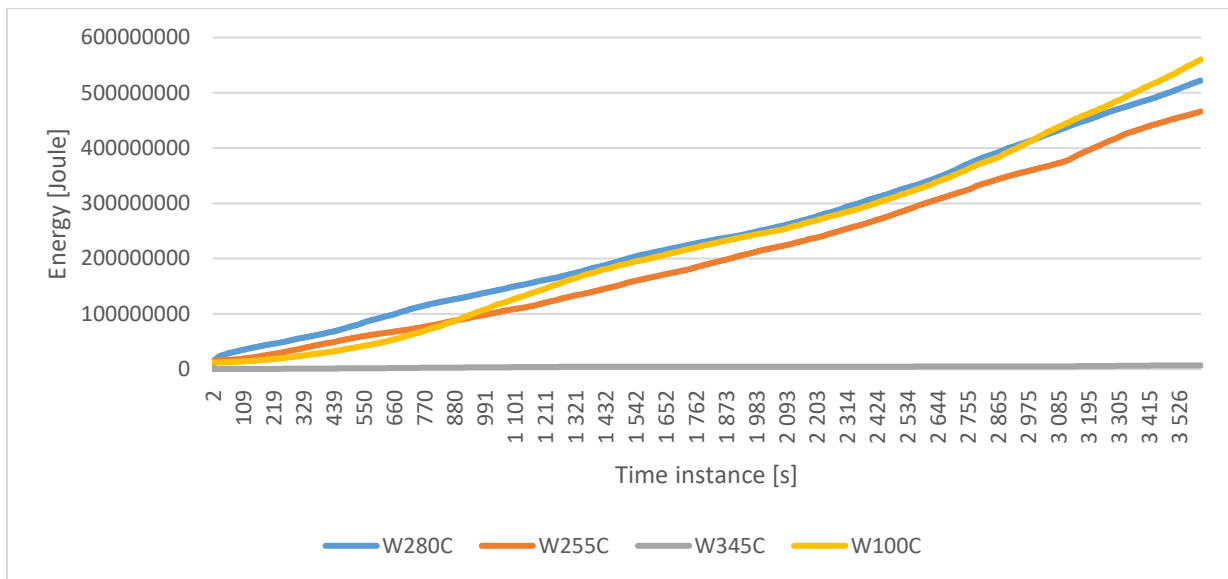


Figure 6-69: Energy dissipation of all set trough dynamic wind loading

6.4.2 Study of Response Periods

On set W280C and W255C in which the greatest acceleration and displacement was noted, four points were placed on the bridge girder in order to measure the time variation of displacement and further investigate the period of oscillations. In each figure two graphs are provided, the first including the displacement response during the 1 hour long analysis and the second providing a closer look at the time interval in which the highest displacement is observed. Simple estimates of the periods of predominant period of motion, and is provided below the figures.

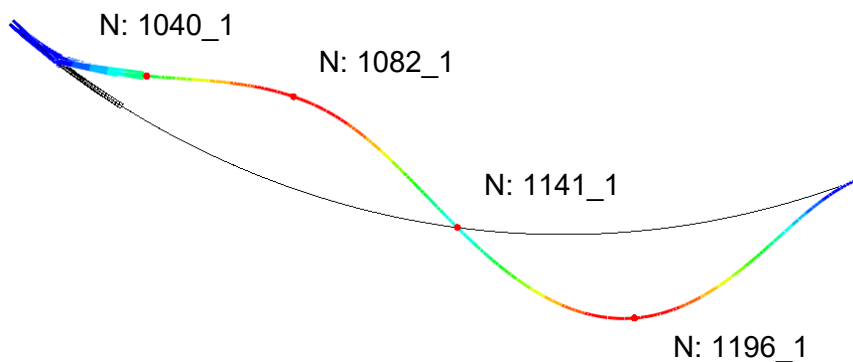


Figure 6-70: Points assessed on bridge girder, plotted on mode shape 1

By studying the following figures and simple period estimates, the major part of the nodes follow either the first or second modes which is expected. It is observed that when all nodes are oscillating displacements in phase at same frequency, the magnitude of displacement increases. Largest displacement magnitudes are seen at the point corresponding to the middle of the high bridge, node 1040_1. Through most of the time histories, the largest magnitudes are observed after 30 minutes of time simulation, by increasing the time period to allow for additional oscillations might have resulted in larger displacement due to resonance.

W280C seed 1

Point	Period [s]	Legend
1040_1	50	U: N: 1040_1
1082_1	120	U: N: 1082_1
1141_1	50	U: N: 1141_1
1196_1	120	U: N: 1196_1

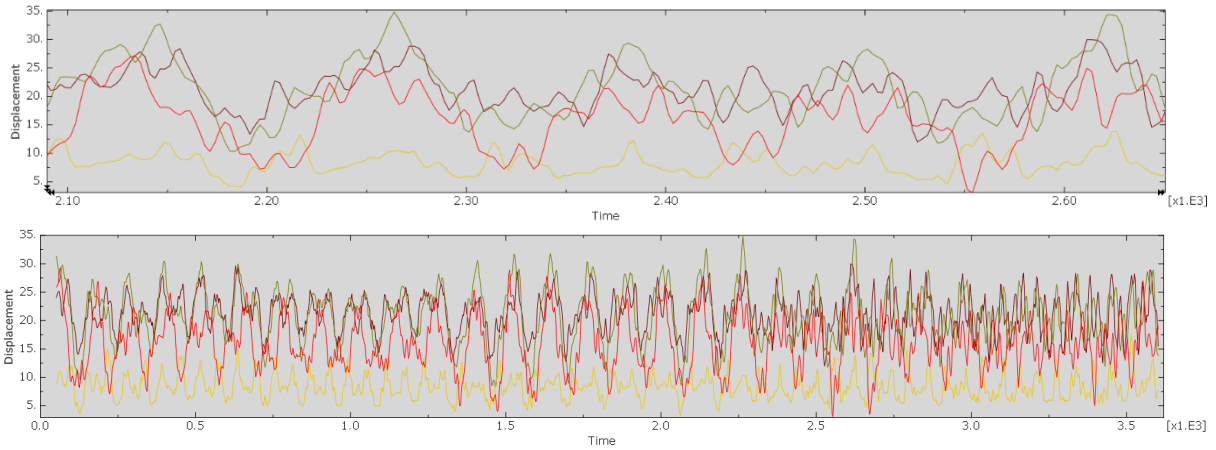


Figure 6-71: Displacement magnitude XY plane W280C seed 1

W280 seed 2

Point	Period [s]	Legend
1040_1	50	U: N: 1040_1
1082_1	1120	U: N: 1082_1
1141_1	50	U: N: 1141_1
1196_1	120	U: N: 1196_1

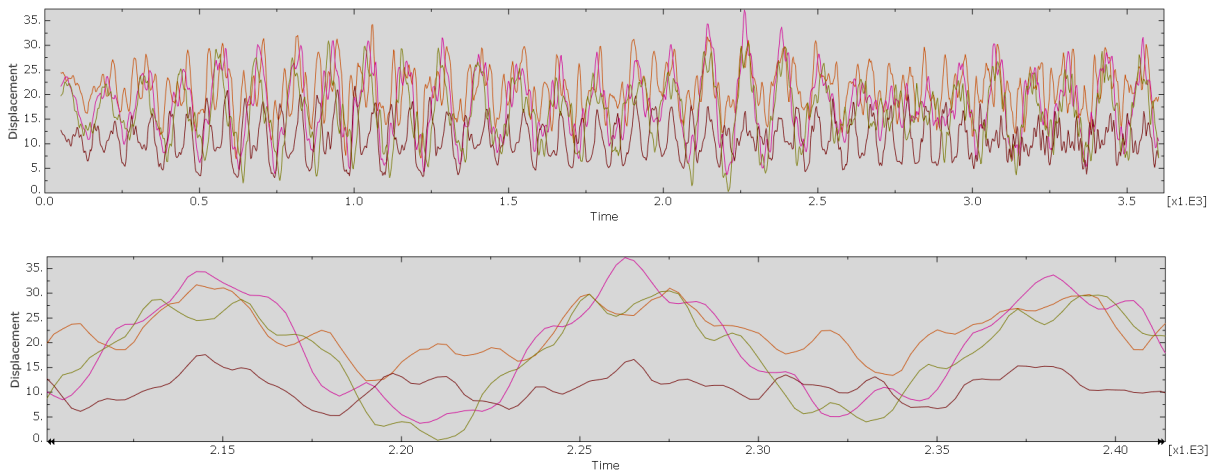


Figure 6-72: Displacement magnitude XY plane W280C seed 2

W280: Seed 3

Point	Period [s]	Legend
1040_1	120	U: N: 1040_1
1082_1	120	U: N: 1082_1
1141_1	120	U: N: 1141_1
1196_1	120	U: N: 1196_1

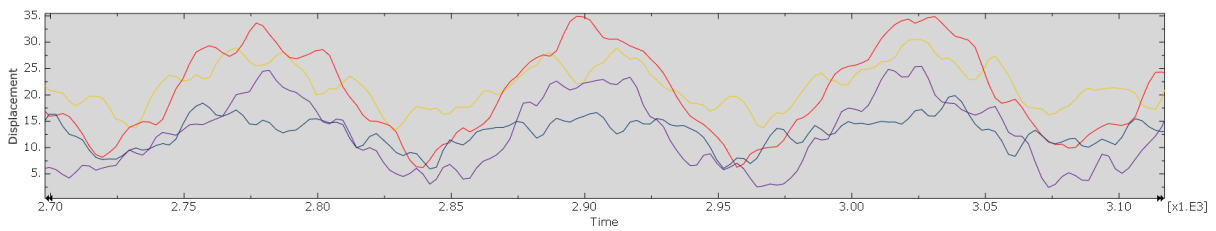
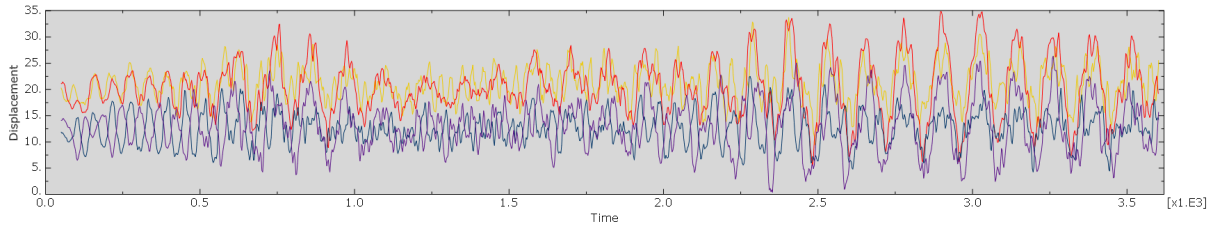


Figure 6-73: Displacement magnitude XY plane W280C seed 3

W280, seed 4

Point	Period [s]	Legend
1040_1	120	U: N: 1040_1
1082_1	120	U: N: 1082_1
1141_1	na	U: N: 1141_1
1196_1	120	U: N: 1196_1

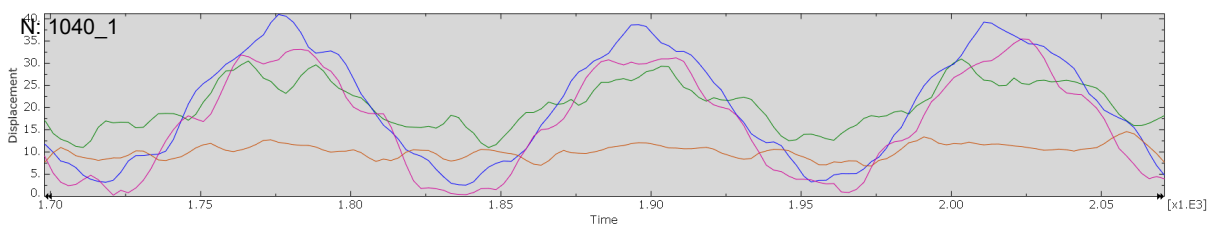
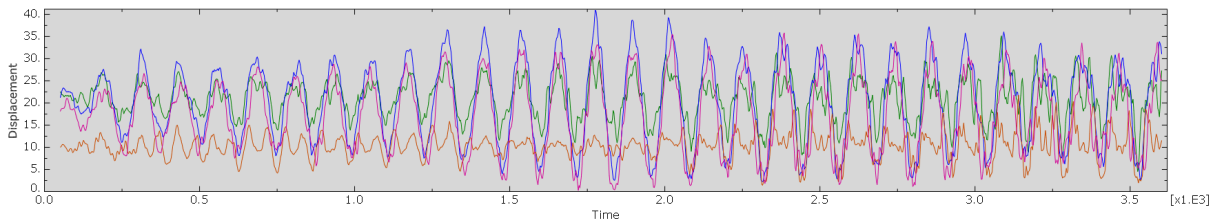


Figure 6-74: Displacement magnitude XY plane W280C seed 4

W255, seed 1

Point	Period [s]	Legend
1040_1	50	U: N: 1040_1
1082_1	120	U: N: 1082_1
1141_1	50	U: N: 1141_1
1196_1	120	U: N: 1196_1

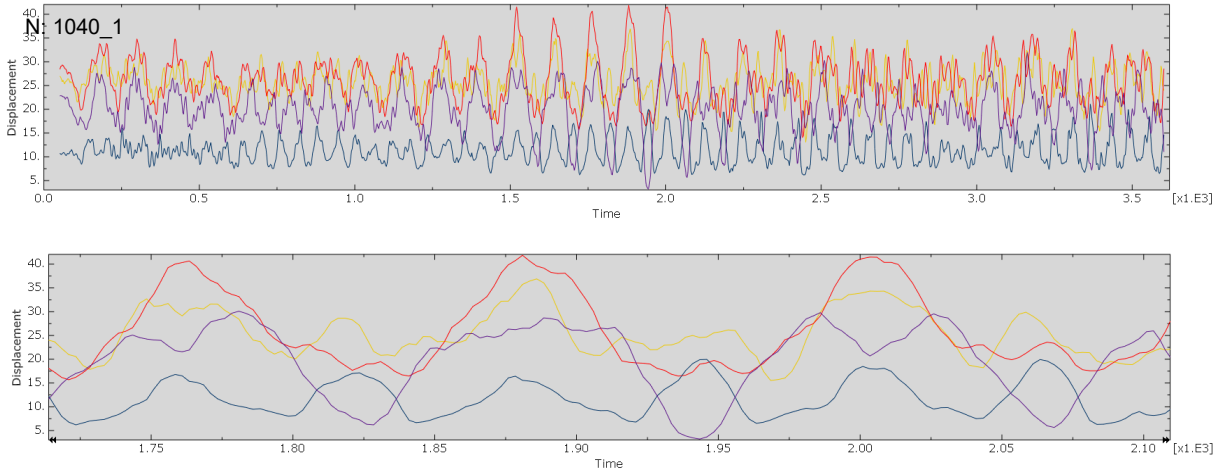


Figure 6-75: Displacement magnitude XY plane W255C seed 1

W255 seed 2

Point	Period [s]	Legend
1040_1	40	U: N: 1040_1
1082_1	130	U: N: 1082_1
1141_1	na	U: N: 1141_1
1196_1	130	U: N: 1196_1



Figure 6-76: Displacement magnitude XY plane W255C seed 2

W255 seed 3

Point	Period [s]	Legend
1040_1	120	U:N: 1040_1
1082_1	120	U:N: 1082_1
1141_1	na	U:N: 1141_1
1196_1	120	U:N: 1196_1

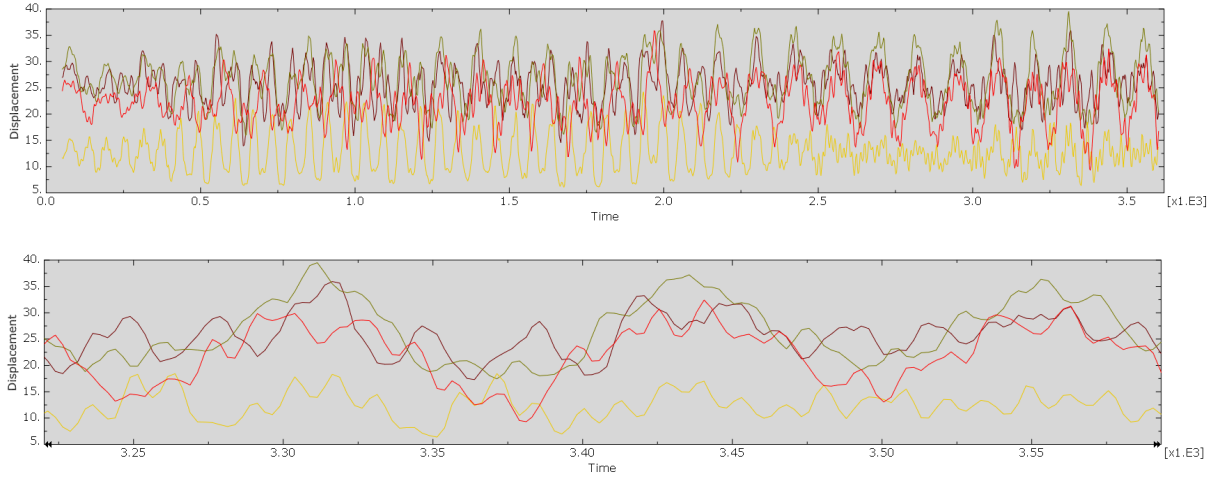


Figure 6-77: Displacement magnitude XY plane W255C seed 3

W255 seed 4

Point	Period [s]	Legend
1040_1	120	U:N: 1040_1
1082_1	100	U:N: 1082_1
1141_1	120	U:N: 1141_1
1196_1	120	U:N: 1196_1

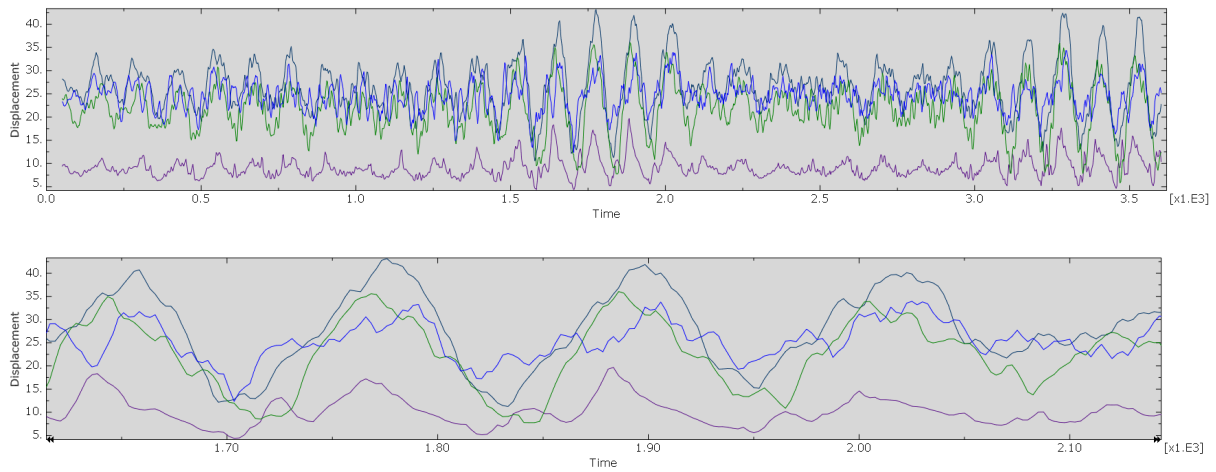


Figure 6-78: Displacement magnitude XY plane W255C seed 4

6.4.3 Stress Variation

W255C Seed 1

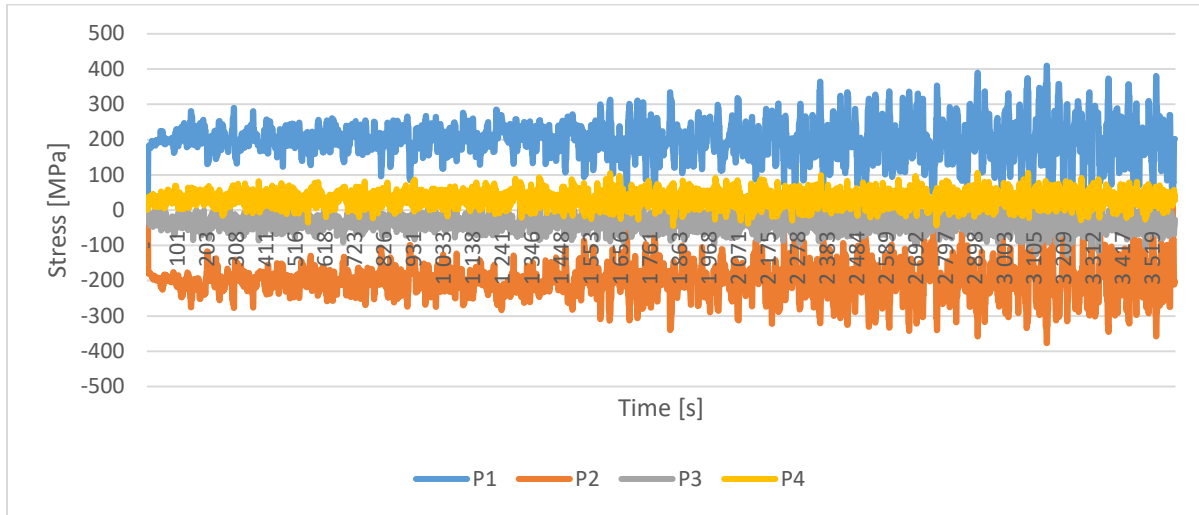


Figure 6-79: W255C seed 4 calculated stress at points

	P1	P2	P3	P4
Max Stress [MPa]	409,6	11,4	57,5	105,9
Min Stress [MPa]	-29,7	-376,9	-127,4	-45,2
$\Delta\sigma$ [MPa]	439,4	388,3	184,9	151,1

W280C Seed 4

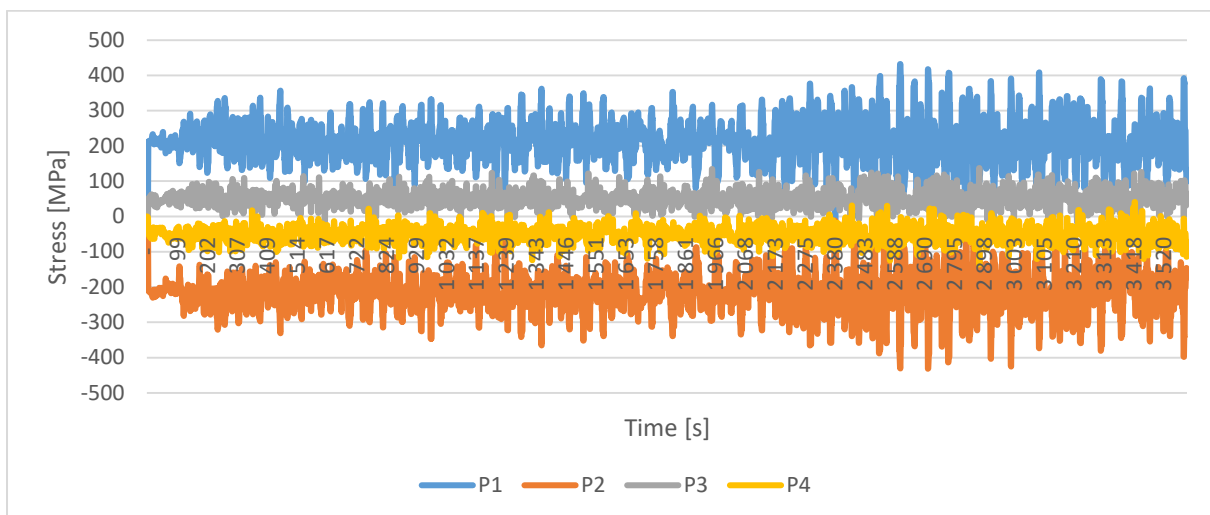


Figure 6-80: W280C seed 4 calculated stress at points

	P1	P2	P3	P4
Max Stress [MPa]	432,4	-11,5	137,5	40,6
Min Stress [MPa]	-32,1	-431,5	-23,8	-136,3
$\Delta\sigma$ [MPa]	464,5	420,0	161,3	176,9

Stresses at points during static wind loading from 255°

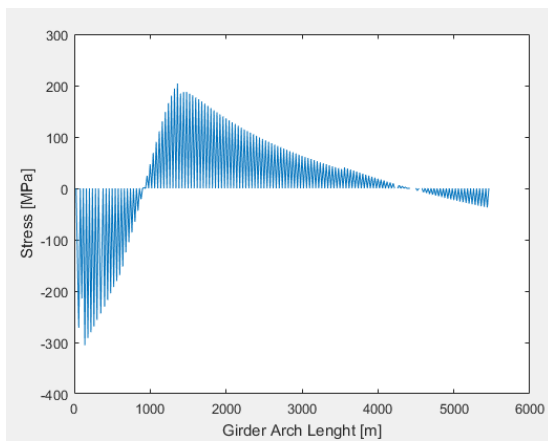


Figure 6-81: P1

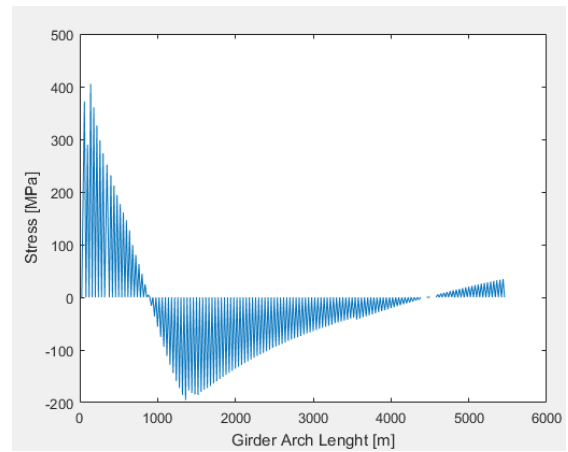


Figure 6-82: P2

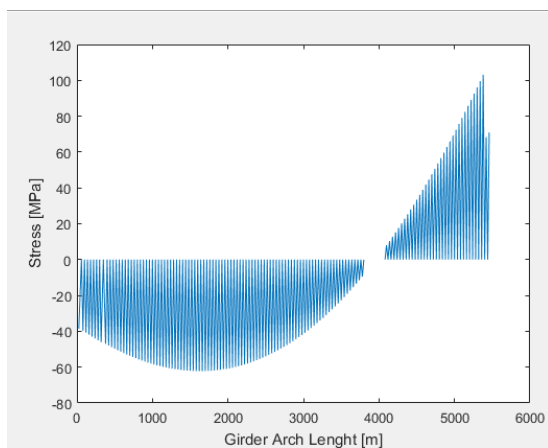


Figure 6-83: P3

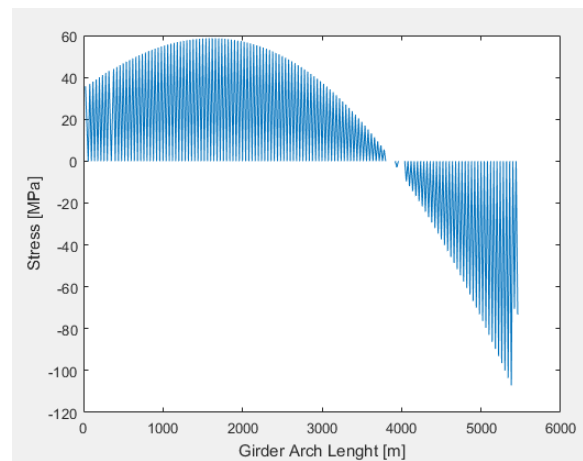


Figure 6-84: P4

Above figures provide an illustration of stress when the structures is loaded with wind which causes an unsymmetrical load on the girder due to the curvature in the horizontal plane. Large stresses are observed mainly in the cable stayed bridge and at the transition from cable stayed bridge to the high part of the floating bridge and at the northern fixed end.

7 Conclusion and Discussion

7.1 Result Considerations

Fundamental Modes

Modal analysis of the bridge exhibits very low eigenfrequencies. The first five modes have eigenperiods ranging from 120 s to 16,6 s. In which the mode shape mainly consist of the horizontal movement of the bridge girder.

Added Mass

Added mass effect is seen to have a great impact on the first modes, such for high frequency wave loading the first eigenperiods may be reduced by 20 s to 100 s, caused by the variable added mass effect. Obtaining almost zero effect of added mass is possible in sea waves with high frequencies.

Confirmation of Results

Comparing the achieved eigenfrequencies form the present model to priors results, very small difference in the first eigenmodes are observed. The structure's eigenfrequencies without the effect of added mass deviate the most during the first modes with 5-10 % compared to about 1-2% for the models including added mass.

Damping

Achieved effective damping ratio is likely unrealistic due to not including all aerodynamic damping terms, only the vertical and horizontal components are assessed. Hydrodynamic damping terms are neither included, however they are known to provide significant damping contribution.

Finite Element Formulation

Some aspects of the model are further analyzed, and a small indication of utilizing a finer mesh of the bridge girder is found, a deviation of eigenfrequencies are observed in mode 260 at frequencies above 4 Hz, linearly increasing with mode number. Utilization of the linear interpolated beam element based on Timoshenko beam theory showcase little variation with the cubic interpolated Euler-Bernoulli beam elements at modes ranging up to 250. At modes above 250 a large deviation of around 25% is observed. Caused by the shear flexible formulation of the Timoshenko beam elements, stiffness is decreased and higher frequencies are observed compared to Euler-Bernoulli beam element based model. Proper care should be taken when utilizing Timoshenko beams to properly define the shear stiffness.

Curvature of Bridge Impact

A simple run of several models with different horizontal bridge girder curvature is performed and implies that curvature changes between 4000 m to 8000 m does not produce large variation of the eigenfrequencies when compared to the current case utilizing a curvature of 5000 m.

Turbulence Simulations

Simulated turbulence has spectral densities in agreement with the target spectra. A deviation of the low frequencies at around 4 Hz deviate the most with the target spectra, which coincide with the first eigenfrequencies which dominate the bridge response. Proper care should be taken to apply a small sampling frequency when simulating the turbulence in order to achieve an even more compliance with the target wind spectra.

Response for Different Wind Directions

The first and second eigenmodes are observed to govern the response of the bridge girder, in which the main amplitude oscillation are noted to be 120 s and 60 s. Four different analysis sets are utilized with varying wind direction. Wind from cardinal direction 255° , a 15° yaw on the middle bridge perpendicular axis triggers a response mainly governed by the first mode. The yaw angle produces unsymmetrical load on the bridge girder coinciding with the first mode shape, large displacements with magnitude of up to 46,1 m are observed. Wind direction perpendicular to the middle bridge with a cardinal direction of 280° produces the highest acceleration of $2,23 \text{ m/s}^2$. However the most critical section forces in the girder is observed with wind from east, applied perpendicular to the middle bridge with a cardinal direction of 100° . Strong axis moment of 6600 MNm are obtained, 300 MNm greater than the second largest moment which is produced by the wind with cardinal direction 280° . Wind from east also produces the greatest axial force in the bridge girder, a compressive force with magnitude 302 MN, almost double the magnitude of the second largest axial force which is tensile with a magnitude of 170 MN. Compressive forces are however of greater concern in relation to buckling. Wind applied from north, cardinal direction 355° is not observed to provide neither displacement, acceleration nor moments in the bridge girder of importance. An axial compressive force is evidently seen in the south end of the bridge, however only with a magnitude below half the force observed while applying wind from cardinal direction 100° , east.

Accuracy of Model

Due to the low eigenfrequency, assessment of turbulent wind loads are of great importance. Wind is observed to primarily excite the first and second mode when assessing several points on the bridge girder. Compliance with prior results are witnessed in the first modes, but damping is assumed to be of greater magnitude than the one achieved. Such that the displacements, accelerations, forces and moments are most likely reduced if a more accurate damping is applied. Wind loads are largely simplified, by applying simplified buffeting load the vertical components of the wind load will be of much greater magnitude than what is observed in this thesis. Simplification of the drag and moment coefficients utilized to compute the load components are not modeled to change with the yaw angle. A reduction of load with increased yaw is therefore not achieved. Simplification of hydrodynamic effect should also be concerned, as only the constant added mass which is applicable for waves with low frequencies are utilized, excitation forces and several other restoring terms are not included. The coupled effect of wind and wave loads are thus not of concern in the analysis. At the end, the large eigenperiods deem long analysis periods, thru a 1 hour dynamic analysis only 30 periods of the first mode is allowed for. Allowing for additional periods may trigger a greater response due to resonance.

7.2 Recommendation for Further Studies

Thesis results show a compliance of the general construction of the bridge, resulting in accurate definitions of stiffness and mass matrices. Further work could build on the given recommendation of Abaqus model definition. To achieve an accurate dynamic analysis the effect of all turbulence buffeting components should be implemented, along with all excitation, frequency dependent added mass, restoring forces and damping effect attributed by the hydrodynamic interaction of the bridge.

Implementation of the above stated effects, utilization of Abaqus sub-structures should be done. Especially including the lift component variation due to deck rotation should be utilized such that vertical response of the bridge girder could be properly analyzed. Implementation of proper definition of the hydrodynamics effect should also be implemented, in which a coupled wind and wave analysis could be carried out.

Further study of applying a 4 dimensional turbulent wind field should be done. Horizontal curvature of the bridge combined with its large span that the Taylor's frozen turbulence hypothesis might be inadequate due to the large extent as the bridge is embedded in a 600 m x 4 500 m x 228 m prismatic volume. Most bridges exposed to great turbulent wind loads are mostly extending through two coordinates, in which the hypothesis have been implemented by several prior turbulent wind field loading of engineering structures.

8 Sources

- [1] Statens vegvesen, "Staten vegvesen," 1 March 2019. [Online]. Available: vegvesen.no/vegprosjekter/ferjefriE39/.
- [2] Statens vegvesen, 15 August 2018. [Online]. Available: https://www.vegvesen.no/_attachment/2399116/binary/1275152?fast_title=Presentasjon+ene+fra+fagseminaret+Ferjefri+E39+Verden+ser+til+Norge.pdf. [Accessed 2 February 2019].
- [3] Statens vegvesen, 19 October 2018. [Online]. Available: <https://vegnett.no/2018/10/soker-eu+midler-for-a-fa-mer-forskning-rundt-ferjefri-e39>. [Accessed 2 February 2019].
- [4] B. Wang, Large Floating Structures, Springer, 2018.
- [5] Our Tasmania, 2019. [Online]. Available: <http://www.visithobartaustralia.com.au/floating-bridge.html>. [Accessed 15 March 2019].
- [6] Structurae, 2019. [Online]. Available: <https://structurae.net/structures/lacey-v-murrow-floating-bridge>. [Accessed 15 March 2019].
- [7] Washington State Department of Transportation, 21 April 2016. [Online]. Available: <https://www.wsdot.wa.gov/Projects/SR520/BridgeAndLandings/default.htm>. [Accessed 15 January 2019].
- [8] Y. Jarlsett, 26 Decemeber 2018. [Online]. Available: https://snl.no/Bergs%C3%B8ysundet_bru. [Accessed 16 March 2019].
- [9] Y. Jarslett, 26 December 2018. [Online]. Available: <https://snl.no/Nordhordlandsbrua>. [Accessed 16 March 2019].
- [10] H. Svensson, Cable-Stayed Bridges - 40 Years of Experience, John Wiley & Sons, 2012.
- [11] Statens vegvesen, 9 Septemeber 2018. [Online]. Available: <https://www.vegvesen.no/Europaveg/e39stordos/nyhetsarkiv/billegare-a-bygge-bru>. [Accessed 12 June 2019].
- [12] S. Rao, Mechanical Vibration - 5th edition, Prentice Hall, 2010.
- [13] R. D. Cook, Conecpts and Application of Finite Element Analysis - 4 th Edition, John Wiley & Sons, 2002.
- [14] Dassault Systemes, *Simulia User Assistance 2017*, Dassault Systemes Simlia Corp., 2016.
- [15] V. Ryaben'kij, A theoretical Introduction to Numerical Analysis, CRC Press, 2006.
- [16] O. Faltinsen, Cambridge Ocean Technology Series: Sea Loads on Ships and Offshore Structures Series Number 1, Cambridge University Press, 1990.
- [17] R. Bronsart, "Hydrostatic and Stability Floating Structures," September 2015. [Online]. Available: https://www.lsb.uni-rostock.de/fileadmin/uni-rostock/Alle_MSF/LSB/Downloads/HydrostaticStabilityNutshell_221.pdf. [Accessed 27 May 2019].
- [18] Standard Norge, Eurocode 1: Actions on structures - Part 1-4: General actions - Wind actions, Standard Norge, 2010.
- [19] E. Strømmen, Theory of Bridge Aerodynamic - 2nd edition, Springer, 2010.
- [20] Y.-L. Xu, Wind Effects on Cable-Supported Bridges, John Wiley & Sons, 2013.
- [21] Statens vegvesen, Håndbok N400: Bruprosjektering, Statens vegvesen, 2015.
- [22] A.G.Davenport, "The application of stastical concepts to the wind loading of structures," *Tall Buildings*, pp. 3-45, 1967.

- [23] A. Davenport, «The spectrum of horizontal gustiness near the ground in high winds,» *Quarterly Journal of Royal Meteorological Society*, pp. 194-211, 1961.
- J. B. J. R. Etienne Cheynet, «Velocity Spectra and Coherence Estimas in the Marine Atmospheric Boundary Layer,» *Boundary-Layer Meteorology*, pp. 429-460, December 2018.
- [24] J. N. Kristensen L, «Lateral coherence in isotropic turbulence and in the natural wind,» *Boundary-Layer Meteorology*, pp. 353-373, 1979.
- [25] R. S. Ivar Langen, *Dynamisk Analyse av Konstruksjoner*, Tapir, 1979.
- [26] D. T. O. O. A. S. Norconsult, "K7 Bjørnafjorden End-anchored floating bridge," Staten vegvesen, 2017.
- [27] A. N. A. E. D. M.M. Hassan, «Determination of optimum post-tensioning cable forces of cbale-stayed bridges,» *Engineering Structures*, pp. 248-259, 2012.
- [28] G. M. T. K. N. Hajdin, «About the equivalent modulus of elasticity of cables of cable-stayed bridges,» *Architecture and Civil Engineering*, pp. 569-575, 1998.
- [29] G. Taylor, "Statistical Theory of Turbulence," *Proceddings of the Roytal Society A - Volume 151 - Issue 873*, p. 151, 2 September 1935.
- [30] H. Fuhr, «SBJ-01-C4-SVV-01-TN-006 Vind Bjørnafjorden - fase 5,» Statens vegvesen, 2018.
- [31] N. Isyumoc, "Announcement of the Alan G. Davenport Wind Loading Chain," in *Layer Wind Tunnel Laboratory*, 2011.
- [32]

Appendix A

Contents

Main input	A.1
MATLAB script bridge model	A.2
MATLAB script wind loads	A.3

A.1 Main Input

Filename: run_bjornafjorden.inp

```
** -----
***** FLOATING BRIDGE ACROSS BJØRNAFJORDEN
*HEADING
Three-dimensional bridge model
SI units (kg, m, s, N)
1-axis north, 2-axis west, 3-axis upwards
**CONSTRAINT CONTROLS, NO CHECKS
** - - - - - FILES TO BE INCLUDED - - - - -
*INCLUDE,INPUT=nodes.inp
*INCLUDE,INPUT=girder01.inp
*INCLUDE,INPUT=girder02.inp
*INCLUDE,INPUT=girder03.inp
*INCLUDE,INPUT=pontoon.inp
*INCLUDE,INPUT=PontoonWeight.inp
*INCLUDE,INPUT=girderreinforcemass.inp
*INCLUDE,INPUT=PYLON.inp
*INCLUDE,INPUT=PYLONtop.inp
*INCLUDE,INPUT=towerhor.inp
*INCLUDE,INPUT=connectors.inp
*INCLUDE,INPUT=cables.inp
*INCLUDE,INPUT=PONTOONSPRINGS.inp
*INCLUDE,INPUT=AerodynamicDamper.inp
** BOUNDARY CONDITIONS
*BOUNDARY
*INCLUDE,INPUT=boundaries.inp
** Thermal boundary
*Initial Conditions, type=TEMPERATURE
*INCLUDE,INPUT=CABLE_InitialTemp.inp
**
** -----
** STEP: StaticDeadLoad
**
*Step, name=StaticDeadLoad, nlgeom=YES, inc=1000000000
*Static
1, 1., 1e-015, 1
**
** Thermal boundary
*INCLUDE,INPUT=CABLE_PreStressTemp.inp
**
** LOADS
*CLOAD
*INCLUDE,INPUT=Pontoon_Buoyancy.inp
**
** Name: Load-1   Type: Gravity
*Dload
GravityElement, GRAV, 9.81, 0., 0., -1.
**
**
*** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, variable=ALL
```



```

**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, variable=ALL
*End Step
** -----
** STEP: Eigenanalysis
**
*Step, name=Eigenanalysis, nlgeom=NO, perturbation
*Frequency, eigensolver=AMS, normalization=mass
100, , 1, , , ,
**
** *Step, name=Eigenanalysis, nlgeom=NO, perturbation
**   *Frequency,      eigensolver=Lanczos,      sim,      acoustic      coupling=on,
normalization=displacement
** 100, , , , ,
**
** PONTOON ADDED MASS
**
*INCLUDE, INPUT=PontoonAddedMass.inp
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-1
**
*Output, field
*Node Output
U,
**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, variable=ALL
*End Step
** -----
**
** STEP: CheckDampingRatio
**
*Step, name=CheckDampingRatio, nlgeom=YES, perturbation
*Complex Frequency, friction damping=NO
100, , ,
**
** OUTPUT REQUESTS
**
** FIELD OUTPUT: F-Output-8
**
*Output, field, variable=ALL
*End Step
** -----
** STEP: StaticWind
**
*Step, name=StaticWind, nlgeom=YES, inc=1000000000
*Static
1, 1., 1e-015, 1
**

```

```

**
** LOADS
**INCLUDE,INPUT=StaticWindLoad.inp
**
**
**
** OUTPUT REQUESTS
**
**Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-1
**
**Output, field, variable=ALL
**
** HISTORY OUTPUT: H-Output-1
**
**Output, history, variable=ALL
**End Step
** -----
**
** STEP: DynamicWind
**
**Step, name=DynamicWind, nlgeom=YES, inc=10000000
DynamicWind
**Dynamic
**INCLUDE,INPUT=dynamicstep.inp
**
** Dynamic Loads
**INCLUDE,INPUT=WindLoadGirder.inp
**INCLUDE,INPUT=WindLoadCable.inp
**INCLUDE,INPUT=WindLoadTower.inp
**INCLUDE,INPUT=WindLoadPontoon.inp
** LOADS
**CLOAD,OP=NEW
**INCLUDE,INPUT=Pontoon_Buoyancy.inp
**Dload,OP=NEW
GravityElement, GRAV, 9.81, 0., 0., -1.
**
** OUTPUT REQUESTS
**
**Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-5
**
**Output, field, variable=ALL
**
** HISTORY OUTPUT: H-Output-4
**
**Output, history, variable=ALL, frequency=9
**End Step

```

A.2 MATLAB script bridge model

```
clear
addpath('functionsForBjornafjorden')
addpath('functionsForWindSim4D')

%% This is the main file of the Bjørnafjorden floating bridge abaqus input
creator

% *** UNITS ***
% Length: metres
% Mass: kg
% Gravitation: m/s^2
% Force: Newton
% Pressure: Pa (N/m^2)
% Degrees: Kelvin

format long

%% Scaling factors
cable_mult=1;

% Draft
draft=5;%
pontoon_mult1=draft;
pontoon_mult24=draft;
pontoon_mult511=draft;
pontoon_mult1246=draft;

%% Density of air
rho=1.2;%kg/m3

%% Damping

FirstFreq=1/120; %
SecondFreq=0.63175;
CriticalDamping =0.005;% 0.005; % 0.5 percent
% Converting frequencies to rad/s
FirstFreq=FirstFreq*2*pi;
SecondFreq=SecondFreq*2*pi;
AlfaDamping =
CriticalDamping*2*(FirstFreq*SecondFreq)/(FirstFreq+SecondFreq);
BetaDamping = CriticalDamping*2/(FirstFreq+SecondFreq);
RedFact=0.33;
DPC=1; % Tuning of aerodynamic dampers

%% Defining horizontal chareacteristics of bridge girder

R=6000; %m; Horizontal curvature of the bridge
nodedL1 = 20; %m; lenght between cables in cable stayed bridge
nodedL2 = 20; %m; lenght between nodeplacement in the floating
bridge part
RotationFromNorth = 10; %The degress the bridge line is rotated from north

%% Defining vertical characteristics of bridge girder
%sectionpoints=[x1,x2,x1,x2,x1,x2,x1,x2] The points which stress are
%calculated in the girder
```

```

girdersectionpoints=[13.8,0,-13.8,0,0,1.93,0,-1.57];
cg=-1.93; %Height from top of girder to center of gravity
CablesConnectionHeight= -0.4389; %Height from top of girder to the cables
connection to the girder

%% Defining the cable tower
CableTower_Girder_Offsett = 6.31; % The offsett of the centerline of girder
relative to the top of tower.
%% Defining the initial draft of the Pontoons
PONTOONDRAFT=5.0;
%% Running files
% Creating the list of lenght of cable stayed part
L1 = [0,25,60]; %%Profile number 38600 is placed at 0
for i = 4:1000
    L1(i)=L1(i-1)+nodedL1;
    if L1(i)>300
        break
    end
end
L1=[L1,350];
L1=[L1,380];
for i =size(L1,2)+1:1000
    L1(i)=L1(i-1)+20;
    if L1(i)+20 >=860
        break
    end
end
%% Creating the list of lenght of floated part
L2=[860];
for i = 2:1000
    L2(i)=L2(i-1)+nodedL2;
    if L2(i)+25>5360
        break
    end
end
for i=1:7 %Last 7 points, the distance between pontoon and shore is 140
metres.
L2=[L2,L2(size(L2,2))+20];
end
clearvars k
%% Combining L1 and L2
L=[L1,L2];
clearvars k s1 s2 L1 L2 k i;
%% Creating x-y coordinates
X_g=[];
Y_g=[];
for i = 1:size(L,2)
    alfa=L(i)/R; %Radians
    X_g(i)=R*sin(alfa); % could also use equation X_g(i)=L_*cos(alfa/2);
    Y_g(i)=R*(1-cos(alfa));
end
clearvars alfa i k L_ ;
%% Creating x-y coordinate for the orioogo of curvature
X_g_Orientation=0;
Y_g_Orientation=R;
%% Creating z-coordinaates
% The list of z coordinates coincide with the L, thus also X,Y coords.

%The center of the girder is places on the center of area/mass. Since the
%section is mostly made up of steel, the middel of the cross section area
%conincide with the center of gravity.

```

```
%Dividing the girder lenght into division based on profile
s=[0,25,800,1435,1685,1935,5340,5580];
```

```
syms a b c x y r
```

```
% s(1)-s(2)Funcion from -100(0) to 25(125)
r1=5000;
in1=56.065;
off1=r1-in1;
f1=symfun(sqrt(r1^2-x^2)-off1,x);
```

```
% s(2)-s(3) Function from 25 (0) to 800(775)
in2=double(f1(125));
a2=0.005;
f2=symfun(in2-a2*x,x);
```

```
% s(3) - s(4) Function from 800(0) to 1435(635)
r3=14500;
in3=double(f2(775));
inx3=72.5;
off3=r3-in3;
f3=symfun(sqrt(r3^2-(x+inx3)^2)-off3,x);
```

```
% s(4) - s(5) Function from 1435(0) to 1685(250)
in4 = f3(635);
a4=0.0497;
f4=symfun(in4-a4*x,x);
```

```
% s(5) - s(6) Function from 1685(0) to 1935(250)
r5=6600; %should be 5000
in5=double(f4(250));
off5=r5-in5;
f5=symfun(sqrt(r5^2-(x)^2)-off5,x);
```

```
% s(6) - s(7) Function from 1935 to 5340
%constant height og 16.16
```

```
% s(7) - s(8) Function from 5340(0) to 5380(40) m
r7=6456.;
in7=16.16;
off7=r7-in7;
f7=symfun(sqrt(r7^2-(x)^2)-off7,x);
counter=size(L);counter=counter(2);
for i = 1:counter
    if L(i)<=25
        h=double(f1(L(i)-100));
    elseif L(i)>25 && L(i)<=800
        h=double(f2(L(i)-25));
    elseif L(i)>800 && L(i)<=1435
        h=double(f3(L(i)-800));
    elseif L(i)>1435 && L(i)<=1685
        h=double(f4(L(i)-1435));
    elseif L(i)>1685 && L(i)<=1935
        ansy=double(f5(L(i)-1685-250));
        h=double(f4(250))-ansy+16.16;
    elseif L(i)>1935 && L(i)<=5340
        h=16.16;
```

```

elseif L(i)>5340
    h=double(f7(L(i)-5340));
end
Z_g(i)=h;
end
clearvars x a a2 a4 and b c count counter f1 f2 f3 f4 f5 f7 h i in1 in2 in3
in4 in5 in7 inx3 off1 off3 off5 off7 r r1 r3 r5 r7 y
%% Z is currently the height to the top of girder, to recalculate to the
%center of gravity:
for i = 1:size(Z_g,2)
    Z_gcg(i)=Z_g(i)+cg;
end
clearvars counter i
%% Creating the nodes to attach the cables to the girder
% at following points the cables are attached
numbcables1=14; %Number of cables to the south of the tower
numbcables2=22; %Number of cables to the north of the tower
L_c=[60.];
for i =2:numbcables1
    L_c(i)=L_c(i-1)+20;
end
L_c(15)=L_c(14)+60;
for i=numbcables1+2:numbcables1+numbcables2
    L_c(i)=L_c(i-1)+20;
end
%the distance from center girder to cable connection is +-15,5 m
%thus r_inner=R-15,5 and r_outer=R+15,5
r_inner=R-13.8;
r_outer=R+13.8;
%creating x-y coordinates
X_gci=[];
Y_gci=[];
X_gco=[];
Y_gco=[];
for i = 1:2
    if i == 1
        r=r_outer;
    else
        r=r_inner;
    end

    for j =1:size(L_c,2)
        alfa=L_c(j)/r;
        ansy=r*(1-cos(alfa));
        ansx=r*sin(alfa);
        if i==1
            Y_gco(j)=double(ansy)-15.5*cos(alfa);
            X_gco(j)=double(ansx)+15.5*sin(alfa);
        else
            Y_gci(j)=double(ansy)+15.5*cos(alfa);
            X_gci(j)=double(ansx)-15.5*sin(alfa);
        end
    end
end

clearvars r i j ansx ansy alfa
end
Z_gc=[];
%Recalulating the height of the cable connectors to the girder. The current
%height is the same as the height of the girder.

```

```

for i=1:size(L_c,2)
    for j=1:size(L,2)
        if abs(abs(L_c(i))-abs(L(j)))<1
            Z_gc(i)=Z_g(j)+CablesConnectionHeight;
        end
    end
end
clearvars k alfa_inner alfa_outer L_inner L_outer r_inner r_outer
numbcables ans i

%% Creating the nodes for the floating pontoons
L_p=[860.]; %the first pontons is placed at L = 860 metres from 38600
for i = 2:46
    L_p(i)=L_p(i-1)+100.;
end
%Creating x-y coordinates
X_p=[];
Y_p=[];
k=size(L_p);k=k(2);
for i = 1:k
    alfa=L_p(i)/R; %Radians
    L_=2*R*sin(alfa/2);
    X_p(i)=R*sin(alfa); % could also use equation X_g(i)=L_*cos(alfa/2);
    Y_p(i)=sqrt((L_)^2-X_p(i)^2);
end
%Define 4 nodes for the pontoons:
%at the start of the beam - the connection between girder and pontoon
%at the end of the beam between girder and pontoon
%at the center of buoyancy
%at the center of gravity
Z_girderpontoon_conn=[];
for i=1:46
    for j=1:size(Z_g,2)
        if L(j)==L_p(i)
            Z_girderpontoon_conn=[Z_girderpontoon_conn,Z_g(j)-0.350-
1.715-1.435];
        end
    end
end
Z_p=4.; %Where beam connects to pontoon
Z_pcb= 0.%;Z of center if buoyancy
Z_pcg= -1.%;Z of center of gravity
%% Creating the cable tower
%starting from bottom in basic 2d design
%w=local coordinate system
W_t=[];
Z_t=[];
% tower legs
% from 5 to 48 m in height and from +-12,5 to +-30
% divided in to 4 metres elements
dh=5;
w1=12.5;
w2=30;
w3=0;
w4=0;
z1=5;
z2=48;
z3=169;
z4=228;
dz1=z2-z1;
n1=round(dz1/dh);

```

```

wslope=(w2-w1)/dz1;
for i=1:n1
    w=w1+wslope*(i-1)*dh;
    z=z1+dh*(i-1);
    if i == n1
        z=z2;
        w=w2;
    end
    W_t(i)=w;
    Z_t(i)=z;
end
dz2=z3-z2;
n2=round(dz2/dh);
wslope=(w3-w2)/dz2;
for i=1:n2-1
    w=w2+wslope*(i)*dh;
    z=z2+dh*(i);
    if i == n2
        z=z3;
        w=w3;
    end
    W_t(i+n1)=w;
    Z_t(i+n1)=z;
end
% Adding the negative side
for i = 1:n1+n2-1
    W_t(n1+n2-1+(i))=-W_t(i);
    Z_t(n1+n2-1+(i))=Z_t(i);
end
% Adding the nodes up to the cable connectors
W_t=[W_t,0];
Z_t=[Z_t,169];
dz3=59-4.;
dh=2.5;
n=dz3/dh;
for i = 1:n+1;
    w=0;
    z=173+dh*(i-1);
    W_t(i-1+(n1+n2)*2)=w;
    Z_t(i-1+(n1+n2)*2)=z;
end
% Adding the vertical beam
dw=2*w2;
wslope=5;
n=round(dw/wslope);
for i = 1:n-1;
    w=w2-wslope*(i);
    z=z2;
    W_t=[W_t,w];
    Z_t=[Z_t,z];
end
%Moving the tower
for i = 1:size(W_t,2);
    W_t(i)=-W_t(i)+CableTower_Girder_Offsett;
end
%Rotating the tower to align with the bridge
theta=4*2*pi/360;
Q=[cos(theta), sin(theta);-sin(theta), cos(theta)];
X_t=[];
Y_t=[];

```



```

for i=1:size(L,2);
    if L(i)==350.;
        j=i;
    end
end
for i=1:size(W_t,2)
    X_t(i)=Q(2)*W_t(i)+X_g(j);
    Y_t(i)=Q(4)*W_t(i)+Y_g(j);
end
clearvars theta Q dh w1 w2 w3 w4 z1 z2 z3 z4 dz1 n1 wslope dz2 n2 dz3 dh n
%% Creating the connection for cables in the tower

L_tc=350.;
r_inner=R-3.;
r_outer=R+3.;

for i = 1:2
    if i == 1
        r=r_inner;

    else
        r=r_outer;
        alfa=L_tc/r;

    end
    alfa=L_tc/r;
    addx=-CableTower_Girder_Offsett*sin(alfa);
    addy=CableTower_Girder_Offsett*cos(alfa);

    ansy=addy+r*(1-cos(alfa));
    ansx=addx+r*sin(alfa);
    if i==1
        Y_tci=ansy+3.*cos(alfa);
        X_tci=ansx-3.*sin(alfa);
    else
        Y_tco=ansy-3.*cos(alfa);
        X_tco=ansx+3.*sin(alfa);
    end
end

Z_tci=[];
Z_tco=[];

for i =1:22
    Z_tci(i)=169.+4+2.5*(i-1);
    Z_tco(i)=Z_tci(i);
end
clearvars alfa ansy ansx r L_tc L_ addx addy
%% Creating the anchor chamber
%Placing the anchor at 38600

dy=16.; %The distance form center of girder to the cable anchor
alfa_xz = 0.1; %the angle of cables in z-x plane
alfa_xyi = atan((16-16.5/2)/350);%the angle of orentation in the xy plane
of the cables and the anchor box of the inner cables
alfa_xyo = atan((16+16.5/2)/350);%the angle of orentation in the xy plane
of the cables and the anchor box of the outer cables
dz=2.5; %distance from each cable in z direction
X_anco=[];

```

```

X_anci=[];
Y_anco=[];
Y_anci=[];
Z_anco=[];
Z_anci=[];

for i = 1:8 %8 cables on each side is connected to the anchor
X_anco(i)=-40+5*(i-1); % X_anco(i)=10+dz*sin((pi/2)-alfa_xz)*(i-1);
X_anci(i)=X_anco(i);
Y_anco(i)=-13-dz*sin((pi/2)-alfa_xz)*sin(alfa_xyo)*(i-1);
Y_anci(i)=17-dz*sin((pi/2)-alfa_xz)*sin(alfa_xyi)*(i-1);
Z_anco(i)=50-dz*cos((pi/2)-alfa_xz)*(i-1);
Z_anci(i)=50-dz*cos((pi/2)-alfa_xz)*(i-1);
end
clearvars dy alfa_xz alfa_xyi alfa_xyo dz i
% Rotating the coordinates system to set x-axis along north; y-axis
along west and z-axis upwards.
%% Rotating the local x,y,z coordinates to the global coordinate system
where x is along north, y along east and z positive upwards.
%Origo is places on profile number 38600 at mean sea height with x-coord
%along north and y along east

X={X_anci,X_anco,X_g,X_p,X_tci,X_tco,X_t,X_gci,X_gco,X_g_Orientation};
Y={Y_anci,Y_anco,Y_g,Y_p,Y_tci,Y_tco,Y_t,Y_gci,Y_gco,Y_g_Orientation};

theta=-0.550; %the angle the system is to be rotated to be able to have be
along the north axis
theta=theta-(RotationFromNorth/360)*2*pi; %the bridge is rotatet 10 degrees
(from north to east)

Q=[cos(theta), sin(theta);
-sin(theta), cos(theta)];

for i=1:size(X,2)
for j = 1:size(X{i},2)
xy=[X{i}(j);Y{i}(j)];
xyd=xy.*Q;
X{i}(j)=xyd(1)+xyd(2);
Y{i}(j)=xyd(3)+xyd(4);
end
end

X_anci= X{1};
X_anco= X{2};
X_g= X{3};
X_p= X{4};
X_tci= X{5};
X_tco= X{6};
X_t= X{7};
X_gci= X{8};
X_gco= X{9};
X_g_Orientation =X{10};

Y_anci= Y{1};
Y_anco= Y{2};
Y_g= Y{3};
Y_p= Y{4};

```

```

Y_tci= Y{5};
Y_tco= Y{6};
Y_t= Y{7};
Y_gci= Y{8};
Y_gco= Y{9};
Y_g_Orientation= Y{10};
clearvars X Y j
%% Create nodes
%% Matching nodes for elements
%%To easily find nodes for later use, each set is indexed by vector
%node_i
%%Also, the node matrix each row have 5 values:
% 1 ID (i.e. node number.
% 2 X
% 3 Y
% 4 Z
% 5 Placement info. i.e. the arclength it is placed or the pontoon number
% etc.

%Girder (1000)
node_Girder= [];
n= [];

for i = 1:size(X_g,2)
    n(1)=1000+(i); %numbering of nodes along girder start at 1000
    n(2)=X_g(i); %x-coord
    n(3)=Y_g(i); %y-coord
    n(4)=Z_gcg(i); %Z-coord
    n(5)=L(i);
    node_Girder=[node_Girder;n];
end
%Pontoons at connection (2000)
node_Pontoon_conn=[];
for i = 1:size(X_p,2)
    n(1)=2000+(i);
    n(2)=X_p(i); %x-coord
    n(3)=Y_p(i); %y-coord
    n(4)=Z_p; %Z-coord
    n(5)=i; %Pontoon numbering is from 1-46
    node_Pontoon_conn=[node_Pontoon_conn;n];
end
% Pontoon cnnction to girder nodes (12000)
node_GirderPontoon_conn=[];
for i = 1:size(X_p,2)
    n(1)=12000+i; %
    n(2)=X_p(i); %x-coord
    n(3)=Y_p(i); %y-coord
    n(4)=Z_girderpontoon_conn(i); %Z-coord
    n(5)=i;
    node_GirderPontoon_conn=[node_GirderPontoon_conn;n];
end
% Nodes at coloumn between girder connection and pontoon (13000)
node_GirderColoumn=[];
for i = 1:46
    n = round((node_GirderPontoon_conn(i,4)-node_Pontoon_conn(i,4))/5);
    for j=1:n-1
        n(1)=13000+(i*10)+j;
        n(2)=node_GirderPontoon_conn(i,2);
        n(3)=node_GirderPontoon_conn(i,3);
        n(4)=node_Pontoon_conn(i,4)+j*5;
        n(5)=i;
    end
end

```

```

        node_GirderColoumn=[node_GirderColoumn;n];
    end
end
%Pontoons at center of buoyancy (3000)
node_Pontoon_CB=[];
for i = 1:size(X_p,2)
    n(1)=3000+i;
    n(2)=X_p(i); %x-coord
    n(3)=Y_p(i); %y-coord
    n(4)=Z_pcb; %Z-coord
    n(5)=i;
    node_Pontoon_CB=[node_Pontoon_CB;n];
end
%Pontoons at center of gravity (4000)
node_Pontoon_CG=[];

for i = 1:size(X_p,2)
    n(1)=4000+i; %
    n(2)=X_p(i); %x-coord
    n(3)=Y_p(i); %y-coord
    n(4)=Z_pcg; %Z-coord
    n(5)=i;
    node_Pontoon_CG=[node_Pontoon_CG;n];
end

% Cable connections
% Each cable has its own number, corresponding ot the place it is
% connected
% Cable numbering in coloumn 5 (n(5))
%
%
%           Inner (W)      Outer   (E)
% Anchorage      101-108      301-308
% South of tower  109-122      309-322
% North of tower  201-222      401-422
% Connector tower 501-522      601-622
%Cable connections to girder at inner (west) (5000)

node_CableGirder_W=[];

for i = 1:36
    n(1)=5000+i;
    n(2)=X_gci(i); %x-coord
    n(3)=Y_gci(i); %y-coord
    n(4)=Z_gc(i); %Z-coord
    if i<=15
        n(5)= 108+(i-1); %Inner south of tower
    else
        n(5) = 201+(i-1); %Inner north of tower
    end
    node_CableGirder_W=[node_CableGirder_W;n];
end

%cable connections to girder at outer (east) (6000)

node_CableGirder_E=[];

for i = 1:36
    n(1)=6000+(i);
    n(2)=X_gco(i); %x-coord
    n(3)=Y_gco(i); %y-coord

```

```

n(4)=Z_gc(i); %Z-coord
if i<=15
    n(5)=307+(i-1); %Outer south of tower
else
    n(5)=401+(i-1); %Outer north of tower
end
node_CableGirder_E=[node_CableGirder_E;n];
end
%cable connections to tower at inner (west) (7000)
node_CableTower_W=[];
for i = 1:22
    n(1)=7000+i;
    n(2)=X_tci; %x-coord
    n(3)=Y_tci; %y-coord
    n(4)=Z_tci(i); %Z-coord
    n(5)=500+i;
    node_CableTower_W=[node_CableTower_W;n];
end
%cable connections to tower at outer (east) (8000)
node_CableTower_E=[];
for i = 1:22
    n(1)=8000+i;
    n(2)=X_tco; %x-coord
    n(3)=Y_tco; %y-coord
    n(4)=Z_tco(i); %Z-coord
    n(5)=600+i;
    node_CableTower_E=[node_CableTower_E;n];
end
%Anchor to cables nodes at inner (west) (9000)
node_CableAnchor_W=[];
for i = 1:8
    n(1)=9000+i;
    n(2)=X_anci(i); %x-coord
    n(3)=Y_anci(i); %y-coord
    n(4)=Z_anci(i); %Z-coord
    n(5)=101+(i-1);
    node_CableAnchor_W=[node_CableAnchor_W;n];
end
%Anchor to cables nodes at outer (east) (10000)
node_CableAnchor_E=[];
for i = 1:8
    n(1)=10000+i;
    n(2)=X_anco(i); %x-coord
    n(3)=Y_anco(i); %y-coord
    n(4)=Z_anco(i); %Z-coord
    n(5)=301+(i-1);
    node_CableAnchor_E=[node_CableAnchor_E;n];
end
%Tower nodes (11000)
node_Tower=[];
for i=1:size(Z_t,2)
    n(1)=11000+i;
    n(2)=X_t(i);
    n(3)=Y_t(i);
    n(4)=Z_t(i);
    if n(4)<=169
        n(5)=0;
    elseif n(4)==173
        start=i-1;
        n(5)=(i-start);
    elseif n(4)>173

```

```

        n(5)=(i-start);
    end
    if n(4)==228
        n(5)=0;
    end
    node_Tower=[node_Tower;n];
end
%% Creating nodes for the heave spring to be offsett
node_Pontoon_Spring_Off=[];

for i = 1:46
    e1=105000+i;
    e2=node_Pontoon_CB(i,2);
    e3=node_Pontoon_CB(i,3);
    e4=node_Pontoon_CB(i,4)+5;
    e5=node_Pontoon_CB(i,1);
    node_Pontoon_Spring_Off=[node_Pontoon_Spring_Off;e1,e2,e3,e4,e5];
end

%% Create Elements
%% Creating girder elements
e=[];
element_Girder=[];

for i=1:size(node_Girder,1)-1
    e(1)=1000+i;
    e(2)=node_Girder(i,1);
    e(3)=node_Girder(i+1,1);
    e(4)=node_Girder(i,5);
    element_Girder=[element_Girder;e];
end
%% Creating Pontoon elements
% Creating the element between girder and pontoon connection point
element_Pontoon=[];
for i=1:46
    n = round((node_GirderPontoon_conn(i,4)-node_Pontoon_conn(i,4))/5);
    for j = 1:n
        e(1)=2000+(10*i+j);
        if j==1
            e(2)=node_Pontoon_conn(i,1);

            for k = 1:size(node_GirderColoumn,1)
                if node_GirderColoumn(k,1) == 13000+(10*i)+1
                    e(3)=node_GirderColoumn(k,1);
                end
            end
        elseif j>1 && j<n

            for k = 1:size(node_GirderColoumn,1)
                if node_GirderColoumn(k,1) == 13000+(10*i)+j
                    e(2)=node_GirderColoumn(k,1);
                    e(3)=node_GirderColoumn(k-1,1);
                end
            end
        elseif j==n
            count=0;
            for k=1:size(node_GirderColoumn,1)
                if node_GirderColoumn(k,5) < i+1
                    count=count+1;
                end
            end
        end
    end
end

```

```

                end
            end
            e(2)=node_GirderPontoon_conn(i,1);
            e(3)=node_GirderColoumn(count,1);
        end
        e(4)=i;
        element_Pontoon=[element_Pontoon;e];
    end
end
% Creating the element between pontoon connection point and center of
% gravity
conn_Pontoon_CG=[];
for i=1:46
    e(1)=3000+i;
    e(2)=node_Pontoon_conn(i,1);
    e(3)=node_Pontoon_CG(i,1);
    e(4)=i;
    conn_Pontoon_CG=[conn_Pontoon_CG;e];
end
% Creating the element between pontoon center of gravity point and center
of
% buoyancy
conn_Pontoon_CB=[];
for i=1:46
    e(1)=4000+i;
    e(2)=node_Pontoon_conn(i,1);
    e(3)=node_Pontoon_CB(i,1);
    e(4)=i;
    conn_Pontoon_CB=[conn_Pontoon_CB;e];
end
%% Cable connection elements to girder
%
% Inner (W)    Outer (E)
% Anchorage   5101-107    5301-307
% South of tower 5108-122    5308-322
% North of tower 5201-222    5401-422
% Connector tower 5501-522    5601-622
conn_CableGirder_W=[];
%Connection to girder at inner (west)
for j =1:size(node_Girder,1)
    if node_Girder(j,5)==60
        start=j; %Where the first connector is connected
to the girder
    end
    end %Calculating strat : first node in the girder where the cables are
connected
Girder_conn_cable=[];
for i = 1:14
    Girder_conn_cable(i)=node_Girder(start+(i-1),1);
end
for i = 15:36
    Girder_conn_cable(i)=node_Girder(start+i,1);
end
for i=1:36
    e(1)=5000+i;
    e(2)=node_CableGirder_W(i,1);
    e(3)=Girder_conn_cable(i);
    e(4)=node_CableGirder_W(i,5);
    conn_CableGirder_W=[conn_CableGirder_W;e];
end
conn_CableGirder_E=[];
%Connection to girder at outer (east)

```

```

for i=1:36
    e(1)=6000+i;
    e(2)=node_CableGirder_E(i,1);
    e(3)=Girder_conn_cable(i);
    e(4)=node_CableGirder_E(i,5);
    conn_CableGirder_E=[conn_CableGirder_E;e];
end
clearvars start
%% Cable connection elements to tower
%west
conn_CableTower_W=[];
for i=1:22
    e(1)=7000+i;
    e(2)=node_CableTower_W(i,1);
    for j = 1:size(node_Tower)
        if node_Tower(j,5) == node_CableTower_W(i,5) -
(node_CableTower_W(1,5)-1)
            e(3)=node_Tower(j,1);
        end
    end
    e(4)=i;
    conn_CableTower_W=[conn_CableTower_W;e];
end
%east
conn_CableTower_E=[];
for i=1:22
    e(1)=8000+i;
    e(2)=node_CableTower_E(i,1);
    for j = 1:size(node_Tower)
        if node_Tower(j,5) == node_CableTower_E(i,5) -
(node_CableTower_E(1,5)-1)
            e(3)=node_Tower(j,1);
        end
    end
    e(4)=i;
    conn_CableTower_E=[conn_CableTower_E;e];
end

%% Tower elements
% LEGS 1:9 and 33:41 #9000 and #9100
element_Tower=[];
e=[];
for i=1:8
    e(1)=9000+i;
    e(2)=node_Tower(i);
    e(3)=node_Tower(i+1);
    e(4)=i; % REFERENCE TO THE CROSS SECTIONAL
PROPERTIES
    element_Tower=[element_Tower;e];
end
for i=1:8
    e(1)=9100+i;
    e(2)=node_Tower(33+(i-1));
    e(3)=node_Tower(33+i);
    e(4)=i;
    element_Tower=[element_Tower;e];
end
% PYLON 10:32 and 42:64 #9200 and #9300
for i=1:size(node_Tower)
    if node_Tower(i,4)==169;
        pylon_int=node_Tower(i,1);
    end
end

```



```

end
end
for i = 1:23
    e(1)=9200+i;
    e(2)=node_Tower(9+(i-1));
    e(3)=node_Tower(9+i);
    e(4)=9+(i-1);
    element_Tower=[element_Tower;e];
    if node_Tower(9+i,4)==163
        e(1)=9200+i+1;
        e(2)=node_Tower(9+i);
        e(3)=pylon_int;
        e(4)=9+i;
        element_Tower=[element_Tower;e];
    end
end
for i= 1:23
    e(1)=9300+i;
    e(2)=node_Tower(41+(i-1));
    e(3)=node_Tower(41+i);
    e(4)=9+(i-1);
    element_Tower=[element_Tower;e];
    if node_Tower(9+i,4)==163
        e(1)=9300+i+1;
        e(2)=node_Tower(41+i);
        e(3)=pylon_int;
        e(4)=9+i;
        element_Tower=[element_Tower;e];
    end
end
end
% PYLON 65:88 #9400
for i =1:23
    e(1)=9400+i;
    e(2)=node_Tower(65+(i-1));
    e(3)=node_Tower(65+i);
    e(4)=32+i;
    element_Tower=[element_Tower;e];
end
% HORIZONTAL BEAM #9500
e(1)=9500+1;
e(2)=node_Tower(9);
e(3)=node_Tower(89);
e(4)=0;
element_Tower=[element_Tower;e];
for i=2:11;
    e(1)=9501+i;
    e(2)=node_Tower(88+(i-1));
    e(3)=node_Tower(88+i);
    e(4)=0;
    element_Tower=[element_Tower;e];
end
e(1)=e(1)+1;
e(2)=node_Tower(99);
e(3)=node_Tower(41);
e(4)=0;
element_Tower=[element_Tower;e];
%% *** Cables ***
%Cable numbering
%
%South of tower      Inner      Outer
%North of tower     3001-3022  4001-4022

```

```

element_Cable=[];
% *** WEST ***
%South side west (inner) anchor to tower number 12 000

for i=1:8
    e(1)=12000+i;
    e(2)=node_CableAnchor_W(i,1);
    e(3)=node_CableTower_W(size(node_CableTower_W,1)-(i-1),1);
    e(4)=1000+i;
    element_Cable=[element_Cable;e];
end
%South side west(inner) girder to tower number 13000
for i=1:14
    e(1)=13000+i;
    e(2)=node_CableGirder_W(i,1);
    e(3)=node_CableTower_W(16-i,1);
    e(4)=1008+i;
    element_Cable=[element_Cable;e];
end
%North side inner (west) girder to tower number 14000
for i=1:22
    e(1)=14000+i;
    e(2)=node_CableGirder_W(14+i,1);
    e(3)=node_CableTower_W(i,1);
    e(4)=3000+i;
    element_Cable=[element_Cable;e];
end
% *** EAST ***
%South side east (outer) anchor to tower number 15000
for i=1:8
    e(1)=15000+i;
    e(2)=node_CableAnchor_E(i,1);
    e(3)=node_CableTower_E(size(node_CableTower_E,1)-(i-1),1);
    e(4)=2000+i;
    element_Cable=[element_Cable;e];
end
%South side east (outer) girder to tower number 16000
for i=1:14
    e(1)=16000+i;
    e(2)=node_CableGirder_E(i,1);
    e(3)=node_CableTower_E(size(node_CableTower_E,1)-7-(i-1),1);
    e(4)=2008+i;
    element_Cable=[element_Cable;e];
end

%North side east (outer) girder to tower number 17000
for i=1:22
    e(1)=17000+i;
    e(2)=node_CableGirder_E(14+i,1);
    e(3)=node_CableTower_E(i,1);
    e(4)=4000+i;
    element_Cable=[element_Cable;e];
end
% Creating the element between pontoon connection and girder
conn_Pontoon_Girder=[];
for i=1:46
    e(1)=18000+i;
    e(2)=node_GirderPontoon_conn(i,1);
    for j = 1:size(node_Girder,1);
        if node_Girder(j,5)-860==100*(i-1)
            e(3)=node_Girder(j,1);

```

```

        end
    end
    e(4)=i;
    conn_Pontoon_Girder=[conn_Pontoon_Girder;e];
end
clearvars e1 e2 e3 e i n
%% Creating a common node and element matrix
node=[node_GirderColoumn;node_Pontoon_Spring_Off;node_GirderPontoon_conn;no
de_CableAnchor_E;node_CableAnchor_W;node_CableGirder_E;node_CableGirder_W;n
ode_CableTower_E;node_CableTower_W;node_Girder;node_Pontoon_CB;node_Pontoon
_CG;node_Pontoon_conn;node_Tower];
element=[conn_Pontoon_Girder;conn_CableGirder_E;conn_CableTower_E;conn_Pont
oon_CB;conn_Pontoon_CG;element_Cable;element_Girder;element_Pontoon;element
_Tower;conn_CableGirder_W;conn_CableTower_W];
%% Creating nodes to define the element orientation
%% Defining cable tower beam orientation
%Finding the orientation of the tower beams in xy plane

for i=1:size(L,2)
    if L(i)==380;
        dx=X_g(i+1)-X_g(i);
        dy=Y_g(i+1)-Y_g(i);
    end
end
alfa_tower=-atan(dy/dx);
clearvars dx dy

% Tower coloumns orientation 105000
node_Tower_Orientation=[];
for i= 1:64
    n1=element_Tower(i,2);
    n2=element_Tower(i,3);
    for j=1:size(node,1)
        if node(j,1)==n1
            n1=node(j,2:4);
        elseif node(j,1) == n2
            n2=node(j,2:4);
        end
    end
    dx=n2(1)-n1(1);
    dy=n2(2)-n1(2);
    dz=n2(3)-n1(3);
    beta_tower=atan(dy/dz);
    n(1)=103000+i;
    n(2)=n1(1)+cos(alfa_tower);
    n(3)=n1(2)+sin(alfa_tower);
    n(4)=n1(3)+sin(beta_tower);
    n(5)=n2(3);
    node_Tower_Orientation=[node_Tower_Orientation;n];
end
% top of tower beam orientaion
for i= 65:87
    n1=element_Tower(i,2);
    n2=element_Tower(i,3);
    for j=1:size(node,1)
        if node(j,1)==n1
            n1=node(j,2:4);
        elseif node(j,1) == n2
            n2=node(j,2:4);
        end
    end
end

```

```

dx=n2(1)-n1(1);
dy=n2(2)-n1(2);
dz=n2(3)-n1(3);
n(1)=103000+i;
n(2)=n1(1)+10*cos(alfa_tower);
n(3)=n1(2)+10*sin(alfa_tower);
n(4)=n1(3);
n(5)=n2(3);
node_Tower_Orientation=[node_Tower_Orientation;n];
end
% Middle beam orientation 104000
for i= 88:99
n1=element_Tower(i,2);
n2=element_Tower(i,3);
for j=1:size(node,1)
if node(j,1)==n1
n1=node(j,2:4);
elseif node(j,1) == n2
n2=node(j,2:4);
end
end
dx=n2(1)-n1(1);
dy=n2(2)-n1(2);
dz=n2(3)-n1(3);
n(1)=104000+i;
n(2)=n1(1);
n(3)=n1(2);
n(4)=n1(3)+1;
n(5)=n2(3);
node_Tower_Orientation=[node_Tower_Orientation;n];
end
%% Defining the orientation of the pontoons in degrees from north (global
x axis)
%The navigation channel is rotated 5 degrees from east towards north
alfa_Pontoon=[];alfa_Pontoon(1:10)=1.3828329;% The first 10 pontoons are
fixed in their rotation along the navigation channel.
for i=11:46
dx=X_p(i)-X_p(i-1);
dy=Y_p(i)-Y_p(i-1);
alfa_Pontoon(i)=atan(dy/dx)+pi/2;
end
% Node for girder beam orientation (101 000)
node_Girder_Orientation=[];
for i = 1:size(node_Girder)
n(1)=101000+i;
n(2)=X_g_Orientation;
n(3)=Y_g_Orientation;
n(4)=Z_g(i);
n(5)=L(i);
node_Girder_Orientation=[node_Girder_Orientation;n];
end

%The first node in the pontoon beam is node_Pontoon_conn (102000)
%Thus the node_Pontoon_Orientation should be placed at same height as
%node_Pontoon_conn and based of it's x and y coords.
node_Pontoon_Orientation=[];
for i = 1:46
n(1)=102000+i;
n(2)=node_Pontoon_conn(i,2)+cos(alfa_Pontoon(i));
n(3)=node_Pontoon_conn(i,3)+1;
n(4)=node_Pontoon_conn(i,4);

```

```

n(5)=i;
node_Pontoon_Orientation=[node_Pontoon_Orientation;n];
end

%% Adding orientation nodes to common node matrix
node=[node_Pontoon_Orientation;node_Girder_Orientation;node_Tower_Orientati
on;node];
% Define section properties such as placements, materials data and
stiffness
% Element and materials data
%Data based on rpeort SBJ-30-C3-NOR-90.RE-102 - Appendix B - Global
%analyses rev 0

%K7 Bjørnafjorden end-anchored floating bridge Appendix 1 - Model

%The arc lenght axis is placed -30 metres compared to this model.
k=30;

Girder.Material.E_modulus=      210000*(1000*1000);%N/m^2
Girder.Material.G_modulus=      80769*(1/(1000*1000)); %N/m^2
Girder.Material.Poisson_ratio=  0.3; %m^4
Girder.Material.Alfa=           1.2*10^-5;%1/K

% Axis 11 is the principal axis - the weak axis of the girder (i.e. local y
% axis) and 22 is the strong axis (i.e. local z axis)

Girder.Section1.I22=            115.62; %m^4
Girder.Section1.I11=            2.68; %m^4
Girder.Section1.J=              6.10; %m^4
Girder.Section1.Area=           1.43; %m^2
Girder.Section1.Width=          31; %m
Girder.Section1.Max_height=      3.5; %m
Girder.Section1.Mass=            17836; %Kg/m
Girder.Section1.Wt=              2.30; %m^3
Girder.Section1.Position=        [80,720,1797.5,5435]; %m
Girder.Section1.Drag_factor=     0.529; % -
Girder.Section1.Lift_factor=     0.133; % -
Girder.Section1.Moment_factor=  0.042; % -

Girder.Section2.I22=            132.47; %m^4
Girder.Section2.I11=            3.2; %m^4
Girder.Section2.J=              7.32; %m^4
Girder.Section2.Area=           1.68; %m^2
Girder.Section2.Width=          31; %m
Girder.Section2.Max_height=      3.5; %m
Girder.Section2.Mass=            19798; %Kg/m
Girder.Section2.Wt=              2.76; %m^3
Girder.Section2.Position=        [720,1797.5]; %m
Girder.Section2.Drag_factor=     0.529; % -
Girder.Section2.Lift_factor=     0.133; % -
Girder.Section2.Moment_factor=  0.042; % -

Girder.Section3.I22=            181.1; %m^4
Girder.Section3.I11=            5.049; %m^4
Girder.Section3.J=              10.86; %m^4
Girder.Section3.Area=           2.634; %m^2
Girder.Section3.Width=          31; %m
Girder.Section3.Max_height=      3.5; %m
Girder.Section3.Mass=            27287; %Kg/m

```

```

Girder.Section3.Wt=          3.61;    %m^3
Girder.Section3.Position=    [25,80,5435,5500]; %m
Girder.Section3.Drag_factor= 0.529; % -
Girder.Section3.Lift_factor= 0.133; % -
Girder.Section3.Moment_factor= 0.042; % -

Pontoon.Material.E_modulus=   210000*(1000*1000); %N/m^2
Pontoon.Material.G_modulus=   80769*(1/(1000*1000)); %N/m^2
Pontoon.Material.Poisson_ratio= 0.3; %m^4
Pontoon.Material.Alfa=        1.2*10^-5; %1/K

Pontoon.Section1.Iz=          5.53;    %m^4
Pontoon.Section1.Iy=          5.53;    %m^4
Pontoon.Section1.It=          11.06;   %m^4
Pontoon.Section1.Area=        0.872;   %m^2
Pontoon.Section1.Mass=        7200;    %Kg/m
Pontoon.Section1.Diameter=    7.16;    %m
Pontoon.Section1.Drag_factor= 1.05;    % -
Pontoon.Section1.Positions=   [11:46]; % -
Pontoon.Section1.Displacement= 2793; % m3
Pontoon.Section1.Weight=      750000; %kg

Pontoon.Section2.Iz=          10.23;   %m^4
Pontoon.Section2.Iy=          10.23;   %m^4
Pontoon.Section2.It=          20.46;   %m^4
Pontoon.Section2.Area=        0.977;   %m^2
Pontoon.Section2.Mass=        7956;    %Kg/m
Pontoon.Section2.Diameter=    9.185;   %m
Pontoon.Section2.Drag_factor= 1.05;    % -
Pontoon.Section2.Positions=   [5:10]; % -
Pontoon.Section2.Displacement= 3325; % m3
Pontoon.Section2.Weight=      900000; % kg

Pontoon.Section3.Iz=          14.369;  %m^4
Pontoon.Section3.Iy=          14.369;  %m^4
Pontoon.Section3.It=          28.738;  %m^4
Pontoon.Section3.Area=        1.158;   %m^2
Pontoon.Section3.Mass=        9429;    %Kg/m
Pontoon.Section3.Diameter=    10;      %m
Pontoon.Section3.Drag_factor= 1.05;    % -
Pontoon.Section3.Positions=   [2:4]; % -
Pontoon.Section3.Displacement= 3850; % m3
Pontoon.Section3.Weight=      1040000; % kg

Pontoon.Section4.Iz=          37.47;   %m^4
Pontoon.Section4.Iy=          37.47;   %m^4
Pontoon.Section4.It=          74.94;   %m^4
Pontoon.Section4.Area=        2.101;   %m^2
Pontoon.Section4.Mass=        17317;   %Kg/m
Pontoon.Section4.Diameter=    12;      %m
Pontoon.Section4.Drag_factor= 1.05;    % -
Pontoon.Section4.Positions=   [1]; % -
Pontoon.Section4.Displacement= 4365; % m3
Pontoon.Section4.Weight=      1180000; %kg

Cable.Material.E_eff_modulus= 0; %E/(1+K*a^2*10^-6); %Effective E-modulus based on horizontal cable projection and tangent modulus
Cable.Material.Tangent_modulus= 0.682; %1/m^2;
K=y^2*E*10^6/(12*sigma^3); %Tangent modulus k

```

```

Cable.Material.E_modulus=      195000*10^6;          %N/m (Pa)
Cable.Material.Density=       7850;                %Kg/m^3
Cable.Material.Alfa=          1.2*10^-5;%1/K

Cable.Section.Drag_factor=    0.8;                %-
Cable.Section.Strand_area=    150*(1/(1000*1000));%m^2

% Concrete cable tower
Concrete.Material.E_modulus=   36000e6;%N/m^2
Concrete.Material.G_modulus=   15117*1/(1000*1000);% N/m^2
Concrete.Material.Poisson_ratio=0.2;%m^4
Concrete.Material.Alfa=       1.45*10^(-5);        %1/K
Concrete.Material.Density=    2550;%kg/m^3
Concrete.Section.Cd=2.0;

%Top of tower
Concrete.Section1.I11=        181.248;            %m^4
Concrete.Section1.I22=        92.448;            %m^4
Concrete.Section1.J=          88.8;              %m^4
Concrete.Section1.Area=       21.6;              %m^2
Concrete.Section1.Placement=   [169:228];        %Top of tower from 169
m to 228 meter

%Horizontal beam
Concrete.Section2.I11=        41.1463;            %m^4 1/12*9.5*4^3-
1/12*6.5*2.6^3
Concrete.Section2.I22=        265.55;            %m^4 1/12*(4*9.5^4-
2.6*6.5^3)
Concrete.Section2.J=          45.9;              %m^4          %%%
MUST BE FURTHER CHECKED %%%
Concrete.Section2.Area=       21.1;              %m^2
Concrete.Section2.Placement=   [48];

Concrete.Section3.I11=        63.312;            %m^4
Concrete.Section3.I22=        50.112;            %m^4
Concrete.Section3.J=          13.2;              %m^4
Concrete.Section3.Area=       14.4;              %m^2
Concrete.Section3.Placement=   [169];           %m

Concrete.Section4.I11=        272.1;              %m^4
Concrete.Section4.I22=        226.1;              %m^4
Concrete.Section4.J=          45.9;              %m^4
Concrete.Section4.Area=       23.8;              %m^2
Concrete.Section4.Placement=   [48];           %m

Concrete.Section5.I11=        733.9;              %m^4
Concrete.Section5.I22=        569.1;              %m^4
Concrete.Section5.J=          164.8;              %m^4
Concrete.Section5.Area=       33.6;              %m^2
Concrete.Section5.Placement=   [5];            %m

%% Define tower cross section variable properties
% TORSIONAL CONSTANT, J, AFTER ST: Venants TORSION CONSTANT FOR VARIABLE
CROSS SECTION BEAM SECTIONS
% AS WELL AS BEDNING AND SHEAR STIFFNESS
h=[12];
dh1=2/8.;
dh2=4/(33-9+89-65);
for i =2:9;
    h(i)=h(i-1)-dh1;
end

```

```

for i =10:33+89-65
    h(i)=h(i-1)-dh2;
end
b=[6];
db1=2/8.;
for i=2:9;
    b(i)=b(i-1)-db1;
end
for i=10:32;
    b(i)=4;
end
for i=33:(89-65)+33;
    b(i)=6;
end
JTower=[];
for i = 1:size(h,2);
    aj=h(i);
    bj=b(i);
    t1=1;
    t2=1.2;
    t3=t1;
    t4=t2;

    JTower(i)=(4*(aj^2)*(bj^2))/((aj/t1)+(aj/t2)+(bj/t2)+(bj/t4));
end
IyyTower=[]; % MAIN AXIS (STRONG)
IxxTower=[]; % WEAK AXIS (WEAK)
for i =1:size(h,2);
    h1=h(i);
    h2=h(i)-2*1.2;
    b1=b(i);
    b2=b(i)-2*1;
    IyyTower(i)=1/12*((b1*h1^3)-(b2*h2^3));
    IxxTower(i)=1/12*((h1*b1^3)-(h2*b2^3));
end
ATower=[];
for i = 1:size(h,2);
    h1=h(i);
    h2=h(i)-2*1.2;
    b1=b(i);
    b2=b(i)-2*1;
    ATower(i)=h1*b1-h2*b2;
end
%% Define and write boundaries to inp file
% Defining boundary conditions
%Bridge is restricted from all motion at L=25, at both tower legs and at L
%= 5500. also, the anchor is restrained in all degrees.
% Defining the girder boundary
boundary_Girder=[];
for i = 1:size(node_Girder)
    if node_Girder(i,5)==25
        boundary_Girder=[boundary_Girder;node_Girder(i,:)];
    elseif node_Girder(i,5)==5480
        boundary_Girder=[boundary_Girder;node_Girder(i,:)];
    end
end
end
% Defining the tower boundary
boundary_Tower=[node_Tower(1,:);node_Tower(33,:)];
% Defining the anchor boundary
boundary_Anchor=[node_CableAnchor_E;node_CableAnchor_W];
% Printing boundary conditions for alle directions

```



```

bound=[boundary_Tower;boundary_Girder];
fileID=fopen('INPUTFILES\boundaries.inp','w');
%% fprintf(fileID,'*BOUNDARY\n');
for i=1:size(bound,1)
line=[bound(i,1),1,6,0.0];
fprintf(fileID,'%d,%d,%d,%f\n',line);
line=[];
end
%% Printing Boundaries for 1-3 directions (Cable anchor)
for i=1:size(boundary_Anchor,1)
line=[boundary_Anchor(i,1),1,6,0.0];
fprintf(fileID,'%d,%d,%d,%f\n',line);
line=[];
end
fprintf(fileID,'**');
fclose(fileID);
%% Defining, loading and creating cables inp file
% Finding effective E-modulus for cables
a=[];
Eff_E_modulus_cable=[];
for i=1:size(element_Cable,1)
a(i)=H_Cable_Projection(element_Cable(i,1),element,node);

Eff_E_modulus_cable=[Eff_E_modulus_cable;element_Cable(i,4),Eff_E_mod(a(i),
Cable.Material.E_modulus)];
end
%% Defining weight per cable
Initial_cable_area=0.01;
Cable_PreStress2=[];
%Finding dL between each cable
for i = 1: size(element_Cable,1)
n1=element_Cable(i,2);
n2=element_Cable(i,3);
for j = 1:size(node,1);
if n1==node(j,1)
n1=node(j,1:4);
elseif n2==node(j,1)
n2=node(j,1:4);
end
end
dx=n1(2)-n2(2);
dy=n1(3)-n2(3);
dz=abs(n1(4)-n2(4));
Lxy=sqrt(dx^2+dy^2);
Lxyz=sqrt(dx^2+dy^2+dz^2);
%F_cable*sin(alfa_cable)=Weight*g
% -> F_cable=Weight*g/sin(alfa_cable)
%alfa_cable = atan(dz/L)
alfa_cable=atan(dz/Lxy);

Weight=Lxyz*Cable_Area(i)*Cable.Material.Density+20*Girder.Section1.Mass;
Cable_PreStress2(i)=Weight*9.81/sin(alfa_cable);
end
%% Creating alfa based on dt = 10 degrees
% Stress = alfa * E * dT -> alfa = Stress / E*dT
Cable_Alfa=[];
for i = 1 : size(Cable_PreStress,2)
Cable_Alfa(i)=520.8e6/(Eff_E_modulus_cable(i,2)*10);
end
%% Finding no of strands in each cable
% A*stress=F

```

```

Cable_Area=[];
Cable_nStrands=[];
for i = 1:size(Cable_PreStress,2);
    Cable_Area(i)=Cable_PreStress2(i)/(520.8*1000000);
end
for i=1:size(Cable_Area,2)
    Cable_nStrands(i)=ceil(Cable_Area(i)/Cable.Section.Strand_area);
    Cable_Area(i)=Cable_nStrands(i)*Cable.Section.Strand_area;
end
%% CREATING CABLES INPUT FILE
% Cables
fileID=fopen('INPUTFILES\cables.inp','w');
for i = 1:size(element_Cable,1)
    fprintf(fileID, '*ELEMENT,TYPE=B31,ELSET=cable%d\n',element_Cable(i,4));
    line=[element_Cable(i,1:3)];
    fprintf(fileID, '%d,%d,%d\n',line);
    fprintf(fileID, '*BEAM GENERAL SECTION, DENSITY=%f,
ELSET=cable%d\n%f,0.01,0.,0.01,0.01\n\n%f,,%f,10\n', [Cable.Material.Density
,element_Cable(i,4),Cable_Area(i),Eff_E_modulus_cable(i,2),Cable_Alfa(i)*ca
ble_mult]);
    fprintf(fileID, '*DAMPING,
ALPHA=%f,BETA=%f\n', [AlfaDamping,BetaDamping]);
    fprintf(fileID, '*ELSET,ELSET=CABLES\ncable%d\n',element_Cable(i,4));
end
fprintf(fileID, '*ELSET,ELSET=GravityElement\nCABLES\n');
fprintf(fileID, '*ELSET,ELSET=cables\n');
for i = 1:size(element_Cable,1)
    fprintf(fileID, '%d\n',element_Cable(i,1));
end
fprintf(fileID, '**');
fclose(fileID);
% Printing temperature boundary conditions
fileID=fopen('INPUTFILES\CABLE_InitialTemp.inp','w');
for i = 1 : size(element_Cable,1)
    for j=2:3
        fprintf(fileID, '%d,10.\n',element_Cable(i,j));
    end
end
fprintf(fileID, '**');
fclose(fileID);
% Printing temperature boundary conditions
fileID=fopen('INPUTFILES\CABLE_PreStressTemp.inp','w');
fprintf(fileID, '*TEMPERATURE\n');
for i = 1 : size(element_Cable,1)
    for j=2:3
        fprintf(fileID, '%d,0.\n',element_Cable(i,j));
    end
end
fprintf(fileID, '**');
fclose(fileID);
%% Defining the pontoon spring stiffnes and connector elements
% PontoonSprings

Pontoon.Section1.Lenght           = 58           ; %m
Pontoon.Section1.Width             = 10           ; %m
Pontoon.Section1.Radius_end        = 5            ; %m
Pontoon.Section1.Draft              = 5            ; %m
Pontoon.Section1.Total_volume      = 5027         ; %m^3
Pontoon.Section1.Displacement       = 2793         ; %m^2
Pontoon.Section1.Marine_growth      = 560e3/9.81    ;
%kg

```

```

Pontoon.Section1.Roll_Stiffness =
stiffness(Pontoon.Section1.Lenght,Pontoon.Section1.Width); %m^4
Pontoon.Section1.Pitch_Stiffness =
stiffness(Pontoon.Section1.Width,Pontoon.Section1.Lenght) ; %m^4
Pontoon.Section1.Heave_Stiffness =
((Pontoon.Section1.Lenght-
2*Pontoon.Section1.Radius_end)*(Pontoon.Section1.Width)+pi*Pontoon.Section1
.Radius_end^2)*9.81*1000 ; %N/m
Pontoon.Section1.Wamit_Reference = 34 ; %
Pontoon.Section1.Drag_coef_x_drag_area_surge = 32 ; %m^2
Pontoon.Section1.Drag_coef_x_drag_area_sway = 232 ; %m^2
Pontoon.Section1.Drag_coef_x_drag_area_surge_submerged = 40 ; %m^2
Pontoon.Section1.Drag_coef_x_drag_area_sway_submerged = 290 ; %m^2;

Pontoon.Section2.Lenght = 58 ; %m
Pontoon.Section2.Width = 12 ; %m
Pontoon.Section2.Radius_end = 6 ; %m
Pontoon.Section2.Draft = 5 ; %m
Pontoon.Section2.Total_volume = 5986 ; %m^3
Pontoon.Section2.Displacement = 3325 ; %m^2
Pontoon.Section2.Marine_growth = 615e3/9.81 ;
%kg
Pontoon.Section2.Roll_Stiffness =
stiffness(Pontoon.Section2.Lenght,Pontoon.Section2.Width); %m^4
Pontoon.Section2.Pitch_Stiffness =
stiffness(Pontoon.Section2.Width,Pontoon.Section2.Lenght); %m^4
Pontoon.Section2.Heave_Stiffness =
((Pontoon.Section2.Lenght-
2*Pontoon.Section2.Radius_end)*(Pontoon.Section2.Width)+pi*Pontoon.Section2
.Radius_end^2)*9.81*1000;%6.68e6 ; %N/m
Pontoon.Section2.Wamit_Reference = 35 ; %
Pontoon.Section2.Drag_coef_x_drag_area_surge = 38.4 ; %m^2
Pontoon.Section2.Drag_coef_x_drag_area_sway = 232 ; %m^2
Pontoon.Section2.Drag_coef_x_drag_area_surge_submerged = 48 ; %m^2
Pontoon.Section2.Drag_coef_x_drag_area_sway_submerged = 290 ; %m^2

Pontoon.Section3.Lenght = 58 ; %m
Pontoon.Section3.Width = 14 ; %m
Pontoon.Section3.Radius_end = 7 ; %m
Pontoon.Section3.Draft = 5 ; %m
Pontoon.Section3.Total_volume = 6929 ; %m^3
Pontoon.Section3.Displacement = 3850 ; %m^2
Pontoon.Section3.Marine_growth = 669e3/9.81 ;
%kg
Pontoon.Section3.Roll_Stiffness =
stiffness(Pontoon.Section3.Lenght,Pontoon.Section3.Width) ; %m^4
Pontoon.Section3.Pitch_Stiffness =
stiffness(Pontoon.Section3.Width,Pontoon.Section3.Lenght) ; %m^4
Pontoon.Section3.Heave_Stiffness =
((Pontoon.Section3.Lenght-
2*Pontoon.Section3.Radius_end)*(Pontoon.Section3.Width)+pi*Pontoon.Section3
.Radius_end^2)*9.81*1000;% 7.73e6 ; %N/m
Pontoon.Section3.Wamit_Reference = 36 ; %
Pontoon.Section3.Drag_coef_x_drag_area_surge = 44.8 ; %m^2
Pontoon.Section3.Drag_coef_x_drag_area_sway = 232 ; %m^2
Pontoon.Section3.Drag_coef_x_drag_area_surge_submerged = 56 ; %m^2
Pontoon.Section3.Drag_coef_x_drag_area_sway_submerged = 290 ; %m^2

Pontoon.Section4.Lenght = 58 ; %m

```

```

Pontoon.Section4.Width = 16 ; %m
Pontoon.Section4.Radius_end = 8 ; %m
Pontoon.Section4.Draft = 5 ; %m
Pontoon.Section4.Total_volume = 7858 ; %m^3
Pontoon.Section4.Displacement = 4365 ; %m^2
Pontoon.Section4.Marine_growth = 723e3/9.81 ;
%kg
Pontoon.Section4.Roll_Stiffness =
stiffness(Pontoon.Section4.Lenght,Pontoon.Section4.Width) ; %m^4
Pontoon.Section4.Pitch_Stiffness =
stiffness(Pontoon.Section4.Width,Pontoon.Section4.Lenght) ; %m^4
Pontoon.Section4.Heave_Stiffness =
((Pontoon.Section4.Lenght-
2*Pontoon.Section4.Radius_end)*(Pontoon.Section4.Width)+pi*Pontoon.Section4
.Radius_end^2)*9.81*1000;%8.78e6 ; %N/m
Pontoon.Section4.Wamit_Reference = 37 ; %
Pontoon.Section4.Drag_coef_x_drag_area_surge = 51.2 ; %m^2
Pontoon.Section4.Drag_coef_x_drag_area_sway = 232 ; %m^2
Pontoon.Section4.Drag_coef_x_drag_area_surge_submerged = 64 ; %m^2
Pontoon.Section4.Drag_coef_x_drag_area_sway_submerged = 290 ; %m^2

%% Defining spring directions
% Coincide with the pontoon direction
% Connected to the pontoon CG
Spring_Orientation=[];
for i =1:46
    xa=node_Pontoon_Orientation(i,2)-node_Pontoon_conn(i,2);
    ya=node_Pontoon_Orientation(i,3)-node_Pontoon_conn(i,3);
    za=node_Pontoon_Orientation(i,4)-node_Pontoon_conn(i,4);
    xb=0;
    yb=0;
    zb=node_Pontoon_Orientation(i,4)-node_Pontoon_conn(i,4);
    Spring_Orientation=[Spring_Orientation;i, xa, ya, za, xb, yb, zb];
end
%% Printing springs into input
% ELEMENT 30,000
Pontoon_Spring=[];
for i=1:46
    e13=40000+i;
    e15=40100+i;
    e14=40200+i;
    e2=node_Pontoon_CB(i);
    e3=(alfa_Pontoon(i)*180/3.14); % IN DEGREES
    if i >=Pontoon.Section1.Positions(1) &&
i<=Pontoon.Section1.Positions(size(Pontoon.Section1.Positions,2))
        e4=Pontoon.Section1.Heave_Stiffness;
        e5=Pontoon.Section1.Roll_Stiffness;
        e6=Pontoon.Section1.Pitch_Stiffness;
    elseif i >=Pontoon.Section2.Positions(1) &&
i<=Pontoon.Section2.Positions(size(Pontoon.Section2.Positions,2))
        e4=Pontoon.Section2.Heave_Stiffness;
        e5=Pontoon.Section2.Roll_Stiffness;
        e6=Pontoon.Section2.Pitch_Stiffness;
    elseif i >=Pontoon.Section3.Positions(1) &&
i<=Pontoon.Section3.Positions(size(Pontoon.Section3.Positions,2))
        e4=Pontoon.Section3.Heave_Stiffness;
        e5=Pontoon.Section3.Roll_Stiffness;
        e6=Pontoon.Section3.Pitch_Stiffness;
    elseif i >=Pontoon.Section4.Positions(1) &&
i<=Pontoon.Section4.Positions(size(Pontoon.Section4.Positions,2))
        e4=Pontoon.Section4.Heave_Stiffness;

```

```

        e5=Pontoon.Section4.Roll_Stiffness;
        e6=Pontoon.Section4.Pitch_Stiffness;
    end
    Pontoon_Spring=[Pontoon_Spring;e13,e15,e14,e2,e3,e4,e5,e6];
end
Pontoon_Spring_Initial(1:46)=-20.;
fileID=fopen('INPUTFILES\PONTOONSPRINGS.inp','w');
for i = 1:46

fprintf(fileID, '*ORIENTATION, NAME=ORISPRING%d\n%f, %f, %f, %f, %f, %f, %f, %f\n',
[i, node_Pontoon_CB(i,2)+cos(alfa_Pontoon(i)), node_Pontoon_CB(i,3)+sin(alfa_Pontoon(i)),
node_Pontoon_CB(i,4)+3.14/2), node_Pontoon_CB(i,2)+cos(alfa_Pontoon(i)+3.14/2),
node_Pontoon_CB(i,3)+sin(alfa_Pontoon(i)+3.14/2), node_Pontoon_CB(i,4),
node_Pontoon_CB(i,2), node_Pontoon_CB(i,3), node_Pontoon_CB(i,4)]);
    fprintf(fileID, '*ELEMENT, TYPE=SPRING1, ELSET=HEAVESPRING%d\n', i);
    fprintf(fileID, '%d, %d\n', Pontoon_Spring(i,1), Pontoon_Spring(i,4));

fprintf(fileID, '*SPRING, ELSET=HEAVESPRING%d, ORIENTATION=ORISPRING%d\n3\n%f\n',
[i, i, Pontoon_Spring(i,6)]);
    fprintf(fileID, '*ELEMENT, TYPE=SPRING1, ELSET=ROLLSPRING%d\n', i);
    fprintf(fileID, '%d, %d\n', Pontoon_Spring(i,2), Pontoon_Spring(i,4));

fprintf(fileID, '*SPRING, ELSET=ROLLSPRING%d, ORIENTATION=ORISPRING%d\n4\n%f\n',
[i, i, Pontoon_Spring(i,7)]);
    fprintf(fileID, '*ELEMENT, TYPE=SPRING1, ELSET=PITCHSPRING%d\n', i);
    fprintf(fileID, '%d, %d\n', Pontoon_Spring(i,3), Pontoon_Spring(i,4));

fprintf(fileID, '*SPRING, ELSET=PITCHSPRING%d, ORIENTATION=ORISPRING%d\n5\n%f\n',
[i, i, Pontoon_Spring(i,8)]);

end
fprintf(fileID, '**');
fclose(fileID);
%% Creating the initial updrift for the pontoons by offsetting the
boundaries of the node connecting to the heave spring
Pontoon_Buoyancy=[];
for i=1:46
    if i >=Pontoon.Section1.Positions(1) &&
i<=Pontoon.Section1.Positions(size(Pontoon.Section1.Positions,2))
        e=Pontoon.Section1.Heave_Stiffness*pontoon_mult1;
        elseif i >=Pontoon.Section2.Positions(1) &&
i<=Pontoon.Section2.Positions(size(Pontoon.Section2.Positions,2))
        e=Pontoon.Section2.Heave_Stiffness*pontoon_mult24;
        elseif i >=Pontoon.Section3.Positions(1) &&
i<=Pontoon.Section3.Positions(size(Pontoon.Section3.Positions,2))
        e=Pontoon.Section3.Heave_Stiffness*pontoon_mult511;
        elseif i >=Pontoon.Section4.Positions(1) &&
i<=Pontoon.Section4.Positions(size(Pontoon.Section4.Positions,2))
        e=Pontoon.Section4.Heave_Stiffness*pontoon_mult1246;
    end
    Pontoon_Buoyancy(i)=e;
end
%% Creating the updrift concentrated load
fileID=fopen('INPUTFILES\Pontoon_Buoyancy.inp','w');
for i=1:46
    NodeNumber=node_Pontoon_CG(i,1);
    fprintf(fileID, '%d, 3, %f\n', [NodeNumber, Pontoon_Buoyancy(i)]);
    line=[];
end
fprintf(fileID, '**');
fclose(fileID);

```

```

%% Create mass elements
% Creating the point mass element for the pontoons (20 000)
% The mass of pontoons + marine growth + ballast (0 during operations)
element_Pontoon_Mass=[];
e=[];
for i = 1:46
    e(1)=20000+i;
    e(2)=node_Pontoon_CG(i,1);
    if i >=Pontoon.Section1.Positions(1) &&
i<=Pontoon.Section1.Positions(size(Pontoon.Section1.Positions,2))
        e(3)=Pontoon.Section1.Weight+Pontoon.Section1.Marine_growth;
    elseif i >=Pontoon.Section2.Positions(1) &&
i<=Pontoon.Section2.Positions(size(Pontoon.Section2.Positions,2))
        e(3)=Pontoon.Section2.Weight+Pontoon.Section2.Marine_growth;
    elseif i >=Pontoon.Section3.Positions(1) &&
i<=Pontoon.Section3.Positions(size(Pontoon.Section3.Positions,2))
        e(3)=Pontoon.Section3.Weight+Pontoon.Section3.Marine_growth;
    elseif i >=Pontoon.Section4.Positions(1) &&
i<=Pontoon.Section4.Positions(size(Pontoon.Section4.Positions,2))
        e(3)=Pontoon.Section4.Weight+Pontoon.Section4.Marine_growth;
    end
    element_Pontoon_Mass=[element_Pontoon_Mass;e];
end
%% Creating point mass element for the reinforced sections of girder deck
element_Girder_Reinforced_Mass=[];
e=[];
% FROM TABLE 3-2 in main report
%axis 3
for i=1:size(node_Girder,1);
    axismass=101000;
    if node_Girder(i,5)==860;
        element_Girder_Reinforced_Mass=[node_Girder(i,1),axismass];
    end
end
%axis 4 - 6
axismass=71000;
axisL=[960,1060,1160];
for i=1:size(node_Girder,1);
    for j=1:size(axisL,2)
        if node_Girder(i,5)==axisL(j);

element_Girder_Reinforced_Mass=[element_Girder_Reinforced_Mass;node_Girder(
i,1),axismass];
        end
    end
end
% axis 7-12
axismass=59000;
axisL=[1260,1360,1460,1560,1660,1760];
for i=1:size(node_Girder,1);
    for j=1:size(axisL,2)
        if node_Girder(i,5)==axisL(j);

element_Girder_Reinforced_Mass=[element_Girder_Reinforced_Mass;node_Girder(
i,1),axismass];
        end
    end
end
%axis 13-47
axismass=55000;
for i=1:47-(13-1)

```

```

    axisL(i)=1760+100*i;
end
for i=1:size(node_Girder,1);
    for j=1:size(axisL,2)
        if node_Girder(i,5)==axisL(j);

element_Girder_Reinforced_Mass=[element_Girder_Reinforced_Mass;node_Girder(
i,1),axismass];
        end
        end
end
%axis 48
axismass=71000;
axisL=axisL(size(axisL,2))+100;
for i=1:size(node_Girder,1);
    for j=1:size(axisL,2)
        if node_Girder(i,5)==axisL(j);

element_Girder_Reinforced_Mass=[element_Girder_Reinforced_Mass;node_Girder(
i,1),axismass];
        end
        end
end
%Adding element number 21000
temp=element_Girder_Reinforced_Mass;
element_Girder_Reinforced_Mass=[];
for i=1:size(temp,1);

element_Girder_Reinforced_Mass=[element_Girder_Reinforced_Mass;21000+i,temp
(i,1),temp(i,2)];
end
%% Writing elements and nodes to inp files
% Creating input file for girder nodes
fileID=fopen('INPUTFILES\nodes.inp','w');
fprintf(fileID,'*NODE,NSET=all\n');
for i = 1:size(node,1)
line=node(i,1:4);
fprintf(fileID,'%d,%f,%f,%f\n',line);
end
fprintf(fileID,'*NSET,NSET=CableAnchor_E\n');
for i = 1:size(node_CableAnchor_E,1)
line=node_CableAnchor_E(i,1);
fprintf(fileID,'%d\n',line);
end
fprintf(fileID,'*NSET,NSET=CableAnchor_W\n');
for i = 1:size(node_CableAnchor_W,1)
line=node_CableAnchor_W(i,1);
fprintf(fileID,'%d\n',line);
end
fprintf(fileID,'*NSET,NSET=CableGirder_E\n');
for i = 1:size(node_CableGirder_E,1)
line=node_CableGirder_E(i,1);
fprintf(fileID,'%d\n',line);
end
fprintf(fileID,'*NSET,NSET=CableGirder_W\n');
for i = 1:size(node_CableGirder_W,1)
line=node_CableGirder_W(i,1);
fprintf(fileID,'%d\n',line);
end

fprintf(fileID,'*NSET,NSET=CableTower_E\n');

```

```

for i = 1:size(node_CableTower_E,1)
line=node_CableTower_E(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=CableTower_W\n');
for i = 1:size(node_CableTower_W,1)
line=node_CableTower_W(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=Girder\n');
for i = 1:size(node_Girder,1)
line=node_Girder(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=GirderOutput\n');
k=-1;
while k<272
    k=k+2;
line=node_Girder(k,1)
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=Girder_Orientation\n');
for i = 1:size(node_Girder_Orientation,1)
line=node_Girder_Orientation(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=GirderPontoon_conn\n');
for i = 1:size(node_GirderPontoon_conn,1)
line=node_GirderPontoon_conn(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=Pontoon_CB\n');
for i = 1:size(node_Pontoon_CB,1)
line=node_Pontoon_CB(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=Pontoon_CG\n');
for i = 1:size(node_Pontoon_CG,1)
line=node_Pontoon_CG(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=Pontoon_conn\n');
for i = 1:size(node_Pontoon_conn,1)
line=node_Pontoon_conn(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=Pontoon_Orientation\n');
for i = 1:size(node_Pontoon_Orientation,1)
line=node_Pontoon_Orientation(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=Tower\n');
for i = 1:size(node_Tower,1)
line=node_Tower(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '*NSET,NSET=Tower_Orientation\n');
for i = 1:size(node_Tower_Orientation,1)
line=node_Tower_Orientation(i,1);
fprintf(fileID, '%d\n', line);
end

```



```

fprintf(fileID, '*NSET,NSET=node_Pontoon_Spring_Off\n');
for i = 1:size(node_Pontoon_Spring_Off,1)
line=node_Pontoon_Spring_Off(i,1);
fprintf(fileID, '%d\n', line);
end
fprintf(fileID, '**');
fclose(fileID);
%% Defining connector elements and printing to inp
connector=[conn_CableGirder_E;conn_CableGirder_W;conn_CableTower_E;conn_CableTower_W;conn_Pontoon_Girder;conn_Pontoon_CB;conn_Pontoon_CG];
fileID=fopen('INPUTFILES\connectors.inp','w');
fprintf(fileID, '*ELEMENT,TYPE=B31,ELSET=connectors\n');
for i = 1:size(connector,1)
line=connector(i,1:3);
fprintf(fileID, '%d,%d,%d\n', line);
end
matdata=[3,50,100,100,210.0e9];
fprintf(fileID, '*BEAM GENERAL SECTION, DENSITY=1, ELSET=connectors\n%f,%f,0.,%f,%f\n1.0,0.0,0.0\n%f\n',matdata);
fclose(fileID);
%% Defining element types and materials and printing to inp
% *** Element 1 ***
% Girder section 1
% Position of elements: Girder.Section1.Position Provided in arc length
% coordinates
girder01=[]; % Put together the element matrixes belonging to element 1
for i = 1:size(element_Girder,1)
    pos=element_Girder(i,4);
    for j = 1:2:size(Girder.Section1.Position,2)
        if pos>=Girder.Section1.Position(j) &&
pos<Girder.Section1.Position(j+1)
girder01=[girder01;element_Girder(i,1:3),node_Girder_Orientation(i,1)];
        end
    end
end
fileID=fopen('INPUTFILES\girder01.inp','w');
for i = 1:size(girder01,1)
    fprintf(fileID, '*ELEMENT,TYPE=B31,ELSET=girder01%d\n',girder01(i,1));
    fprintf(fileID, '%d,%d,%d\n',girder01(i,1:3));
    n1=girder01(i,2);
    n2=girder01(i,4);
    for j = 1:size(node,1);
        if n1==node(j,1);
            x1=node(j,2);
            y1=node(j,3);
        elseif n2==node(j,1);
            x2=node(j,2);
            y2=node(j,3);
        end
    end
    xcomp=(x2-x1);
    ycomp=(y2-y1);
    zcomp=0;
    xcomp=xcomp/sqrt(xcomp^2+ycomp^2);
    ycomp=ycomp/sqrt(xcomp^2+ycomp^2);

matdata=[Girder.Section1.Mass/Girder.Section1.Area,girder01(i,1),Girder.Section1.Area,Girder.Section1.I11,Girder.Section1.I22,Girder.Section1.J,xcomp,ycomp,zcomp,Girder.Material.E_modulus];

```

```

        fprintf(fileID, '*BEAM GENERAL SECTION, DENSITY=%f,
ELSET=girder01%d\n%f,%f,0.,%f,%f\n%f,%f,%f\n%f\n',matdata);
        fprintf(fileID, '*DAMPING,
ALPHA=%f,BETA=%f\n', [AlfaDamping,BetaDamping]);
        fprintf(fileID, '*SECTION
POINTS\n%f,%f,%f,%f,%f,%f,%f,%f\n',girdersectionpoints);
        fprintf(fileID, '*ELSET,ELSET=GIRDER\ngirder01%d\n',girder01(i,1));
end
fprintf(fileID, '**');
fclose(fileID);

% *** Element 2 ***
% Girder section 2
% Material: Girder.Material
girder02=[];
for i = 1:size(element_Girder,1)
    pos=element_Girder(i,4);
    for j = 1:2:size(Girder.Section2.Position,2)
        if pos>=Girder.Section2.Position(j) &&
pos<Girder.Section2.Position(j+1)

girder02=[girder02;element_Girder(i,1:3),node_Girder_Orientation(i,1)];
        end
    end
end
fileID=fopen('INPUTFILES\girder02.inp','w');
for i = 1:size(girder02,1)
    fprintf(fileID, '*ELEMENT,TYPE=B31,ELSET=girder02%d\n',girder02(i,1));
    fprintf(fileID, '%d,%d,%d\n',girder02(i,1:3));

    n1=girder02(i,2);
    n2=girder02(i,4);
    for j = 1:size(node,1);
        if n1==node(j,1);
            x1=node(j,2);
            y1=node(j,3);
        elseif n2==node(j,1);
            x2=node(j,2);
            y2=node(j,3);
        end
    end
    xcomp=(x2-x1);
    ycomp=(y2-y1);
    zcomp=0;
    xcomp=xcomp/sqrt(xcomp^2+ycomp^2);
    ycomp=ycomp/sqrt(xcomp^2+ycomp^2);
    matdata=[Girder.Section2.Mass/Girder.Section2.Area,girder02(i,1),Girder.Sec
tion2.Area,Girder.Section2.I11,Girder.Section2.I22,Girder.Section2.J,xcomp,
ycomp,zcomp,Girder.Material.E_modulus];
    fprintf(fileID, '*BEAM GENERAL SECTION, DENSITY=%f,
ELSET=girder02%d\n%f,%f,0.,%f,%f\n%f,%f,%f\n%f\n',matdata);
    fprintf(fileID, '*DAMPING, ALPHA=%f,BETA=%f\n', [AlfaDamping,BetaDamping]);
    fprintf(fileID, '*SECTION
POINTS\n%f,%f,%f,%f,%f,%f,%f,%f\n',girdersectionpoints);
    fprintf(fileID, '*ELSET,ELSET=GIRDER\ngirder02%d\n',girder02(i,1));
end
fprintf(fileID, '**');
fclose(fileID);

% *** Element 3 ***

```

```

% Girder section 3
% Material: Girder.Material
girder03=[];
for i = 1:size(element_Girder,1)
    pos=element_Girder(i,4);
    for j = 1:2:size(Girder.Section3.Position,2)
        if pos>=Girder.Section3.Position(j) &&
pos<Girder.Section3.Position(j+1)

girder03=[girder03;element_Girder(i,1:3),node_Girder_Orientation(i,1)];
        end
    end
end
end
fileID=fopen('INPUTFILES\girder03.inp','w');
for i = 1:size(girder03,1)
    fprintf(fileID, '*ELEMENT,TYPE=B31,ELSET=girder03%d\n',girder03(i,1));
    fprintf(fileID, '%d,%d,%d\n',girder03(i,1:3));
    n1=girder02(i,2);
    n2=girder02(i,4);
    for j = 1:size(node,1);
        if n1==node(j,1);
            x1=node(j,2);
            y1=node(j,3);
        elseif n2==node(j,1);
            x2=node(j,2);
            y2=node(j,3);
        end
    end
    xcomp=(x2-x1);
    ycomp=(y2-y1);
    zcomp=0;
    xcomp=xcomp/sqrt(xcomp^2+ycomp^2);
    ycomp=ycomp/sqrt(xcomp^2+ycomp^2);
matdata=[Girder.Section3.Mass/Girder.Section3.Area,girder03(i,1),Girder.Sec
tion3.Area,Girder.Section3.I11,Girder.Section3.I22,Girder.Section3.J,xcomp,
ycomp,zcomp,Girder.Material.E_modulus];
    fprintf(fileID, '*BEAM GENERAL SECTION, DENSITY=%f,
ELSET=girder03%d\n%f,%f,0.,%f,%f\n%f,%f,%f\n%f\n',matdata);
    fprintf(fileID, '*DAMPING, ALPHA=%f,BETA=%f\n',[AlfaDamping,BetaDamping]);
    fprintf(fileID, '*SECTION
POINTS\n%f,%f,%f,%f,%f,%f,%f\n',girdersectionpoints);
    fprintf(fileID, '*ELSET,ELSET=GIRDER\ngirder03%d\n',girder03(i,1));
end
fprintf(fileID, '*ELSET,ELSET=GravityElement\nGIRDER\n');
fprintf(fileID, '***');
fclose(fileID);
% *** Element 4 ***
%% Pontoon type 1
fileID=fopen('INPUTFILES\pontoon.inp','w');
for i = 1:size(element_Pontoon,1)
    if element_Pontoon(i,4)==1
        type=4;
    elseif element_Pontoon(i,4)>1 && element_Pontoon(i,4)<5
        type=3;
    elseif element_Pontoon(i,4)>4 && element_Pontoon(i,4)<11
        type=2;
    else
        type=1;
    end
end
fprintf(fileID, '*ELEMENT,TYPE=B31,ELSET=pontoon%d\n',type);
fprintf(fileID, '%d,%d,%d\n',element_Pontoon(i,1:3));

```

```

end
matdata=[Pontoon.Section1.Mass/Pontoon.Section1.Area,Pontoon.Material.Poiss
on_ratio,Pontoon.Section1.Area,Pontoon.Section1.Iy,Pontoon.Section1.Iz,Pont
oon.Section1.It,Pontoon.Material.E_modulus];
fprintf(fileID,'*BEAM GENERAL SECTION, DENSITY=%f, ELSET=pontoon1,
POISSON=%f\n%f,%f,0.,%f,%f\n1.0,0.0,0.0\n%f\n',matdata);
fprintf(fileID,'*DAMPING, ALPHA=%f,BETA=%f\n',[AlfaDamping,BetaDamping]);
fprintf(fileID,'*ELSET,ELSET=PONTOONS\npontoon1\n');
%% Pontoon type 2
matdata=[Pontoon.Section2.Mass/Pontoon.Section2.Area,Pontoon.Material.Poiss
on_ratio,Pontoon.Section2.Area,Pontoon.Section2.Iy,Pontoon.Section2.Iz,Pont
oon.Section2.It,Pontoon.Material.E_modulus];
fprintf(fileID,'*BEAM GENERAL SECTION, DENSITY=%f, ELSET=pontoon2,
POISSON=%f\n%f,%f,0.,%f,%f\n1.0,0.0,0.0\n%f\n',matdata);
fprintf(fileID,'*DAMPING, ALPHA=%f,BETA=%f\n',[AlfaDamping,BetaDamping]);
fprintf(fileID,'*ELSET,ELSET=PONTOONS\npontoon2\n');
% *** Element 6 ***
%% Pontoon type 3
matdata=[Pontoon.Section3.Mass/Pontoon.Section3.Area,Pontoon.Material.Poiss
on_ratio,Pontoon.Section3.Area,Pontoon.Section3.Iy,Pontoon.Section3.Iz,Pont
oon.Section3.It,Pontoon.Material.E_modulus];
fprintf(fileID,'*BEAM GENERAL SECTION, DENSITY=%f, ELSET=pontoon3,
POISSON=%f\n%f,%f,0.,%f,%f\n1.0,0.0,0.0\n%f\n',matdata);
fprintf(fileID,'*DAMPING, ALPHA=%f,BETA=%f\n',[AlfaDamping,BetaDamping]);
fprintf(fileID,'*ELSET,ELSET=PONTOONS\npontoon3\n');
% *** Element 7 ***
%% Pontoon type 4
% Material: Pontoon.Material
matdata=[Pontoon.Section4.Mass/Pontoon.Section4.Area,Pontoon.Material.Poiss
on_ratio,Pontoon.Section4.Area,Pontoon.Section4.Iy,Pontoon.Section4.Iz,Pont
oon.Section4.It,Pontoon.Material.E_modulus];
fprintf(fileID,'*BEAM GENERAL SECTION, DENSITY=%f, ELSET=pontoon4,
POISSON=%f\n%f,%f,0.,%f,%f\n1.0,0.0,0.0\n%f\n',matdata);
fprintf(fileID,'*DAMPING, ALPHA=%f,BETA=%f\n',[AlfaDamping,BetaDamping]);
fprintf(fileID,'*ELSET,ELSET=PONTOONS\npontoon4\n');
fprintf(fileID,'*ELSET,ELSET=GravityElement\nPONTOONS');
fclose(fileID);
%% *** Element 8 ***
% Pylon towers up to height 169
Pylon=[];
for i=1:64
    Pylon(i,:)=[element_Tower(i,1:3),node_Tower_Orientation(i)];
end
fileID=fopen('INPUTFILES\PYLON.inp','w');
fprintf(fileID,'');
    Area_test=[];
    I11_test=[];
    I22_test=[];
for i = 1:size(Pylon,1)
fileID=fopen('INPUTFILES\PYLON.inp','a');
    fprintf(fileID,'*ELEMENT, TYPE=B31,ELSET=PYLONeL%d\n',i);
    fprintf(fileID,'%d,%d,%d\n',Pylon(i,1:3));
    Ai=ATower(element_Tower(i,4));
    Aj=ATower(element_Tower(i+1,4));
    I11i=IxxTower(element_Tower(i,4));
    I11j=IxxTower(element_Tower(i+1,4));
    I22i=IyyTower(element_Tower(i,4));
    I22j=IyyTower(element_Tower(i+1,4));
    J=(JTower(element_Tower(i,4))+JTower(element_Tower(i,4)))/2;
    Area=(Ai+sqrt(Ai*Aj)+Aj)/3;

```

```

I11=(I11i+((I11i^3)*I11j)^(1/4)+sqrt(I11i*I11j)+((I11j^3)*I11i)^(1/4)+I11j)
/5;

I22=(I22i+((I22i^3)*I22j)^(1/4)+sqrt(I22i*I22j)+((I22j^3)*I22i)^(1/4)+I22j)
/5;

    %Finding the direction
    n1=Pylon(i,2);
    n2=Pylon(i,4);
for j = 1:size(node,1);
    if n1==node(j,1);
        x1=node(j,2);
        y1=node(j,3);
        z1=node(j,4);
    elseif n2==node(j,1);
        x2=node(j,2);
        y2=node(j,3);
        z2=node(j,4);
    end
end
xcomp=x2-x1;
ycomp=y2-y1;
zcomp=z2-z1;
matdata=[Concrete.Material.Density,i,Concrete.Material.Poisson_ratio,Area,I
11,I22,J,xcomp,ycomp,zcomp,Concrete.Material.E_modulus];
fprintf(fileID,'*BEAM GENERAL SECTION, DENSITY=%f, ELSET=PYLONel%d,
POISSON=%f\n%f,%f,0.,%f,%f\n%f,%f,%f\n%f\n',matdata);
fprintf(fileID,'*DAMPING, ALPHA=%f,BETA=%f\n',[AlfaDamping,BetaDamping]);
fprintf(fileID,'*ELSET,ELSET=CABLETOWER\nPYLONel%d\n',i);
end
fprintf(fileID,'***');
fclose(fileID);
%% *** Element 9 ***
% Tower horizontal beam
towerhor=[];
for i=88:99
    towerhor=[towerhor;element_Tower(i,1:3),node_Tower_Orientation(i,1)];
end
fileID=fopen('INPUTFILES\towerhor.inp','w');
for i = 1:size(towerhor,1)
fprintf(fileID,'*ELEMENT,TYPE=B31,ELSET=towerhor%d\n',i);
line=towerhor(i,1:3);
fprintf(fileID,'%d,%d,%d\n',line);
n1=towerhor(i,2);
n2=towerhor(i,4);
for j = 1:size(node,1);
    if n1==node(j,1);
        x1=node(j,2);
        y1=node(j,3);
        z1=node(j,4);
    elseif n2==node(j,1);
        x2=node(j,2);
        y2=node(j,3);
        z2=node(j,4);
    end
end
xcomp=x2-x1;
ycomp=y2-y1;
zcomp=z2-z1;

```

```

matdata=[Concrete.Material.Density,i,Concrete.Material.Poisson_ratio,Concrete
te.Section2.Area,Concrete.Section2.I11,Concrete.Section2.I22,Concrete.Secti
on2.J,xcomp,ycomp,zcomp,Concrete.Material.E_modulus];
fprintf(fileID,'*BEAM GENERAL SECTION, DENSITY=%f, ELSET=towerhor%d,
POISSON=%f\n%f,%f,0.,%f,%f\n%f,%f,%f\n%f\n',matdata);
fprintf(fileID,'*DAMPING, ALPHA=%f,BETA=%f\n',[AlfaDamping,BetaDamping]);
fprintf(fileID,'*ELSET,ELSET=CABLETOWER\ntowerhor%d\n',i);
fprintf(fileID,'*ELSET,ELSET=GravityElement\ntowerhor%d\n',i);
end
fprintf(fileID,'**');
fclose(fileID);
%% *** Element 11 ***
% Tower constant cross section at top
PYLONtop=[];
for i=65:87
PYLONtop=[PYLONtop;element_Tower(i,1:3),node_Tower_Orientation(i)];
end
fileID=fopen('INPUTFILES\PYLONtop.inp','w');
for i = 1:size(PYLONtop,1)
fprintf(fileID,'*ELEMENT,TYPE=B31,ELSET=PYLONtop\n');
line=PYLONtop(i,1:3);
fprintf(fileID,'%d,%d,%d\n',line);
n1=PYLONtop(i,2);
n2=PYLONtop(i,4);
for j = 1:size(node,1);
    if n1==node(j,1);
        x1=node(j,2);
        y1=node(j,3);
        z1=node(j,4);
    elseif n2==node(j,1);
        x2=node(j,2);
        y2=node(j,3);
        z2=node(j,4);
    end
end
xcomp=x2-x1;
ycomp=y2-y1;
zcomp=z2-z1;
matdata=[Concrete.Material.Density,Concrete.Material.Poisson_ratio,Concrete
.Section1.Area,Concrete.Section1.I11,Concrete.Section1.I22,Concrete.Section
1.J,xcomp,ycomp,zcomp,Concrete.Material.E_modulus];
fprintf(fileID,'*BEAM GENERAL SECTION, DENSITY=%f, ELSET=PYLONtop,
POISSON=%f\n%f,%f,0.,%f,%f\n%f,%f,%f\n%f\n',matdata);
fprintf(fileID,'*DAMPING, ALPHA=%f,BETA=%f\n',[AlfaDamping,BetaDamping]);
fprintf(fileID,'*ELSET,ELSET=CABLETOWER\nPYLONtop\n');
fprintf(fileID,'*DAMPING\n%f,%f\n',[AlfaDamping,BetaDamping]);
end
fprintf(fileID,'*ELSET,ELSET=GravityElement\nCABLETOWER\n');
fprintf(fileID,'**');
fclose(fileID);
%% *** Element 12 ***
% Point mass elements for pontoons
pontonmass=[element_Pontoon_Mass];
fileID=fopen('INPUTFILES\PontoonWeight.inp','w');
for i = 1:size(pontonmass,1)
    fprintf(fileID,'*ELEMENT,TYPE=MASS,ELSET=PONTOON_MASS%d\n',[i]);
line=pontonmass(i,1:2);
fprintf(fileID,'%d,%d\n',line);
fprintf(fileID,'*MASS,ELSET=PONTOON_MASS%d,ALPHA=%f\n',[i,AlfaDamping]);
fprintf(fileID,'%f\n',pontonmass(i,3));
fprintf(fileID,'*ELSET,ELSET=GravityElement\nPONTOON_MASS%d\n',i);

```

```

end
fprintf(fileID, '**');
fclose(fileID);
%% Element 13
% Point mass elements for reinforced sections at axis
fileID=fopen('INPUTFILES\girderreinforcemass.inp','w');
fprintf(fileID, '**THIS IS ADDED MASS AT AXIS 3:48 due reinforcments at
pontoon connections\n');
for i = 1:size(element_Girder_Reinforced_Mass,1)

fprintf(fileID, '*ELEMENT,TYPE=MASS,ELSET=GIRDER_REINFORCEMENT_MASS%d\n',i);
line=element_Girder_Reinforced_Mass(i,1:2);
fprintf(fileID, '%d,%d\n',line);
fprintf(fileID, '*MASS,ELSET=GIRDER_REINFORCEMENT_MASS%d,ALPHA=%f\n',[i,Alfa
Damping]);
fprintf(fileID, '%f\n',element_Girder_Reinforced_Mass(i,3));
fprintf(fileID, '*ELSET,ELSET=GravityElement\nGIRDER_REINFORCEMENT_MASS%d\n'
,i);
end
fprintf(fileID, '**');
fclose(fileID);
%% Defining the added mass of the pontoons
Added_Mass=[];
for i=1:46
    e1=node_Pontoon_CB(i);
    if i >=Pontoon.Section1.Positions(1) &&
i<=Pontoon.Section1.Positions(size(Pontoon.Section1.Positions,2)) %11-46
        e2=0.50e7;
        e3=3.0e5;
        e4=3.0e6;
    elseif i >=Pontoon.Section2.Positions(1) &&
i<=Pontoon.Section2.Positions(size(Pontoon.Section2.Positions,2)) %5-10
        e2=0.66e7;
        e3=3.9e5;
        e4=3.0e6;
    elseif i >=Pontoon.Section3.Positions(1) &&
i<=Pontoon.Section3.Positions(size(Pontoon.Section3.Positions,2)) %2-4
        e2=0.85e7;
        e3=4.6e5;
        e4=3.0e6;
    elseif i >=Pontoon.Section4.Positions(1) &&
i<=Pontoon.Section4.Positions(size(Pontoon.Section4.Positions,2)) %1
        e2=1.08e7; % HEAVE 33
        e3=5.6e5; % SURGE 11
        e4=3.0e6; %SWAY 22
    end
    e5=i;
    Added_Mass=[Added_Mass;e1,e2,e3,e4,e5]; %
end
fileID=fopen('INPUTFILES\PontoonAddedMass.inp','w');
for i=1:46

%fprintf(fileID, '*ORIENTATION,NAME=ORIMASS%d\n%f,%f,%f,%f,%f,%f,%f,%f\n'
,[i,node_Pontoon_CB(i,2)+cos(alfa_Pontoon(i)),node_Pontoon_CB(i,3)+sin(alfa
_Pontoon(i)),node_Pontoon_CB(i,4),node_Pontoon_CB(i,2)+cos(alfa_Pontoon(i)+
3.14/2),node_Pontoon_CB(i,3)+sin(alfa_Pontoon(i)+3.14/2),node_Pontoon_CB(i,
4),node_Pontoon_CB(i,2),node_Pontoon_CB(i,3),node_Pontoon_CB(i,4)]);
    fprintf(fileID, '*ELEMENT,TYPE=MASS,ELSET=PONTOONADDEDMASS%d\n',[i]);
    fprintf(fileID, '%d,%d\n',[20100+i,node_Pontoon_CB(i,1)]);

```

```

fprintf(fileID, '*MASS, ELSET=PONTOONADDEDMASS%d, type=ANISOTROPIC, orientation
=ORISPRING%d\n', [i], [47-i]); %

fprintf(fileID, '%f,%f,%f\n', [Added_Mass(i,3), Added_Mass(i,4), Added_Mass(i,2
)]);
end
fprintf(fileID, '**');
fclose(fileID);
%% Creating nodes to apply wind loading and damping
% Defining applicable elements to be loaded
U=importdata('U.mat');
element_Girder=[girder01;girder02;girder03];
element_Load=[element_Girder;element_Cable;element_Pontoon;element_Tower];
numbel=[size(element_Girder,1),size(element_Cable,1)+size(element_Girder,1)
,size(element_Pontoon,1)+size(element_Cable,1)+size(element_Girder,1),size(
element_Tower,1)+size(element_Pontoon,1)+size(element_Cable,1)+size(element
_Girder,1)];
dashpot.girder.coefxy=[];
dashpot.girder.coefz=[];
dashpot.girder.node1=[];
dashpot.girder.node2=[];
dashpot.pontoon.coef=[];
dashpot.pontoon.node=[];
ip=0;
% Elements and their orientation
geometry.element.X=[];
geometry.element.Y=[];
geometry.element.Z=[];
geometry.element.ID=[];
geometry.element.node1=[];
geometry.element.node2=[];
geometry.element.AxCd=[]; % Perpendicular to orientation
geometry.element.AyCd=[]; % same direction as orientation
geometry.element.AwCl=[]; % Upwards
geometry.element.AmCm=[]; % Around the axis which is in the same
direction as the orientation
geometry.element.type=[];
Orientation = [];
for i = 1:size(element_Load,1)
    geometry.element.ID(i)=element_Load(i,1);
    for j=1:size(node,1)
        if node(j,1) == element_Load(i,2)
            node1=node(j,2:4);
            nodenumb1=node(j,1);
        elseif node(j,1) == element_Load(i,3)
            node2=node(j,2:4);
            nodenumb2=node(j,1);
        end
    end
    N = sqrt((node1(2)-node2(2))^2+(node1(1)-node2(1))^2);
    if N>0
        diffY = (node1(2)-node2(2))/N;
        diffX = (node1(1)-node2(1))/N;
        Orientation(i) = 180/pi.*atan2(diffX,diffY);
    else
        Orientation(i) = 0;
    end
    geometry.element.X(i) = (node1(1)+node2(1))/2;
    geometry.element.Y(i) = (node1(2)+node2(2))/2;
    geometry.element.Z(i) = (node1(3)+node2(3))/2;
end

```



```

geometry.element.node1(i)=nodenumb1(1);
geometry.element.node2(i)=nodenumb2(1);
if i<=nubel(1)
    % GIRDER ELEMENTS
    for j=1:size(node,1)
        if node(j,1) == element_Girder(i,2)
            node1=node(j,1:4);
        elseif node(j,1) == element_Girder(i,3)
            node2=node(j,1:4);
        end
    end
    dx=node1(2)-node2(2);dy=node1(3)-node2(3);dz=node1(4)-
node2(4);
    L=sqrt(dx^2+dy^2);

geometry.element.AxCd(i)=rho*Girder.Section1.Drag_factor*Girder.Section1.Ma
x_height; % NORMAL - vx - Drag

geometry.element.AmCm(i)=rho*L*Girder.Section1.Moment_factor*Girder.Section
1.Max_height^2;% NORMAL - vx - Moment

geometry.element.AwC1(i)=rho*Girder.Section1.Lift_factor*Girder.Section1.Wi
dth; % VERTICAL NORMAL - w - LIFT
    geometry.element.type(i)=1;
        %Fd=1/2 p v^2 Cd A
    dashpot.girder.coefxy(i)=geometry.element.AxCd(i)*L;

dashpot.girder.coefx(i)=abs(dashpot.girder.coefxy(i)*cos(Orientation(i)*pi/
180));

dashpot.girder.coefy(i)=abs(dashpot.girder.coefxy(i)*sin(Orientation(i)*pi/
180));

dashpot.girder.coefz(i)=rho*0.5*(Girder.Section1.Lift_factor*Girder.Section
1.Width+Girder.Section1.Drag_factor*Girder.Section1.Max_height)*L;

dashpot.girder.coefrot(i)=geometry.element.AmCm(i)*Girder.Section1.Max_heig
ht*RedFact/8;
    dashpot.girder.node1(i)=geometry.element.node1(1);
    dashpot.girder.node2(i)=geometry.element.node2(1);
    elseif i>nubel(1) && i<nubel(2)
        % CABLE ELEMENTS
        cablenumb=i-nubel(1);
        for j=1:size(node,1)
            if node(j,1) == element_Cable(cablenumb,2)
                node1=node(j,1:4);
            elseif node(j,1) == element_Cable(cablenumb,3)
                node2=node(j,1:4);
            end
        end
        dx=node1(2)-node2(2);dy=node1(3)-node2(3);dz=node1(4)-
node2(4);
        L=sqrt(dx^2+dy^2+dz^2);

geometry.element.AxCd(i)=rho*L*Cable.Section.Drag_factor*2*sqrt(Cable_Area(
cablenumb)/3.14); % NORMAL - vx - Drag

geometry.element.AwC1(i)=rho*L*Cable.Section.Drag_factor*2*sqrt(Cable_Area(
cablenumb)/3.14); % VERTICAL NORMAL - w - LIFT
    geometry.element.type(i)=2;
    elseif i>nubel(2) && i<nubel(3)

```

```

% PONTOON COLOUMNS
ip=ip+1;
pontoonnumb=i+1- numb(2);
pontoonID=element_Pontoon(pontoonnumb,4);
    if pontoonID==1
        type=4;
        diameter=Pontoon.Section4.Diameter;
    elseif pontoonID>1 && pontoonID<5
        type=3;
        diameter=Pontoon.Section3.Diameter;
    elseif pontoonID>4 && pontoonID<11
        type=2;
        diameter=Pontoon.Section2.Diameter;
    else
        type=1;
        diameter=Pontoon.Section1.Diameter;
    end
    for j=1:size(node,1)
        if node(j,1) == element_Pontoon(pontoonnumb,2)
            node1=node(j,1:4);
        elseif node(j,1) == element_Pontoon(pontoonnumb,3)
            node2=node(j,1:4);
        end
    end
    dragfactor=Pontoon.Section1.Drag_factor;
    dx=node1(2)-node2(2);dy=node1(3)-node2(3);dz=node1(4)-
node2(4);
    L=sqrt(dx^2+dy^2+dz^2);
    geometry.element.AxCd(i)=rho*dragfactor*diameter; % - v -
Direction given by the wind.
    geometry.element.AyCd(i)=rho*dragfactor*diameter; % - u -
    geometry.element.type(i)=3;
    dashpot.pontoon.coef(ip)=0.5*geometry.element.AxCd(i)*L*U(i);
    dashpot.pontoon.node(ip)=node1(1);
elseif i>numbel(3)
% TOWER
Orientation(i)=alfa_tower*180/pi;
towernumb=i- numb(3);
for j=1:size(node,1)
    if node(j,1) == element_Tower(towernumb,2)
        node1=node(j,1:4);
    elseif node(j,1) == element_Tower(towernumb,3)
        node2=node(j,1:4);
    end
end
dx=node1(2)-node2(2);dy=node1(3)-node2(3);dz=node1(4)-
node2(4);
Lxyz=sqrt(dx^2+dy^2+dz^2);
Lz=dz;
% Defining dimensions
if element_Tower(towernumb,4)>0 % ALL NON VERTICAL ELEMENTS
    Avx = Lz*h(element_Tower(towernumb,4));% FROM THE SIDE
    Avy = Lz*b(element_Tower(towernumb,4)); % FROM "AXIAL"
elseif element_Tower(towernumb,4) == 0
    Avx = 0; % NO LOAD FROM THE SIDE
    Avy = Lxyz*9.5; % FROM " AXIAL"
end
geometry.element.AxCd(i)=rho*abs(Concrete.Section.Cd*Avx);
geometry.element.AyCd(i)=rho*abs(Concrete.Section.Cd*Avy);
geometry.element.AmCm(i)=0;
geometry.element.AwCl(i)=0;

```

```

        geometry.element.type(i)=4;
    end
end
save('geometry.mat','geometry');
save('Orientation.mat','Orientation');
save('numbel.mat','numbel');
%% Creating wind damper dashpots
U=importdata('U.mat');
fileID=fopen('INPUTFILES\AerodynamicDamper.inp','w');
c=1;
GDP=3675*c;
GDPz=16275*c;
GDPr=46894*c;
PDPx=PDT*290000; %%Pontoons are damp tuned according ot reports
PDPy=PDT*80000;
for i = 1:size(dashpot.girder.nodel,2)
    elnumbl=19000+i;
    fprintf(fileID,'*ELEMENT,TYPE=DASHPOT1,ELSET=GDashPot%d\n',i);
    fprintf(fileID,'%d,%d\n',[elnumbl,dashpot.girder.nodel(i)]);
    fprintf(fileID,'*DASHPOT,ELSET=GDashPot%d\n1\n',i);
    fprintf(fileID,'%f\n',DPC*GDP*abs(cos(Orientation(i)*pi/180)));
    fprintf(fileID,'*DASHPOT,ELSET=GDashPot%d\n2\n',i);
    fprintf(fileID,'%f\n',DPC*GDP*abs(sin(Orientation(i)*pi/180)));
    fprintf(fileID,'*DASHPOT,ELSET=GDashPot%d\n3\n',i);
    fprintf(fileID,'%f\n',DPC*GDPz);
    %   fprintf(fileID,'*DASHPOT,ELSET=DashPot%d\n4\n',i);
    %   fprintf(fileID,'%f\n',GDPr*abs(cos(Orientation(i)*pi/180)));
    %   fprintf(fileID,'*DASHPOT,ELSET=DashPot%d\n5\n',i);
    %   fprintf(fileID,'%f\n',GDPr*abs(sin(Orientation(i)*pi/180)));
    fprintf(fileID,'*ELSET,ELSET=GirderDASHPOT\nGDashPot%d\n',i);
end
for i = 1:size(dashpot.pontoon.node,2)
    elnumbl=elnumbl+1;
    fprintf(fileID,'*ELEMENT,TYPE=DASHPOT1,ELSET=CDashPot%d\n',i);
    fprintf(fileID,'%d,%d\n',[elnumbl,dashpot.pontoon.node(i)]);
    fprintf(fileID,'*DASHPOT,ELSET=CDashPot%d\n1\n',i);
    fprintf(fileID,'%f\n',DPC*dashpot.pontoon.coef(i));
    fprintf(fileID,'*DASHPOT,ELSET=CDashPot%d\n2\n',i);
    fprintf(fileID,'%f\n',DPC*dashpot.pontoon.coef(i));
    fprintf(fileID,'*ELSET,ELSET=PontoonDASHPOT\nCDashPot%d\n',i);
end
fprintf(fileID,'**');
fclose(fileID);
fileID=fopen('INPUTFILES\HydrodynamicDamper.inp','w');
for i=1:46
    if i >=Pontoon.Section1.Positions(1) &&
i<=Pontoon.Section1.Positions(size(Pontoon.Section1.Positions,2))
        X=Pontoon.Section1.Drag_coef_x_drag_area_surge_submerged*1000;
%front
        Y=Pontoon.Section1.Drag_coef_x_drag_area_sway_submerged*1000; %side
        orientation=alfa_Pontoon(47-i);
    elseif i >=Pontoon.Section2.Positions(1) &&
i<=Pontoon.Section2.Positions(size(Pontoon.Section2.Positions,2))
        X=Pontoon.Section1.Drag_coef_x_drag_area_surge_submerged*1000;
        Y=Pontoon.Section1.Drag_coef_x_drag_area_sway_submerged*1000;
        orientation=alfa_Pontoon(47-i);
    elseif i >=Pontoon.Section3.Positions(1) &&
i<=Pontoon.Section3.Positions(size(Pontoon.Section3.Positions,2))
        X=Pontoon.Section1.Drag_coef_x_drag_area_surge_submerged*1000;
        Y=Pontoon.Section1.Drag_coef_x_drag_area_sway_submerged*1000;
        orientation=alfa_Pontoon(47-i);
end

```

```

elseif i >=Pontoon.Section4.Positions(1) &&
i<=Pontoon.Section4.Positions(size(Pontoon.Section4.Positions,2))
    X=Pontoon.Section1.Drag_coef_x_drag_area_surge_submerged*1000;
    Y=Pontoon.Section1.Drag_coef_x_drag_area_sway_submerged*1000;
    orientation=alfa_Pontoon(47-i);
end
Z=928000;
PDPx=PDT*X*abs(cos(orientation))+PDT*Y*abs(sin(orientation));
PDPy=PDT*X*abs(sin(orientation))+PDT*Y*abs(cos(orientation));
PDPz=PDT*Z;
elnumb2=200000+i;
fprintf(fileID, '*ELEMENT,TYPE=DASHPOT1,ELSET=PDashPot%d\n',i);
fprintf(fileID, '%d,%d\n',[elnumb2,node_Pontoon_CG(i)]);
fprintf(fileID, '*DASHPOT,ELSET=PDashPot%d\n1\n',i);
fprintf(fileID, '%f\n',PDPx);
fprintf(fileID, '*DASHPOT,ELSET=PDashPot%d\n2\n',i);
fprintf(fileID, '%f\n',PDPy);
fprintf(fileID, '*DASHPOT,ELSET=PDashPot%d\n3\n',i);
fprintf(fileID, '%f\n',PDPz);
fprintf(fileID, '*ELSET,ELSET=PDASHPOT\nPDashPot%d\n',i);
end
fclose(fileID);

```

A.3 MATLAB script wind loads

The wind 4D simulation code is not included here, only the file which processes the turbulent wind field which is simulated.

```
% Create Wind Loads

vx=importdata('vx.mat');
vy=importdata('vy.mat');
u=importdata('u.mat');
v=importdata('v.mat');
w=importdata('w.mat');
t=importdata('t.mat');
windDirection=importdata('windDirection.mat');
fs=importdata('fs.mat');
TimeLength=importdata('TimeLength.mat');
geometry=importdata('geometry.mat');
Orientation=importdata('Orientation.mat');
numbel=importdata('numbel.mat');

%numbel=[size(element_Girder,1),size(element_Cable,1)+size(element_Girder,1),size(element_Pontoon,1)+size(element_Cable,1)+size(element_Girder,1),size(element_Tower,1)+size(element_Pontoon,1)+size(element_Cable,1)+size(element_Girder,1)];

if TimeLength >= size(u,1)
TimeLength = size(u,1)
end

steps=TimeLength;

girder.FX=[];
girder.FY=[];
girder.FZ=[];
girder.MX=[];
girder.MY=[];

i_cable=0;
cable.FX=[];
cable.FY=[];
cable.FZ=[];

i_pontoon=0;
pontoon.FX=[];
pontoon.FY=[];

i_tower=0;
tower.FX=[];
tower.FY=[];

for i = 1:size(geometry.element.X,2)
    for T=1:steps
        if geometry.element.type(i) == 1 % GIRDER ELEMENT
            ii=i;
```

```

girder.FX(T,ii)=geometry.element.AxCd(i)*0.5*(vx(T,i)*sin(Orientation(i)*pi/180))*abs(vx(T,i)*sin(Orientation(i)*pi/180));

girder.FY(T,ii)=geometry.element.AxCd(i)*0.5*(vx(T,i)*cos(Orientation(i)*pi/180))*abs(vx(T,i)*cos(Orientation(i)*pi/180));
girder.FZ(T,ii)=geometry.element.AwCl(i)*0.5*w(T,i)*abs(w(T,i));
girder.MX(T,ii)=geometry.element.AmCm(i)*0.5*(w(T,i)*sin(pi-(pi/2)-Orientation(i)*pi/180))*abs(w(T,i)*sin(pi-(pi/2)-Orientation(i)*pi/180));
girder.MY(T,ii)=geometry.element.AmCm(i)*0.5*(w(T,i)*cos(pi-(pi/2)-Orientation(i)*pi/180))*abs(w(T,i)*cos(pi-(pi/2)-Orientation(i)*pi/180));
girder.node1(ii)=geometry.element.node1(i);
girder.node2(ii)=geometry.element.node2(i);
girder.element(ii)=geometry.element.ID(i);

elseif geometry.element.type(i) == 2 % CABLE ELEMENT
ii=i-numbel(1);

cable.FX(T,ii)=geometry.element.AxCd(i)*0.5*(vx(T,i)*sin(Orientation(i)*pi/180))*abs(vx(T,i)*sin(Orientation(i)*pi/180));

cable.FY(T,ii)=geometry.element.AxCd(i)*0.5*(vx(T,i)*cos(Orientation(i)*pi/180))*abs(vx(T,i)*cos(Orientation(i)*pi/180));
cable.FZ(T,ii)=geometry.element.AwCl(i)*0.5*(w(T,i))*abs(w(T,i));
cable.node1(ii)=geometry.element.node1(i);
cable.node2(ii)=geometry.element.node2(i);
cable.element(ii)=geometry.element.ID(i);

elseif geometry.element.type(i) == 3 % PONTOON ELEMENT
ii=i-numbel(2);
uxx=-u(T,i)*cos(windDirection*pi/180);
uyy=u(T,i)*sin(windDirection*pi/180);
vxx=-v(T,i)*cos(pi-windDirection*pi/180);
vyy=-v(T,i)*sin(pi-windDirection*pi/180);

pontoon.FX(T,ii)=geometry.element.AxCd(i)*0.5*(uxx+vxx)*abs(uxx+vxx);

pontoon.FY(T,ii)=geometry.element.AxCd(i)*0.5*(uyy+vyy)*abs(uyy+vyy);
pontoon.node1(ii)=geometry.element.node1(i);
pontoon.node2(ii)=geometry.element.node2(i);
pontoon.element(ii)=geometry.element.ID(i);
elseif geometry.element.type(i) == 4 % TOWER ELEMENT
ii=i-numbel(3);
VxLoad=geometry.element.AxCd(i)*0.5*vx(T,i)*abs(vx(T,i));
VyLoad=geometry.element.AyCd(i)*0.5*vx(T,i)*abs(vx(T,i));
tower.FX(T,ii)=VxLoad*cos(Orientation(i)*pi/180)+VyLoad*sin(Orientation(i)*pi/180);

tower.FY(T,ii)=VxLoad*sin(Orientation(i)*pi/180)+VyLoad*cos(Orientation(i)*pi/180);
tower.node1(ii)=geometry.element.node1(i);
tower.node2(ii)=geometry.element.node2(i);
tower.element(ii)=geometry.element.ID(i);

```

```

end
end

```

```

end

%% WRITING INPUT FILES

fileID=fopen('INPUTFILES\dynamicstep.inp','w');
fprintf(fileID,'%f,%f,1e-015,%f\n',[1/fs,t(TimeLength),1/fs]);
fclose(fileID);
%% GIRDER INPUT

fileID=fopen('INPUTFILES\WindLoadGirder.inp','w');

for i = 1:size(girder.FX,2)
    fprintf(fileID,'*DLOAD,OP=NEW,AMPLITUDE=girderFX%d\n',i);
    fprintf(fileID,'%d,PX,1\n',girder.element(i));
    fprintf(fileID,'*AMPLITUDE, NAME=girderFX%d\n',i);

    fprintf(fileID,'%f,%f\n',transpose([t(1:TimeLength),girder.FX(1:TimeLength,
i)]));
end

for i = 1:size(girder.FY,2)
    fprintf(fileID,'*DLOAD,OP=NEW,AMPLITUDE=girderFY%d\n',i);
    fprintf(fileID,'%d,PY,1\n',girder.element(i));
    fprintf(fileID,'*AMPLITUDE, NAME=girderFY%d\n',i);

    fprintf(fileID,'%f,%f\n',transpose([t(1:TimeLength),girder.FY(1:TimeLength,
i)]));
end

for i = 1:size(girder.FZ,2)
    fprintf(fileID,'*DLOAD,OP=NEW,AMPLITUDE=girderFZ%d\n',i);
    fprintf(fileID,'%d,PZ,1\n',girder.element(i));
    fprintf(fileID,'*AMPLITUDE, NAME=girderFZ%d\n',i);

    fprintf(fileID,'%f,%f\n',transpose([t(1:TimeLength),girder.FZ(1:TimeLength,
i)]));
end

for i = 1:size(girder.MX,2)
    fprintf(fileID,'*CLOAD,OP=NEW,AMPLITUDE=girderMX%d\n',i);
    fprintf(fileID,'%d,4,0.5\n',girder.node1(i));
    fprintf(fileID,'%d,4,0.5\n',girder.node2(i));
    fprintf(fileID,'*AMPLITUDE, NAME=girderMX%d\n',i);

    fprintf(fileID,'%f,%f\n',transpose([t(1:TimeLength),girder.MX(1:1:TimeLength,
i)]));
end

for i = 1:size(girder.MY,2)
    fprintf(fileID,'*CLOAD,OP=NEW,AMPLITUDE=girderMY%d\n',i);
    fprintf(fileID,'%d,5,0.5\n',girder.node1(i));
    fprintf(fileID,'%d,5,0.5\n',girder.node2(i));
    fprintf(fileID,'*AMPLITUDE, NAME=girderMY%d\n',i);

    fprintf(fileID,'%f,%f\n',transpose([t(1:TimeLength),girder.MY(1:TimeLength,
i)]));
end

fclose(fileID);
%% CABLE INPUT
fileID=fopen('INPUTFILES\WindLoadCable.inp','w');
for i = 1:size(cable.FX,2)
    fprintf(fileID,'*DLOAD,OP=NEW,AMPLITUDE=cableFX%d\n',i);

```

```

    fprintf(fileID, '%d, PX, 1\n', cable.element(i));
    fprintf(fileID, '*AMPLITUDE, NAME=cableFX%d\n', i);

fprintf(fileID, '%f,%f\n', transpose([t(1:TimeLength), cable.FX(1:TimeLength, i)
]));
end
for i = 1:size(cable.FY,2)
    fprintf(fileID, '*DLOAD, OP=NEW, AMPLITUDE=cableFY%d\n', i);
    fprintf(fileID, '%d, PY, 1\n', cable.element(i));
    fprintf(fileID, '*AMPLITUDE, NAME=cableFY%d\n', i);

fprintf(fileID, '%f,%f\n', transpose([t(1:TimeLength), cable.FY(1:TimeLength, i)
]));
end
fclose(fileID);
%% PONTOON INPUT
fileID=fopen('INPUTFILES\WindLoadPontoon.inp', 'w');
for i = 1:size(pontoon.FX,2)
    fprintf(fileID, '*DLOAD, OP=NEW, AMPLITUDE=pontoonFX%d\n', i);
    fprintf(fileID, '%d, PX, 1\n', pontoon.element(i));
    fprintf(fileID, '*AMPLITUDE, NAME=pontoonFX%d\n', i);

fprintf(fileID, '%f,%f\n', transpose([t(1:TimeLength), pontoon.FX(1:TimeLength, i)
]));
end
for i = 1:size(pontoon.FY,2)
    fprintf(fileID, '*DLOAD, OP=NEW, AMPLITUDE=pontoonFY%d\n', i);
    fprintf(fileID, '%d, PY, 1\n', pontoon.element(i));
    fprintf(fileID, '*AMPLITUDE, NAME=pontoonFY%d\n', i);

fprintf(fileID, '%f,%f\n', transpose([t(1:TimeLength), pontoon.FY(1:TimeLength, i)
]));
end
fclose(fileID);
%% TOWER INPUT
fileID=fopen('INPUTFILES\WindLoadTower.inp', 'w');
for i = 1:size(tower.FX,2)
    fprintf(fileID, '*DLOAD, OP=NEW, AMPLITUDE=towerFX%d\n', i);
    fprintf(fileID, '%d, PX, 1\n', tower.element(i));
    fprintf(fileID, '*AMPLITUDE, NAME=towerFX%d\n', i);

fprintf(fileID, '%f,%f\n', transpose([t(1:TimeLength), tower.FX(1:TimeLength, i)
]));
end
for i = 1:size(tower.FY,2)
    fprintf(fileID, '*DLOAD, OP=NEW, AMPLITUDE=towerFY%d\n', i);
    fprintf(fileID, '%d, PY, 1\n', tower.element(i));
    fprintf(fileID, '*AMPLITUDE, NAME=towerFY%d\n', i);

fprintf(fileID, '%f,%f\n', transpose([t(1:TimeLength), tower.FY(1:TimeLength, i)
]));
end

fclose(fileID);

%% CREATING STATIC WIND LOADS FOR FIRST STEP i.e. 1 time increment

```



```

T=1;
t(T)=1; % 1 second lenght

% GIRDER INPUT
fileID=fopen('INPUTFILES\StaticWindLoad.inp','w');
fprintf(fileID,'*DLOAD\n');
for i = 1:size(girder.FX,2)
    fprintf(fileID,'%d,PX,%f\n',[girder.element(i),girder.FX(T,i)]);
end
for i = 1:size(girder.FY,2)
    fprintf(fileID,'%d,PY,%f\n',[girder.element(i),girder.FY(T,i)]);

end
for i = 1:size(girder.FZ,2)
    fprintf(fileID,'%d,PZ,%f\n',[girder.element(i),girder.FZ(T,i)]);
end
fprintf(fileID,'*CLOAD\n');
for i = 1:size(girder.MX,2)
    fprintf(fileID,'%d,4,%f\n',[girder.node1(i),0.5*girder.MX(T,i)]);
    fprintf(fileID,'%d,4,%f\n',[girder.node2(i),0.5*girder.MX(T,i)]);
end
for i = 1:size(girder.MY,2)
    fprintf(fileID,'%d,5,%f\n',[girder.node1(i),0.5*girder.MY(T,i)]);
    fprintf(fileID,'%d,5,%f\n',[girder.node2(i),0.5*girder.MY(T,i)]);
end
fprintf(fileID,'*DLOAD\n');
% CABLE INPUT
for i = 1:size(cable.FX,2)
    fprintf(fileID,'%d,PX,%f\n',[cable.element(i),cable.FX(T,i)]);
end
for i = 1:size(cable.FY,2)
    fprintf(fileID,'%d,PY,%f\n',[cable.element(i),cable.FY(T,i)]);

end
% PONTOON INPUT
for i = 1:size(pontoon.FX,2)
    fprintf(fileID,'%d,PX,%f\n',[pontoon.element(i),pontoon.FX(T,i)]);
end
for i = 1:size(pontoon.FY,2)
    fprintf(fileID,'%d,PY,%f\n',[pontoon.element(i),pontoon.FY(T,i)]);
end
% TOWER INPUT
for i = 1:size(tower.FX,2)
    fprintf(fileID,'%d,PX,%f\n',[tower.element(i),tower.FX(T,i)]);
end
for i = 1:size(tower.FY,2)
    fprintf(fileID,'%d,PY,%f\n',[tower.element(i),tower.FY(T,i)]);
end
fprintf(fileID,'**');
fclose(fileID);

```