Department of Structural Engineering
Faculty of Engineering
NTNU – Norwegian University of Science and Technology

ACCESSIBILITY
Open

MASTER THESIS 2021

SUBJECT AREA:	DATE:	NO. OF PAGES:
Structural dynamics	09.06.2021	172 (Including appendix)

TITLE:

Implementation of seismic soil-structure interaction in OpenFAST and application to a 10MW offshore wind turbine on jacket structure

Implementasjon av seismisk jord-struktur interaksjon i OpenFAST og applikasjon på en 10MW havvindturbin på jacket konstruksjon

BY:

Daniel Martens Pedersen and Henrik Askheim

SUMMARY:
The planned offshore wind farm developments in areas prone to seismic action, such as Taiwan, China, Japan and North America, has made the industry question the performance of offshore wind turbine (OWT) foundations due to earthquake loading. The most common and cost-effective foundation solution is the monopile foundation, which has been developed and well tested over the last three decades in the less seismic active areas of Northern Europe. A piled jacket structure has been purposed as an alternative solution, and has been shown to perform well in terms of handling the overturning moments at the structure base. However, further research is needed to fully understand the behaviour of the jacket foundation during seismic action, and adequate numerical models including the soil-structure interaction (SSI) effects are required.

One of the challenges in design of OWTs is that the analyses are performed using specialized software dedicated to hydro-aero-servo-elasto-dynamic analyses which often cannot perform seismic SSI analyses rigorously. This thesis presents a methodology to extend these tools to include seismic SSI analyses in the open source OWT analysis tool OpenFAST. The developed tool is then applied to an offshore wind turbine on a jacket structure founded on piles. The SSI is implemented using a multi-step method. The method provides the SSI stiffness and kinematic interaction on basis of superpositioning, thus, limiting the analysis strictly speaking to linear effects. The jacket base is attached to linear elastic springs, and excited by forces calculated from the pile-head motions during the earthquake. The spring stiffness and pile-head motions are obtained from a complementary integrated model made in the finite element program Abaqus. The motions are obtained after exciting the soil domain with a massless jacket present. The integrated Abaqus model is also used to verify the implementation of the multi-step method in OpenFAST. The approach is verified by comparing the earthquake response in OpenFAST against the Abaqus model. A realistic earthquake motion together with the IEA 10MW reference OWT on the INNWIND reference jacket are used in the verification.

Using the developed model, the thesis then attempts to investigate some of the characteristic earthquake responses of the
OWT structure. Simulations show how the top of tower displacements are dominated by the wind-induced forces during production form the rotor-nacelle-assembly, while the tower top accelerations and base overturning moments are dominated by the earthquake-induced loads. Further the Abaqus model is extended to include Mohr-Coulomb plasticity in the soil model, and non-linear earthquake excitation analysis are run. The results reveal how the production force from strong winds can induce permanent tilting of the structure during an earthquake, and how the tilt accumulation is highly dependent on the intensity of the earthquake motion. No environmental loads are included in the Abaqus model.

Since only a temporary reference design is analysed, and structural optimization is outside the scope of this thesis, more authentic model designs should be used to obtain specific numerical values of the behaviour. Yet, the outlined modelling framework could be utilized to further study the jacket structure as a solution to the rising challenges of establishing offshore wind farms in seismic active areas.

RESPONSIBLE TEACHER: Prof. Amir M. Kaynia

SUPERVISOR(S): Prof. Amir M. Kaynia

CARRIED OUT AT: Department of Structural Engineering, NTNU

NTNU
Norwegian University of Science and Technology

Department of Structural Engineering

Master's thesis in Civil and Environmental Engineering

Implementation of seismic soil-structure interaction in OpenFAST and application to a 10MW offshore wind turbine on jacket structure

Authors:
Daniel Martens Pedersen and Henrik Askheim

Supervisor:
Prof. Amir M. Kaynia

June, 2021

Abstract

The planned offshore wind farm developments in areas prone to seismic action, such as Taiwan, China, Japan and North America, has made the industry question the performance of offshore wind turbine (OWT) foundations due to earthquake loading. The most common and cost-effective foundation solution is the monopile foundation, which has been developed and well tested over the last three decades in the less seismic active areas of Northern Europe. A piled jacket structure has been purposed as an alternative solution, and has been shown to perform well in terms of handling the overturning moments at the structure base. However, further research is needed to fully understand the behaviour of the jacket foundation during seismic action, and adequate numerical models including the soil-structure interaction (SSI) effects are required.

One of the challenges in design of OWTs is that the analyses are performed using specialized software dedicated to hydro-aero-servo-elasto-dynamic analyses which often cannot perform seismic SSI analyses rigorously. This thesis presents a methodology to extend these tools to include seismic SSI analyses in the open source OWT analysis tool OpenFAST. The developed tool is then applied to an offshore wind turbine on a jacket structure founded on piles. The SSI is implemented using a multi-step method. The method provides the SSI stiffness and kinematic interaction on basis of superpositioning, thus, limiting the analysis strictly speaking to linear effects. The jacket base is attached to linear elastic springs, and excited by forces calculated from the pile-head motions during the earthquake. The spring stiffness and pile-head motions are obtained from a complementary integrated model made in the finite element program Abaqus. The motions are obtained after exciting the soil domain with a massless jacket present. The integrated Abaqus model is also used to verify the implementation of the multi-step method in OpenFAST. The approach is verified by comparing the earthquake response in OpenFAST against the Abaqus model. A realistic earthquake motion together with the IEA 10MW reference OWT on the INNWIND reference jacket are used in the verification.

Using the developed model, the thesis then attempts to investigate some of the characteristic earthquake responses of the OWT structure. Simulations show how the top of tower displacements are dominated by the wind-induced forces during production form the rotor-nacelle-assembly, while the tower top accelerations and base overturning moments are dominated by the earthquake-induced loads. Further the Abaqus model is extended to include Mohr-Coulomb plasticity in the soil model, and non-linear earthquake excitation analysis are run. The results reveal how the production force from strong winds can induce permanent tilting of the structure during an earthquake, and how the tilt accumulation is highly dependent on the intensity of the earthquake motion. No environmental loads are included in the Abaqus model.

Since only a temporary reference design is analysed, and structural optimization is outside the scope of this thesis, more authentic model designs should be used to obtain specific numerical values of the behaviour. Yet, the outlined modelling framework could be utilized to further study the jacket structure as a solution to the rising challenges of establishing offshore wind farms in seismic active areas.

Sammendrag

Den planlagte utbyggingen av havvindparker i områder utsatt for seismisk aktivitet, som Taiwan, Kina, Japan og Nord-Amerika, har fått industrien til å stille spørsmål ved ytelsen til fundamentene som tidligere har blitt brukt. Den mest vanlige og kostnadseffektive løsningen; monopel, som er utviklet og godt testet de siste tre tiårene i de mindre seismisk aktive områdene i Nord-Europa. En pelet jacket har vært foreslått som en alternativ løsning, som har vist seg å fungere bra når det kommer til å håndtere veltemomentet på sjøbunnen. Samtidig er det nødvendig med ytterligere undersøkelser for å fullt ut forstå oppførselen til jacketen under seismisk aktivitet og tilstrekkelige numeriske modeller som inkluderer interaksjonseffektene mellom jord og konstruksjon.

En av utfordringene i utformingen av havvindkonstruksjoner er at analysene utføres ved hjelp av spesialisert programvare dedikert til hydro-aero-servo-elastisk-dynamiske analyser som ofte ikke kan håndtere interaksjonseffektene mellom jord og konstruksjon på en god nok måte. Denne oppgaven presenterer en metodikk for å utvide den åpne kildekoden til programvaren OpenFAST til å ta hensyn til disse effektene. Det utviklede verktøyet blir deretter brukt på en vindturbin som er plassert på en pelet jacket. Interaksjonseffektene mellom jord og konstruksjon implementeres ved hjelp av en flertrinnsmetode. Metoden angir fjærstivhet og kinematisk interaksjon på grunnlag av superposisjonering, og dermed begrenses analysen strengt tatt til lineære effekter mellom jord og konstruksjon. Bunnen av jacketen er festet til lineært elastiske fjærer, og eksiteres av krefter beregnet fra bevegelsene på toppen av pelene under jordskjelvet. Bevegelsene er oppnådd ved å eksitere en jordmodell, bestående av en gitt jordprofil og peler samt med en masseløs jacket konstruksjon på toppen. Fjærstivhetene og bevegelsene er hentet fra en komplementær modell laget i elementprogrammet Abaqus. Abaqus-modellen brukes også til å verifisere implementeringen av flertrinnsmetoden i OpenFAST. Verifiseringen er gjort ved å bekrefte jordskjelvresponsen fra OpenFAST mot Abaqus-modellen. Et realistisk jordskjelv sammen med en modell av IEA 10MW referansevindturbin på INNWINDs referanse jacket brukes i verifiseringen.

Ved hjelp av den utviklede metoden undersøker oppgaven noen av de karakteristiske jordskjelresponsene til en havvind-konstruksjon. Resultatene viser hvordan forskyvningene i toppen av turbinen domineres av de vindinduserte kreftene under produksjon, mens tårnets akselerasjoner og veltemomenter domineres av belastningene fra jordskjelvet. Videre utvides Abaqus-modellen til å omfatte Mohr-Coulomb-plastisitet i jorden, og det kjøres en ikke-lineær jordskjelvanalyse. Resultatene viser hvordan kreftene fra turbinen under sterk vind kan indusere permanent vipping av konstruksjonen under et jordskjelv, og også hvordan akkumuleringen av permanent vipping er sterkt avhengig av intensiteten til jordskjelvbevegelsen. Under den ikke-lineære analysen er det ikke påført noen andre miljølaster.

Oppgaven har kun tatt for seg et midlertidig referansedesign, og strukturell optimalisering er utenfor omfanget. Mer autentisk modelldesign bør brukes til å oppnå spesifikke numeriske verdier for den jordskjelvinduserte responsen. Likevel kan det skisserte modelleringsrammeverket brukes videre til å studere jacket konstruksjonen som en løsning på de oppstående utfordringene med å etablere havvindmølleparker i seismisk aktive områder.

Table of Contents

List of Figures

List of Tables

1 Introduction

As Denmark marked the start of offshore wind technologies when they built the first offshore wind turbine in 1991, Europe has taken the lead when it comes to offshore wind turbine innovation. The research and development for the last three decades in Europe has established offshore wind as a cost effective choice for governments, while the European offshore wind market has grown with an annual growth of 11% for the last decade [2]. All over the world renewable energy sources are wanted to reduce the CO_2 emissions. With EU enshrining in legislation the ambition of becoming climate neutral by 2050, the long-term and climate policies in Europe are exceptionally favorable to offshore wind.

The Asian offshore wind market was at a stand-still until the Chinese central government released the National Offshore Wind Development Plan. China passed UK as the world's top market in new installations in 2018 and is at the end of 2019 the world's third largest in total offshore wind turbine installations, behind UK and Germany [2].

With new wind farms being planned in oceans prone to earthquake in North America, Japan and China the monopile substructure which dominates the industry is questioned when it comes to large turbines excited by earthquake in deeper oceans. One alternative could be the jacket foundation which can sustain large lateral loads due to axial stresses. Georgiou et al. have shown that a jacket foundation can outperform a regular monopile foundation when it comes to developing rotations at the mud line [4].

1.1 State of the art

With the trend in offshore wind turbine size being driven by the goal of reducing the levelized cost of energy, the turbines have grown bigger and bigger to extract more energy per wind turbine. The largest wind turbine in prototype operation today is the GE Haliade-X 14MW offshore wind turbine [5]. This turbine has a 220m rotor diameter and is 248m high. The reason for building bigger is due to that the generated power of a wind turbine is proportional to the swept area and the relative wind speed cubed, as presented in equation (1.1.1).

$$P = \frac{1}{2}\rho_{air} C_P A_S V_{rel}^3 \tag{1.1.1}$$

where ρ_{air} is the air density, C_P is the power coefficient, A_S is the swept area and V_{rel} is the wind speed relative to the wind turbine. This means that the only way to increase the generated power of a wind turbine is to increase it's swept area and hub height. As the air density is largest at sea, the wind less turbulent and with a higher wind speed, it favors the offshore wind turbines. The higher wind turbines also utilizes that the wind speed increases with height.

The more stable wind conditions, reduced impact on other economic activities and less visual impact on the coastline are arguments for wanting to build further out in the ocean. The large new turbines as well as the wish to build further into the ocean requires the use of other substructures than the widely used monopile. The lattice design of a steel jacket provides a lightweight and stiff structure [1].

The average installed rated capacity for Europe in 2020 was 8.2MW which is an increase of 5% from 2019 compared to the constant annual growth of 16% since 2015. The growth in the average rated turbine capacity is shown in figure 1.1. New orders in Europe for 2020 show a trend towards the next generation of turbines with a rated power of 10 to 13 MW for projects after 2022 [1].

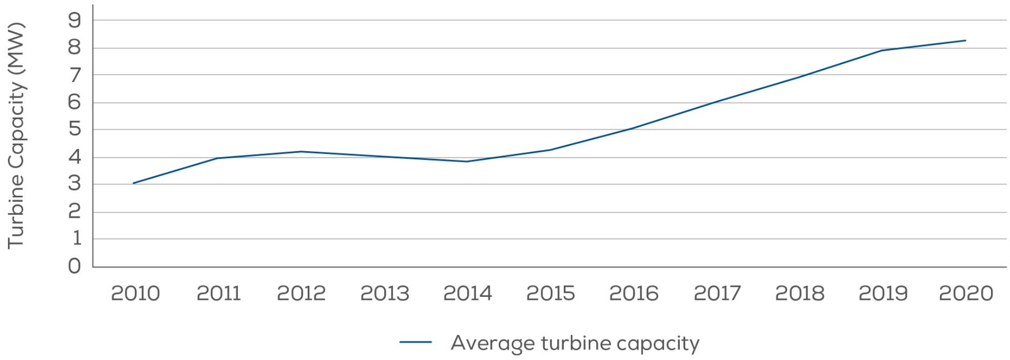


Figure 1.1: Yearly average of total installed offshore wind turbine rated capacity in Europe [1]

1.2 Offshore wind turbines

The offshore wind turbines (OWTs) are generally the same as the onshore wind turbines when it comes to the materials and properties of the tower and rotor-nacelle assembly (RNA). Onshore turbines can be placed everywhere on land given it is a place with strong and constant wind. With land being used for agriculture as well as for housing, the land available to wind turbines are limited. Onshore wind turbines also have to take into account the noise and visual pollution. OWTs, on the other hand, has the luxury of not having to take into account visual or noise pollution in the same way, such that the offshore wind turbines can be bigger in size. Offshore winds are also stronger and more constant compared to the wind onshore, such that the efficiency of offshore wind turbines are higher than for their onshore siblings. The development in the offshore wind turbines has reduced the levelised cost of energy (LCOE) by 67% since 2012 and the cost is estimated to reduce further as shown in figure 1.2.

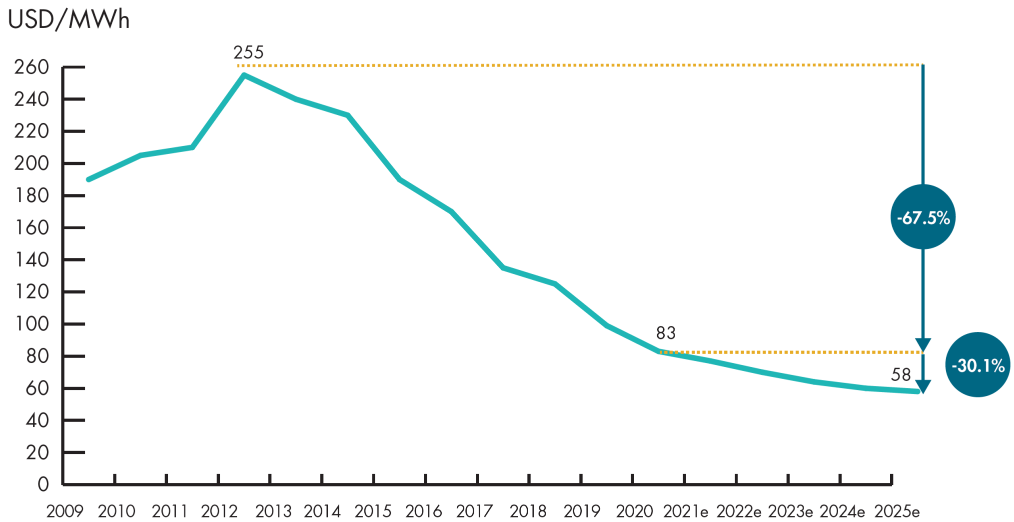


Figure 1.2: LCOE from offshore wind turbines [2]

The OWTs need substructures to hold them in place and there are several types of common offshore wind turbine substructures. The alternatives are monopile, mono-pod, jacket, tripod and several types of floating wind turbines. These substructures are used at different locations depending on water depth and other requirements. The monopile foundation is widely used for the majority of offshore wind turbines as shown in figure 1.3. This is due to the easy installations in shallow water where the turbines has been built.

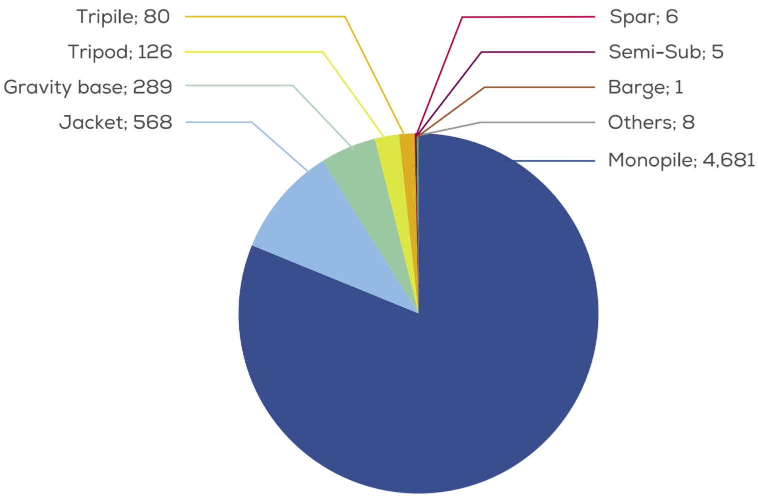


Figure 1.3: Cumulative number of foundations installed by end of 2020 in Europe [1]

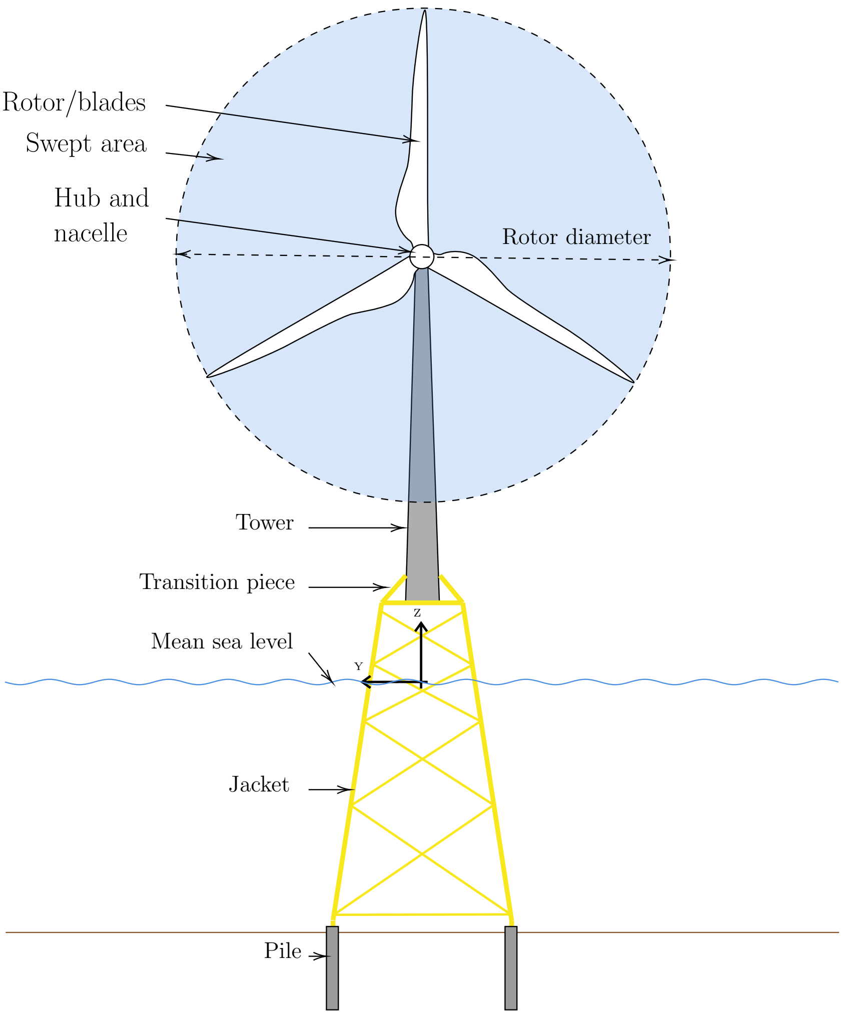


Figure 1.4: Bottom fixed offshore wind turbine nomenclature

The nomenclature for bottom fixed offshore wind turbines on jacket structures is presented in figure 1.4. The motion of a OWT is referred to as side-side and fore-aft motion. The fore-aft motion refers to motion normal to the plane of the blades, while side-side motion refers to motion in or parallel to the plane of the blades.

1.3 The reference offshore wind turbine

The structure analysed in this project is an OWT on a four legged steel jacket support structure. The OWT design used is based on the International Energy Agency's (IEA) 10-MW OWT [6], which is a further development of the 10-MW reference wind turbine (RWT) [7], referred to as the DTU 10-MW RWT, developed by the Technical University of Denmark (DTU). The jacket design is based on Rambøll's Reference Jacket design [8] from the INNWIND project. The jacket is mounted to the seabed by friction piles, but there is not presented any reference pile design accompanying the reference jacket. However, in a preliminary design report in the INNWIND project, Rambøll has presented a pile design for an earlier draft of the jacket [9]. This particular pile design is therefore used along with the reference jacket in this project. The connection between jacket and tower is performed with a so-called transition piece. Different types of transition pieces could be used for such constructions, but Rambøll presents a generic strutted steel beam transition piece along with the reference jacket, which will suit its purpose for this project. The Reference Jacket design report also presents a soil profile for the seabed, but the profile is to soft for this project. Therefore, the profile used is the presented profile with adjusted elasticity moduli, see appendix section A.1 for the profile used. The chosen design is more closely described in section 3.

An acknowledgement to the chosen design for the different parts is that the reference jacket is made for the DTU 10-MW RWT and not the further developed IEA 10-MW OWT. The latter OWT design is based on a monopile foundation with a different foundation/tower intersection level than for the jacket, and with a larger RNA, but with the same hub height. When the chosen tower then is placed on the chosen jacket, the tower hub height becomes higher and the jacket gets a larger structure upon it. The IEA tower and RNA structure actually has double the mass compared to the structure used when Rambøll developed the reference jacket. As the scope of this project is not optimization of structural design, the chosen design is assumed adequate for further analyses. Figure 1.5 shows an illustration of the IEA OWT placed on the reference jacket and table 1.1 summarizes the key dimensions of the structure.

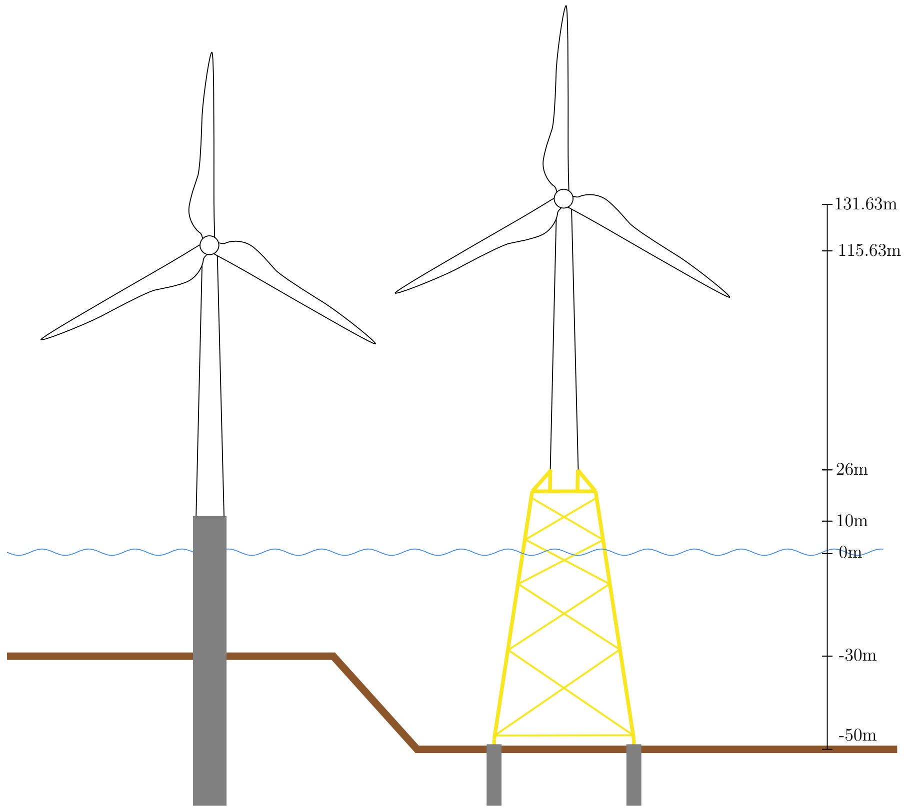


Figure 1.5: Illustration of the IEA OWT placed on the reference jacket.

Table 1.1: Key dimensions of the modelled structure

Measure	Value [m]
Tower length (not including transition piece)	105.63
Jacket length (not including transition piece)	66.5
Jacket top width	14
Jacket base width	34
Transition piece length	8
Pile length	43.5
Pile soil penetration	42

1.4 Earthquake consideration

Several studies have been conducted on OWT situated on monopile substructure, but there are only a few studies that has looked at the earthquake effect on OWT situated on jacket substructures. Georgiou et al. [4] has studied the non-linear soil effect of a 10MW OWT situated on both a monopile and a jacket substructure. Both models were excited with several different earthquake acceleration time series. The results show that the accumulated foundation rotation are much bigger for the monopile than for the jacket. This is good results when it comes to the performance of the jacket compared to the monopile, but further investigations are necessary to fully understand the seismic effects on OWT on jacket substructures.

An article written by Kaynia [10] reviews some of the key issues when it comes to earthquake analysis and design of OWT. He points out that in many cases OWTs are analyzed with the traditional p-y spring approach. Many studies has pointed out the inaccuracies of this approach, especially for large piles. Kaynia demonstrated in his article that in the case of soil-structure interaction (SSI) and OWT structures, settlement and permanent tilting could arise due to soil non-linearity and pore-pressure generation. He further highlighted the importance of performance based analysis in seismic design.

The main goal of this project is to establish a numerical model of the reference OWT able to include the SSI effects in an adequate manner, and include both environmental and earthquake loading. The purpose is then to present the chosen method and examined qualities for further research.

1.5 Modelling approach

Approaching the complex geometry and dynamics of an OWT makes the aero-hydro-servo-elastic computational software OpenFAST [11] highly relevant for this project. OpenFAST is custom made for simulating the environmental loads and dynamics of wind turbines, also including waves, current and submerged effects for an offshore structure. However, OpenFAST lack the opportunity of attaching a soil domain to the OWT structure. This leads to the choice of two complementary models; (1) an OpenFAST model attached to springs representing the pile and soil foundation, and (2) a fully integrated finite element model including both structure and soil. The latter is made in the finite element analysis tool Abaqus [12], and the geometry of the RNA is included only as added mass and mass moment of inertias.

The Abaqus model is first of all used to verify the establishing of the OpenFAST model, as OpenFast has no graphical user interface, and Abaqus has a wider documented and confirmed use. The Abaqus model is also used to get the stiffness and earthquake load applied to the OpenFAST model. For the analysis of non-linear soil dynamics, the Abaqus model, obviously, has to be used, but environmental loads on the OWT structure is neglected.

An introduction to the OpenFAST software is given in section 4.

2 Theory

2.1 Structural dynamics

This section presents the relevant theory of structural dynamics and the applied finite element approach. The theory of structural dynamics is based on Chopra's *Dynamics of structures* [13] and the finite element theory is based on Cook's *Concepts and applications of finite element analysis* [14]. Matrices and vectors are identified with boldface type, specified with brackets ("[ ]") for matrices and braces ("{ }") for vectors.

2.1.1 Equation of motion

Figure 2.1 shows a single degree of freedom (SDOF) system including a mass, m, able to move frictionless in the horizontal direction. The mass is attached to a linear spring with stiffness k and a dashpot working as a viscous damper with damping coefficient c. The system is subjected to an externally applied dynamic force, $P(t)$, working in the direction of the degree of freedom (DOF) u. The dynamic force varies with time, t, and thus the resulting mass displacement, $u(t)$.

The forces acting on the mass at a point in time are shown at the free body diagram (FBD) in figure 2.1. The acting forces are shown as continuous lined arrows, and include the external force, $P(t)$, the elastic force, f_S, and the damping resisting force, f_D. The horizontal resultant force and Newton's second law of motion gives

$$P(t) - f_S - f_D = m\ddot{u} \quad \text{or} \quad m\ddot{u} + f_D + f_S = P(t) \tag{2.1.1}$$

For a linear spring, the relationship between the elastic force, f_S, and displacement, u, is

$$f_S = ku \tag{2.1.2}$$

And for a viscous damper, the damping resisting force is related to the velocity, $\dot{u}$, by

$$f_D = c\dot{u} \tag{2.1.3}$$

Substituting equation (2.1.2) and (2.1.3) into equation (2.1.1) the equation of motion (EOM) for the SDOF system yields;

$$m\ddot{u} + c\dot{u} + ku = P(t) \tag{2.1.4}$$

This equation governs the displacement, $u(t)$, of a linearly elastic system subjected to an external dynamic force, $P(t)$. It is a second order differential equation, and the initial displacement $u(0)$ and velocity $\dot{u}(0)$ must be specified to define the problem completely.

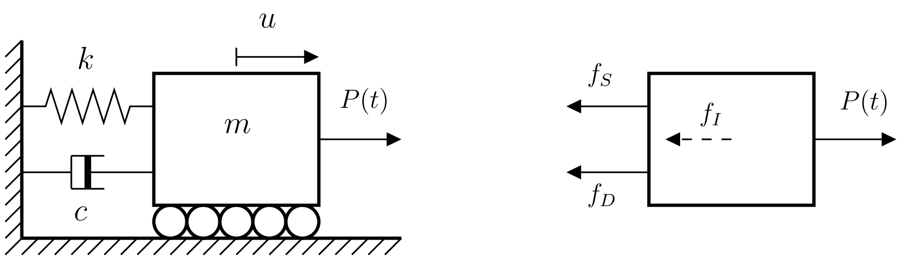


Figure 2.1: Left: SDOF system. Right: FBD of the system. Dashed lined arrow shows fictitious inertia force.

Structural engineers are trained to think in terms of equilibrium of forces, and the D'Alembert's principle of *dynamic equilibrium* is therefore a more common way to interpret the setup of the EOM. The principle is based on the concept of fictitious *inertia forces*, a force equal to the product of mass times its acceleration and acting in the opposite direction of the acceleration. It states that with inertia forces included, a system is in equilibrium at each time instant. By considering the system in figure 2.1 and its FBD (f_I representing the inertia force), the equation of motion can be developed by the principles of statics.

The D'Alembert's principle especially come in handy when formulating the equation of motion for a system based on assemblage of rigid bodies. A rigid body with distributed mass can be included in the equilibrium by considering the distributed inertia resultant as a force acting at the centre of mass (CM), and the rigid body mass moment of inertia as a moment acting around the CM. An example of this, and a justification of the approach used in the Abaqus model, treating the RNA as a rigid body and including it as a point mass and its mass moment of inertia at the tower top, is shown in Appendix B.1.

The complete solution of the SDOF EOM stated in equation (2.1.4) consists of the sum of a homogeneous and a particular solution;

$$u(t) = u_h(t) + u_p(t) \tag{2.1.5}$$

where the homogeneous solution, $u_h(t)$, often is referred to as the transient solution, and the particular solution, $u_p(t)$, often is referred to as the steady-state solution. Both the transient and the steady-state solution could be interesting individually. For convenience, the EOM in equation (2.1.4) is modified by dividing of the mass, m, and introducing some new variables;

$$\ddot{u} + 2\zeta\omega_n\dot{u} + \omega_n^2 u = \frac{P(t)}{m} \tag{2.1.6}$$

$\omega_n = \sqrt{\frac{k}{m}}$ denotes the natural frequency, $\zeta = \frac{c}{2m\omega_n}$ denotes the damping ratio and $2m\omega_n$ is referred to as the critical damping coefficient, c_{cr}.

The transient solution of a damped system with so-called under-critical damping , i.e., $\zeta < 1 \Rightarrow c < c_{cr}$, is

$$u_h(t) = \left[u(0)\cos(\omega_D t) + \frac{\dot{u}(0) + \zeta\omega_n u(0)}{\omega_D} \cdot \sin(\omega_D t)\right] \cdot e^{-\zeta\omega_n t} \tag{2.1.7}$$

where $\omega_D = \omega_n\sqrt{1-\zeta^2}$ and are called the damped natural frequency. A more convenient way of writing the transient response equation is

$$u_h(t) = \rho \cdot \cos(\omega_D t - \phi) \cdot e^{-\zeta\omega_n t} \tag{2.1.8}$$

where

$$\begin{aligned} \rho &= \sqrt{u(0)^2 + \left(\frac{\dot{u}(0) + \zeta\omega_n u(0)}{\omega_D}\right)^2} \\ \phi &= \tan^{-1}\left[\left(\frac{\dot{u}(0) + \zeta\omega_n u(0)}{\omega_D}\right)/u(0)\right] \end{aligned} \tag{2.1.9}$$

Equation (2.1.7) and (2.1.8) indicate that oscillation of a damped system has a modified angular frequency compared to an undamped system. This change in angular frequency is, however, very small for the most practical situations. E.g., for a system with 5% damping ratio ($\zeta = 0.05$) the relation is $\omega_D = 0.9987\omega_n$. The damped oscillation versus the undamped oscillation is visualized in figure 2.2, and the role of the damping term, $\rho e^{-\zeta\omega_n t}$, is also highlighted.

The steady-state solution is in general a product of a static response, $P(t)/k$, and a transfer function, $H(\omega)$;

$$u_p(t) = H(\omega) \cdot \frac{P(t)}{k} \tag{2.1.10}$$

The transfer function is a frequency dependent function, often referred to as a frequency response function. The function will achieve its maximum value when the loading frequency, ω, equals the natural frequency of the system. This phenomenon is known as resonance. However, the steady-state solution is only available for loading that can be described analytically, such as harmonic, step and pulse forces.

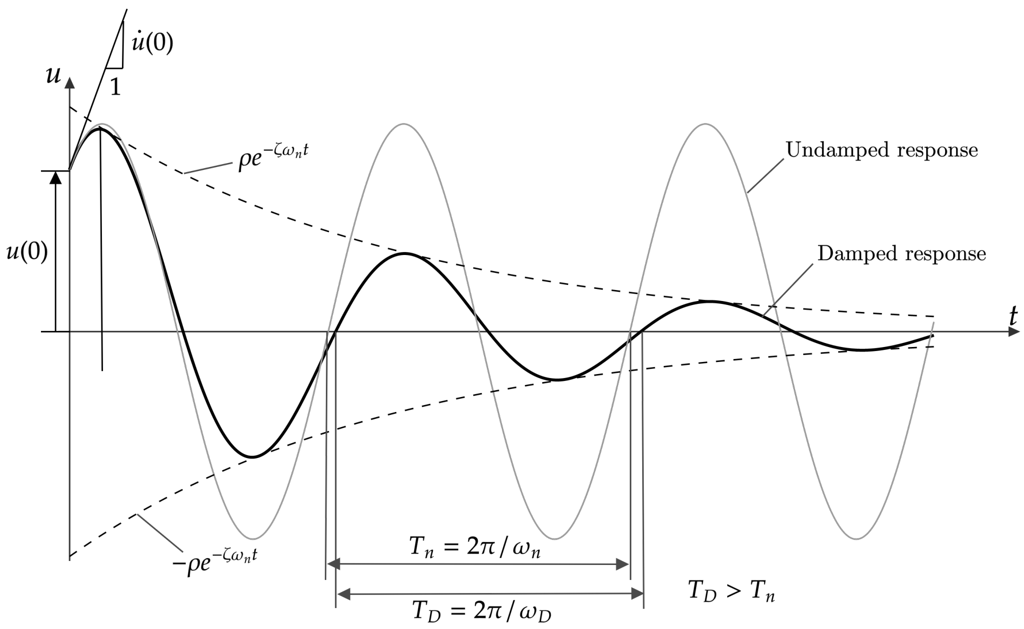


Figure 2.2: Effect of damping on the transient response.

Until now, the SDOF system has been considered; however, real structures are rarely represented by only one DOF. Idealization of a structure may need several DOFs to describe the system, and if a finite element (FE) approach is used, thousands of DOFs may be present. Systems described by more than one DOF are referred to as a multiple degree of freedom (MDOF) system.

The equation of motion for a MDOF system follows the same principles as for a SDOF system. Each DOF has an associated EOM, and the total response of the system is then described by solving each EOM in relation to the others. More precisely; a MDOF system represented by N DOFs is described by N coupled equations. The equations can be written on a compact matrix form as

$$[\mathbf{M}]\{\ddot{\mathbf{u}}\} + [\mathbf{C}]\{\dot{\mathbf{u}}\} + [\mathbf{K}]\{\mathbf{u}\} = \{\mathbf{P}(t)\} \tag{2.1.11}$$

where $[\mathbf{M}]$, $[\mathbf{C}]$ and $[\mathbf{K}]$ are the mass, damping and stiffness matrices and $\{\mathbf{u}\}$, $\{\dot{\mathbf{u}}\}$, and $\{\ddot{\mathbf{u}}\}$ are column vectors holding the DOF displacement and its time derivatives.

The system's natural frequencies and the corresponding shape of vibration, also known as mode shapes, are found by solving the eigenvalue problem

$$([\mathbf{K}] - \omega_n^2[\mathbf{M}])\{\boldsymbol{\phi}\}_n = \{\mathbf{0}\} \Rightarrow \det([\mathbf{K}] - \omega_n^2[\mathbf{M}]) = 0 \tag{2.1.12}$$

where $\{\boldsymbol{\phi}_n\}$ is the eigenvector and mode shape corresponding to the n-th eigenvalue, or natural frequency, ω_n. The mode shape vector represents the relative displacement between each DOF and not the actual physical values for the displacements.

To get the total system response, equation (2.1.11) need to be solved. It represents a coupled system, i.e., the response of one DOF is dependent on the response of the other DOFs. It is several ways of solving this system of equations, and one way, referred to as the *modal method* or *modal superpositioning*, is by utilizing the *orthogonality properties* of the mode shape vectors to make an uncoupled system of equations. The orthogonality property gives the following relation:

$$\begin{aligned} \{\boldsymbol{\phi}\}_n^T[\mathbf{M}]\{\boldsymbol{\phi}\}_n &= M_n^m \\ \{\boldsymbol{\phi}\}_n^T[\mathbf{K}]\{\boldsymbol{\phi}\}_n &= K_n^m \end{aligned} \tag{2.1.13}$$

where the m superscript denotes the *modal property.* As the mode shape vector only describes the relation between the DOFs, it can by scaled arbitrary, $\{\boldsymbol{\phi}\}_n' = \alpha \cdot \{\boldsymbol{\phi}\}_n$. A common way of scaling it, is a so-called *mass normalization*, which makes

$$\begin{aligned}\{\hat{\boldsymbol{\phi}}\}_n^T[\mathbf{M}]\{\hat{\boldsymbol{\phi}}\}_n &= 1\\ \{\hat{\boldsymbol{\phi}}\}_n^T[\mathbf{K}]\{\hat{\boldsymbol{\phi}}\}_n &= \omega_n^2\end{aligned} \tag{2.1.14}$$

where the hat superscript indicates mass normalization. By assuming classical damping, the same property yields for the damping matrix;

$$\begin{aligned}\{\boldsymbol{\phi}\}_n^T[\mathbf{C}]\{\boldsymbol{\phi}\}_n &= C_n^m = 2M_n^m\zeta_n\omega_n\\ \{\hat{\boldsymbol{\phi}}\}_n^T[\mathbf{C}]\{\hat{\boldsymbol{\phi}}\}_n &= \hat{C}_n^m = 2\zeta_n\omega_n\end{aligned} \tag{2.1.15}$$

where ζ_n then is the *modal damping ratio*, i.e., the damping ratio of mode n.

Now gathering all the mass normalized mode shapes and the squared eigenvalues in matrices:

$$\begin{aligned}[\hat{\boldsymbol{\Phi}}] = \begin{bmatrix}\{\hat{\boldsymbol{\phi}}\}_1 & \{\hat{\boldsymbol{\phi}}\}_2 & ... & \{\hat{\boldsymbol{\phi}}\}_n & ... & \{\hat{\boldsymbol{\phi}}\}_N\end{bmatrix} &= \begin{bmatrix}\hat{\phi}_{1,1} & \hat{\phi}_{1,2} & \cdots & \hat{\phi}_{1,n} & \cdots & \hat{\phi}_{1,N}\\ \hat{\phi}_{2,1} & \hat{\phi}_{2,2} & \cdots & \hat{\phi}_{2,n} & \cdots & \hat{\phi}_{2,N}\\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots\\ \hat{\phi}_{n,1} & \hat{\phi}_{n,2} & \cdots & \hat{\phi}_{n,n} & \cdots & \hat{\phi}_{n,N}\\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots\\ \hat{\phi}_{N,1} & \hat{\phi}_{N,2} & \cdots & \hat{\phi}_{N,n} & \cdots & \hat{\phi}_{N,N}\end{bmatrix}\\ [\boldsymbol{\Omega}] &= \begin{bmatrix}\omega_1^2 & & & & 0 & \\ & \omega_2^2 & & & & \\ & & \ddots & & & \\ & & & \omega_n^2 & & \\ & 0 & & & \ddots & \\ & & & & & \omega_N^2\end{bmatrix}\end{aligned} \tag{2.1.16}$$

Let $\{\mathbf{u}\} = [\hat{\boldsymbol{\Phi}}]\{\mathbf{y}\}$, where $\{\mathbf{y}\}$ is the generalized DOFs, often called *modal coordinates*, and substitute into equation (2.1.11);

$$[\mathbf{M}][\hat{\boldsymbol{\Phi}}]\{\ddot{\mathbf{y}}\} + [\mathbf{C}][\hat{\boldsymbol{\Phi}}]\{\dot{\mathbf{y}}\} + [\mathbf{K}][\hat{\boldsymbol{\Phi}}]\{\mathbf{y}\} = \{\mathbf{P}(t)\} \tag{2.1.17}$$

Pre-multiply with $[\hat{\boldsymbol{\Phi}}]^T$:

$$[\hat{\boldsymbol{\Phi}}]^T[\mathbf{M}][\hat{\boldsymbol{\Phi}}]\{\ddot{\mathbf{y}}\} + [\hat{\boldsymbol{\Phi}}]^T[\mathbf{C}][\hat{\boldsymbol{\Phi}}]\{\dot{\mathbf{y}}\} + [\hat{\boldsymbol{\Phi}}]^T[\mathbf{K}][\hat{\boldsymbol{\Phi}}]\{\mathbf{y}\} = [\hat{\boldsymbol{\Phi}}]^T\{\mathbf{P}(t)\} \tag{2.1.18}$$

$$\Rightarrow [\mathbf{I}]\{\ddot{\mathbf{y}}\} + [\mathbf{C}]^m\{\dot{\mathbf{y}}\} + [\boldsymbol{\Omega}]\{\mathbf{y}\} = \{\mathbf{P}(t)\}^m \tag{2.1.19}$$

$$\Rightarrow \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & \ddots & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{Bmatrix}\ddot{y}_1(t)\\ \ddot{y}_2(t)\\ \vdots\\ \ddot{y}_N(t)\end{Bmatrix} + \begin{bmatrix}2\zeta_1\omega_1 & 0 & 0 & 0\\ 0 & 2\zeta_2\omega_2 & 0 & 0\\ 0 & 0 & \ddots & 0\\ 0 & 0 & 0 & 2\zeta_N\omega_N\end{bmatrix}\begin{Bmatrix}\dot{y}_1(t)\\ \dot{y}_2(t)\\ \vdots\\ \dot{y}_N(t)\end{Bmatrix} + \begin{bmatrix}\omega_1^2 & 0 & 0 & 0\\ 0 & \omega_2^2 & 0 & 0\\ 0 & 0 & \ddots & 0\\ 0 & 0 & 0 & \omega_N^2\end{bmatrix}\begin{Bmatrix}y_1(t)\\ y_2(t)\\ \vdots\\ y_N(t)\end{Bmatrix} = \begin{Bmatrix}P_1^m(t)\\ P_2^m(t)\\ \vdots\\ P_N^m(t)\end{Bmatrix}$$

The above system of equations is now the uncoupled MDOF system and is a mathematically exact representation of equation (2.1.11). Each equation can now be solved, analytically or numerically, independently for $y_n(t)$. After solving all equations, the total response of the system is retrieved by the $\{\mathbf{u}\} = [\hat{\boldsymbol{\Phi}}]\{\mathbf{y}\}$ relation. The uncoupled matrices is referred to as the modal matrices, and in this case the mass normalized modal matrices.

Worth noticing is that in many practical problems only the first modes contribute significantly to the response solution. Thus, giving the reduced order approximation [13]:

$$\{\mathbf{u}\} \approx \sum_{n=1}^{k}\{\boldsymbol{\phi}_n\}y_n(t) \Rightarrow \{\mathbf{u}_{\text{red}}\} = [\{\boldsymbol{\phi}\}_1\ \{\boldsymbol{\phi}\}_2\ ...\ \{\boldsymbol{\phi}\}_k] = [\boldsymbol{\Phi}_{\text{red}}] \tag{2.1.20}$$

The modal matrices then become size $k \times k$ and typically $k << N$ provides satisfactory accuracy. E.g., for a structure excited by an earthquake, k may be less than 20 while N exceeds 1000. This approximation becomes useful for systems with many degrees of freedoms and only k of N modes extracted, such as the case in finite element analysis, discussed further in section 2.1.4. Note that the reduced mode set must include all lower modes, without omission, up to a mode with a chosen frequency. Choosing the number of modes to include is an engineering judgement and needs to be done with caution. The main rule is to ensure that the k-th mode's frequency surpasses the highest important frequency of the loading. But the choice of included modes should also concern the spatial complexity of the loading, whether results in addition to displacements are required, and the wanted accuracy of the results.

2.1.2 Damping

Damping is the process where the free vibration of the system response steadily diminishes in amplitude. This happens due to dissipation of energy, which happens due to several mechanisms. E.g., friction in steel connections, opening and closing of microcracks in concrete and the influence from the surroundings such as water. Describing all these mechanisms mathematically is cumbersome and not practical, and damping are therefore often highly idealized. A common representation in structural engineering is linear viscous dampers or dashpots. The damping is then represented by a damping coefficient giving the same energy dissipation as all the damping mechanisms combined. This idealization is therefore called *equivalent viscous damping* and is often referred to as *classical damping*. As shown in equation (2.1.3), the damping force from a linear viscous damper is directly related to the velocity.

One form of classical damping is the *Rayleigh damping*, which is based on linearly combining the mass and stiffness matrices, i.e., a combination of mass and stiffness proportional damping. The damping matrix then becomes

$$[\mathbf{C}] = \alpha[\mathbf{M}] + \beta[\mathbf{K}] \tag{2.1.21}$$

With symmetric mass and stiffness matrices, the damping matrix also becomes symmetric, and the orthogonality properties will apply. The damping ratio for mode n, ζ_n, is then given by:

$$\zeta_n = \alpha \cdot \frac{1}{2\omega_n} + \beta \cdot \frac{\omega_n}{2} \tag{2.1.22}$$

where ω_n is the natural frequency of mode n. This damping ratio will relate, according to equation (2.1.15), to the modal damping coefficient as

$$C_n^m = 2M_n^m \zeta_n \omega_n \tag{2.1.23}$$

and figure 2.3 shows the varying damping ratio as a function of frequency.

The Rayleigh coefficients, α and β, can be determined from specified damping ratios for two modes, mode i and j. Expressing equation (2.1.22) for these two modes on matrix form leads to

$$\frac{1}{2}\begin{bmatrix} 1/\omega_i & \omega_i \\ 1/\omega_j & \omega_j \end{bmatrix} \begin{Bmatrix} \alpha \\ \beta \end{Bmatrix} = \begin{Bmatrix} \zeta_i \\ \zeta_j \end{Bmatrix} \tag{2.1.24}$$

Experimental data shows that different modes may have approximately the same damping ratio, and if $\zeta_i = \zeta_j$, the Rayleigh coefficients are:

$$\alpha = \zeta \frac{2\omega_i \omega_j}{\omega_i + \omega_j} \qquad \beta = \zeta \frac{2}{\omega_i + \omega_j} \tag{2.1.25}$$

Damping coefficients for any other mode is then decided by equation (2.1.22).

Applying Rayleigh damping to a practical problem usually involve *tuning* the coefficients from the already obtained natural frequencies of two modes. The modes chosen should ensure reasonably damping for the modes contributing significantly to the response. As most practical systems involves a response dominated by the first modes, the first tuning mode usually is the first mode. The second tuning mode should then be chosen in conjunction with the largest expected load

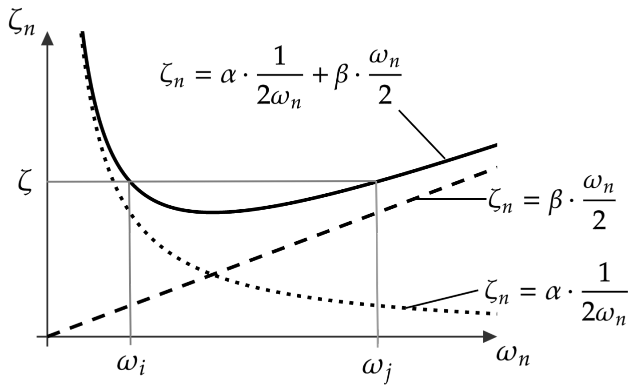


Figure 2.3: Variation of modal damping with natural frequency. Continuous line: Rayleigh damping. Dotted line: Mass proportional damping. Dashed line: Stiffness proportional damping.

frequency. As seen in figure 2.3 the modes with frequencies in between the two chosen tuning frequencies will get a lower damping ratio than the tuning ratio. The damping ratio of modes higher than the second tuning mode will increase monotonically with frequency, and the corresponding modal response will essentially be eliminated due to their high damping.

Another way of defining the modal damping, is of course to use an appropriate or experimentally determined ζ_n-value for each mode directly.

2.1.3 Damping estimation

The damping ratio, ζ, is impossible to determine analytically for real structures, and damping must be estimated in a different way. One approach is to perform a free decay test. I.e., Apply initial conditions and allow the damped system to do a free vibration, offers the opportunity to measure the damping of the response, assuming it behaves like equation (2.1.8). One common method of measuring the damping from such a free decay test is by the *logarithmic decrement method* as derived in Chopra's Book [13]. The prerequisite is that the peak values, u_i, and the corresponding time instances, t_i, of the decaying time series are known. The peak values then represents the decaying amplitude and the corresponding time instances indicate the damped period of the oscillation. The damping ratio is then estimated as:

$$\zeta = \frac{1}{2\pi} \ln \left(\frac{u_i}{u_i + u_j} \right) \tag{2.1.26}$$

where i represents the number of the first peak (largest) in the estimation, and j represents the number of the last (smallest) peak in the estimation. This estimation is, however, based on the assumption that ζ is small and that, $\sqrt{1-\zeta^2} \simeq 1$.

Another method, based on the concepts of *least square fitting* (LSF) [15], allows several peaks to be included in the estimation. The peaks are defined as u_i for $i = 1,\ 2,\ ...,\ n$, ordered by increasing time. The method is referred to as the *LSF damping estimation method* and is derived in Appendix B.2. The estimated damping ratio for this method is given as:

$$\delta = \ln \left(\frac{u_1^{n-1}}{\prod_{i=2}^{n} u_i} \right) \cdot \frac{1}{\pi n(n-1)} \tag{2.1.27}$$

$$\zeta = \frac{\delta}{\sqrt{1+\delta^2}} \tag{2.1.28}$$

2.1.4 FE formulation of the equation of motion

In finite element (FE) analysis the equation of motion is derived in terms of the *principle of virtual work*; virtual work over the imagined displacement $\{\delta \mathbf{u}\}$, and corresponding imagined strains $\{\delta \boldsymbol{\varepsilon}\}$, done by internal and dissipative (damping) forces equals the virtual work done by external forces over the same displacement:

$$\begin{aligned}\int \big(\overbrace{\{\delta\mathbf{u}\}^T\rho\{\ddot{\mathbf{u}}\}}^{\text{internal work}} + \overbrace{\{\delta\mathbf{u}\}^T c\{\dot{\mathbf{u}}\}}^{\text{dissipative work}} + \overbrace{\{\delta\boldsymbol{\varepsilon}\}^T\{\boldsymbol{\sigma}\}}^{\text{internal work}}\big)dV = \\ \underbrace{\int\{\delta\mathbf{u}\}^T\{\mathbf{F}\}dV + \int\{\delta\mathbf{u}\}^T\{\boldsymbol{\Phi}\}dS + \sum_{i=1}^{n}\{\delta\mathbf{u}\}_i^T\{\mathbf{p}\}_i}_{\text{external work}}\end{aligned} \tag{2.1.29}$$

where ρ represents mass density, c a damping parameter and $\{\mathbf{F}\}$, $\{\boldsymbol{\Phi}\}$ and $\{\mathbf{p}\}_i$ represent prescribed body forces, surface tractions and concentrated forces at node i, respectively.

The FE discretization is assumed as

$$\{\mathbf{u}\} = [\mathbf{N}]\{\mathbf{d}\} \tag{2.1.30}$$

$$\{\dot{\mathbf{u}}\} = [\mathbf{N}]\{\dot{\mathbf{d}}\} \tag{2.1.31}$$

$$\{\ddot{\mathbf{u}}\} = [\mathbf{N}]\{\ddot{\mathbf{d}}\} \tag{2.1.32}$$

$$\{\boldsymbol{\varepsilon}\} = [\mathbf{B}]\{\mathbf{d}\} \tag{2.1.33}$$

where $[\mathbf{N}] = [\mathbf{N}(x, y, z)]$ is the spatial shape functions of the elements, $[\mathbf{B}] = [\boldsymbol{\partial}][\mathbf{N}]$, where $[\boldsymbol{\partial}]$ is the strain-displacement operator, and $\{\mathbf{d}\} = \{\mathbf{d(t)}\}$ is the nodal displacements as a function of time. The EOM then becomes

$$[\mathbf{m}]\{\ddot{\mathbf{d}}\} + [\mathbf{c}]\{\dot{\mathbf{d}}\} + \{\mathbf{r}^{\text{int}}\} = \{\mathbf{r}^{\text{ext}}\} \tag{2.1.34}$$

for each element. $[\mathbf{m}]$ and $[\mathbf{c}]$ here denotes the consistent mass and damping matrices defined as:

$$[\mathbf{m}] = \int \rho[\mathbf{N}]^T[\mathbf{N}]dV \tag{2.1.35}$$

$$[\mathbf{c}] = \int c[\mathbf{N}]^T[\mathbf{N}]dV \tag{2.1.36}$$

and the internal, $\{\mathbf{r}^{\text{int}}\}$, and external, $\{\mathbf{r}^{\text{ext}}\}$, force vectors read:

$$\{\mathbf{r}^{\text{int}}\} = \int[\mathbf{B}]^T\{\boldsymbol{\sigma}\}dV \tag{2.1.37}$$

$$\{\mathbf{r}^{\text{ext}}\} = \int[\mathbf{N}]^T\{\mathbf{F}\}dV + \int[\mathbf{N}]^T\{\boldsymbol{\Phi}\}dS + \sum_{i=1}^{n}\{\mathbf{p}\}_i \tag{2.1.38}$$

For linear elastic material the internal force vector, $\{\mathbf{r}^{\text{int}}\}$, may be expressed in terms of the element stiffness matrix, $[\mathbf{k}]$, times the nodal displacements, $\{\mathbf{d}\}$;

$$\{\mathbf{r}^{\text{int}}\} = \int[\mathbf{B}]^T\{\boldsymbol{\sigma}\}dV = [\mathbf{k}]\{\mathbf{d}\} \tag{2.1.39}$$

Gathering all the element matrices by their connectivity to the global system; the governing global EOM yields:

$$[\mathbf{M}]\{\ddot{\mathbf{D}}\} + [\mathbf{C}]\{\dot{\mathbf{D}}\} + \{\mathbf{R}^{\text{int}}\} = \{\mathbf{R}^{\text{ext}}\} \tag{2.1.40}$$

$$[\mathbf{M}]\{\ddot{\mathbf{D}}\} + [\mathbf{C}]\{\dot{\mathbf{D}}\} + [\mathbf{K}]\{\mathbf{D}\} = \{\mathbf{R}^{\text{ext}}\} \tag{2.1.41}$$

where the capital letter notation symbolizes the global properties and $\{\mathbf{D}\}$ and $\{\mathbf{R}^{\text{ext}}\}$ are equivalents to $\{\mathbf{u}\}$ and $\{\mathbf{P}(t)\}$ in equation (2.1.11). The choice of different notation of the latter is to indicate that a FE approach is considered. These equations represent a semi-discretization, as the response is spatially discretized by a finite number of nodes, but the nodal motions are continuous functions of time.

Solving the FE EOM can be done with modal superpositioning, as discussed in section (2.1.1), if the system is linear, but the modal method has disadvantages as it incurs the computational expense of solving an eigenvalue problem. Also, the uncoupled equations need, in many cases, to be solved numerically. The coupled FE system is therefore often solved numerically by direct integration, discussed further in section 2.1.5. A numerical solution will make the FE equation fully discretized by obtaining solutions at a finite number of time instances.

The natural frequencies and mode shapes are, however, often of interest anyways in a structural analysis, especially the lower frequency modes. The remedy is therefore solving the eigenvalue problem with a solution algorithm, extracting modes only for a specified frequency range. Computational software, as Abaqus, utilizes several eigenvalue extraction algorithms such as the *Lanczos*-, *AMS*- and *Subspace iteration*-algorithm [16]. The EOM can then be solved by modal superpositioning of only the first relevant modes (reduced order).

A way of reducing the computational cost, is to utilize *substructuring*. The procedure is somewhat analogous to static condensation, as the given substructure matrices is reduced to include only certain nodes, preferably the nodes connecting the substructure to the rest of the model. The substructure is then included as all other elements and can be seen as an element with many internal DOFs. The name *superelement* is therefore often used to describe a substructure.

Dynamic substructuring, in contrast to static substructuring, do not preserve the full information of the complete system, but is highly effective to reduce computational time. Substructuring is also convenient when different design groups or firms need to work on different parts of a structure. Redesign of one substructure does not affect the internal modes of others. One of the most widely used dynamic substructuring techniques is the *Craig-Bampton reduction method* discussed in section 2.1.7.

2.1.5 Direct integration of equation of motion

Solving the EOM *directly* means solving it without first changing the form of the equation. Solving by *integration* then alludes on the fact that the wanted displacement is defined by an equation that relates it to its derivatives, and to get rid of derivatives, integration is needed. *Direct integration* methods solve the EOM for $\{\mathbf{D}\}$ at a given time step by representing the derivatives by their numerical integration (approximation) in time. The considered time interval is divided into N, usually equal, time increments, and the equation of motion at time step t_n is:

$$[\mathbf{M}]\{\ddot{\mathbf{D}}\}_n + [\mathbf{C}]\{\dot{\mathbf{D}}\}_n + \{\mathbf{R}^{\text{int}}\}_n = \{\mathbf{R}^{\text{ext}}\}_n \tag{2.1.42}$$

$$[\mathbf{M}]\{\ddot{\mathbf{D}}\}_n + [\mathbf{C}]\{\dot{\mathbf{D}}\}_n + [\mathbf{K}]\{\mathbf{D}\}_n = \{\mathbf{R}^{\text{ext}}\}_n \tag{2.1.43}$$

with $n = 1,\ 2,\ _{...},\ N$. The first form is better suited to a non-linear problem in which $[\mathbf{K}]$ change from one time step to the next.

Methods of direct integration calculate conditions at time step t_{n+1}, and classifies as either *explicit* or *implicit*. Explicit methods utilizes conditions only from previous time steps where the solution is already known, while implicit methods includes conditions at t_{n+1} as well. The implicit methods then need to solve additional equation for each time step to predict the t_{n+1}-values. The implicit methods are therefore more computational demanding for each step, but the overall computational expense may be lower as explicit integration requires sufficient small time increments to be numerically stable. Common implicit methods, on the other hand, are unconditionally stable, giving the opportunity for larger time steps. Explicit methods are ideal for high-speed dynamic simulations, where very small time increments are required, but for problems where the response period is long, such as earthquake response, implicit methods are preferred. Conceptually is the difference between the methods shown by their general forms:

$$\{\mathbf{D}\}_{n+1} = f(\{\mathbf{D}\}_n, \{\dot{\mathbf{D}}\}_n, \{\ddot{\mathbf{D}}\}_n, \{\mathbf{D}\}_{n-1}, ...) \quad \text{explicit} \tag{2.1.44}$$

$$\{\mathbf{D}\}_{n+1} = f(\{\dot{\mathbf{D}}\}_{n+1}, \{\ddot{\mathbf{D}}\}_{n+1}, \{\mathbf{D}\}_n, \{\dot{\mathbf{D}}\}_n, \{\ddot{\mathbf{D}}\}_n, \{\mathbf{D}\}_{n-1}, ...) \quad \text{implicit} \tag{2.1.45}$$

(2.1.46)

A more specific description of the procedure is that the next time step condition, $\{\mathbf{D}\}_{n+1}$, is calculated form a static equilibrium equivalent equation;

$$[\mathbf{K}^{\text{eff}}]\{\mathbf{D}\}_{n+1} = \{\mathbf{R}^{\text{eff}}\}_n \quad \text{explicit} \tag{2.1.47}$$

$$[\mathbf{K}^{\text{eff}}]\{\mathbf{D}\}_{n+1} = \{\mathbf{R}^{\text{eff}}\}_{n+1} \quad \text{implicit} \tag{2.1.48}$$

where $\{\mathbf{R}^{\text{eff}}\}_{n+1}$ for the implicit methods need to be calculated before solving the equilibrium, while the $\{\mathbf{R}^{\text{eff}}\}_n$ for the explicit methods are given by the previous step conditions.

Direct integration of the EOM applies to all situations, even non-linear systems and non-classical damped systems. It should, however, be noticed that unconditional stable implicit methods in linear problems does not guarantee unconditional stability in a non-linear problem.

The solutions will inhabit numerical errors, but converge towards the exact solution of the system discretization as the time steps become smaller. Implicit methods on non-linear cases also depends on the convergence of the non-linear equilibrium equation solving at each time step.

A common implicit integration method is the Newmark method, but an extended, and more sophisticated method, is the Hilber-Hughes-Taylor method [17] which is used by Abaqus when performing dynamic implicit analysis [18].

2.1.6 Solving Non-Linear FE problems

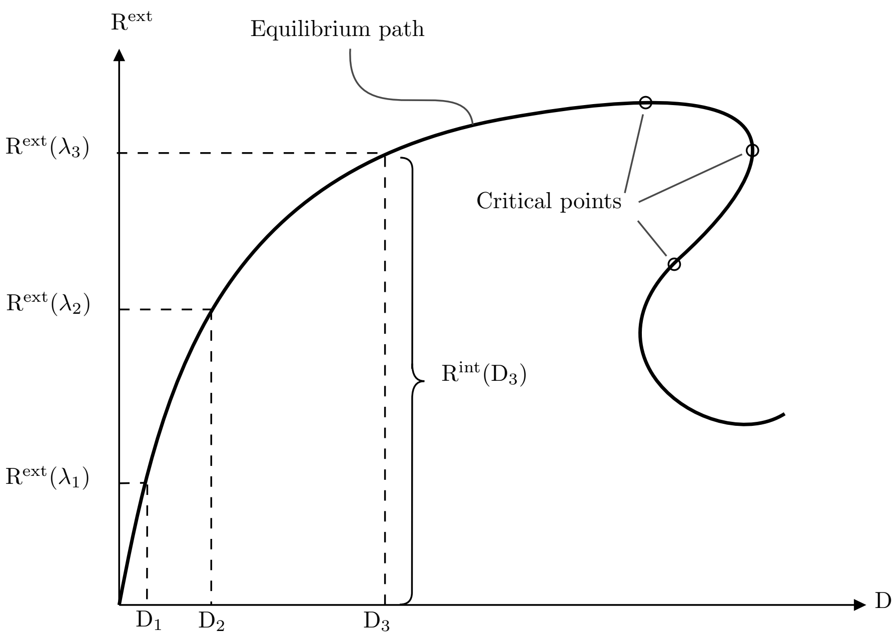


Figure 2.4: Non-linear equilibrium path. External force not dependent on displacements for simplicity.

A brief introduction to the concept of solving the non-linear FE problem will now be discussed. Starting off by the FE approximation of the global equilibrium for a non-dynamic case;

$$\overbrace{\{\mathbf{R}^{\text{ext}}\}}^{\text{externally applied load}} = \overbrace{\{\mathbf{R}^{\text{int}}\}}^{\text{nodal forces from internal element stresses}} \tag{2.1.49}$$

In order to satisfy this, the internal forces must be in balance with the external forces, hence, the residual force, $\{\mathbf{R}^{\text{res}}\}$, has to be zero.

$$\{\mathbf{R}^{\text{res}}\} = \{\mathbf{R}^{\text{ext}}\} - \{\mathbf{R}^{\text{int}}\} = \{\mathbf{0}\} \tag{2.1.50}$$

For non-linear problems, both the external forces and the stiffness may be dependent on the displacements;

$$\{\mathbf{R}^{\text{ext}}\} = \{\mathbf{R}^{\text{ext}}(\mathbf{D}, \lambda)\}, \quad \{\mathbf{R}^{\text{int}}\} = [\mathbf{K}(\mathbf{D})]\{\mathbf{D}\} \tag{2.1.51}$$

where λ denotes the *pseudo-time*, or *load step*. Thus, the problem consists of finding the displacement producing an internal force balancing out the externally applied force. The problem is visualized in figure 2.4. To achieve this numerically, an incremental form of the problem is needed. The incremental form of the equilibrium equation, expressed in terms of incremental nodal displacements $\{\Delta\mathbf{D}\}_n$, is obtained by linearizing the residual equation on basis of Taylor series expansion:

$$\{\mathbf{R}^{\text{res}}\}_{n+1} = \{\mathbf{R}^{\text{res}}\}_n + \left[\frac{\partial\mathbf{R}^{\text{res}}}{\partial\mathbf{D}}\right]_n \{\Delta\mathbf{D}\}_n = \mathbf{0} \tag{2.1.52}$$

Introducing the tangent stiffness $[\mathbf{K}_T]$ as

$$[\mathbf{K}_T] = -\left[\frac{\partial\mathbf{R}^{\text{res}}}{\partial\mathbf{D}}\right] = \left[\frac{\partial\mathbf{R}^{\text{int}}}{\partial\mathbf{D}}\right] - \left[\frac{\partial\mathbf{R}^{\text{ext}}}{\partial\mathbf{D}}\right] \Rightarrow [\mathbf{K}_T] = [\mathbf{K}_T(\mathbf{D},\lambda)] \tag{2.1.53}$$

gives the *global equilibrium on incremental form*:

$$[\mathbf{K}_T]_n\{\Delta\mathbf{D}\}_n = \{\mathbf{R}^{\text{res}}\}_n \tag{2.1.54}$$

The tangent stiffness matrix and the residual force vector are obtained by evaluating equation (2.1.53) and (2.1.50) at the next load step λ_{n+1}, and the previous known displacements $\{\mathbf{D}\}_n$. Solving the incremental equation for $\{\Delta\mathbf{D}\}_n$ then gives the next displacement conditions by:

$$\{\mathbf{D}\}_{n+1} = \{\mathbf{D}\}_n + \{\Delta\mathbf{D}\}_n \tag{2.1.55}$$

The most frequently used solution procedures consists of a predictor step involving forward Euler load incrementation, as in equation (2.1.55), and a corrector step in which some kind of Newton iterations are used to enforce equilibrium [19]. Such a procedure is therefore called an *incremental-iterative procedure*. A purely incremental (only applying the predictor step) procedure may lead to a progressive drift-off from the true equilibrium path, and the iterative equilibrium-enforcing (the corrector step) is therefore highly recommended. Figure 2.5 shows the incremental-iteration of the *Newton-Raphson* method. After solving the predictor step, this method iterates towards equilibrium by solving the incremental equation with updated tangent stiffness matrix and residual vector obtained from evaluating equation (2.1.53) and (2.1.50) at the previous iteration displacement approximation (starting with the solution from the predictor step, $\{\mathbf{D}\}^0_{n+1}$) and the next load step λ_{n+1}. The iteration continues until the convergence criteria is met, which for this method is a given tolerance value of the evaluated residual vector.

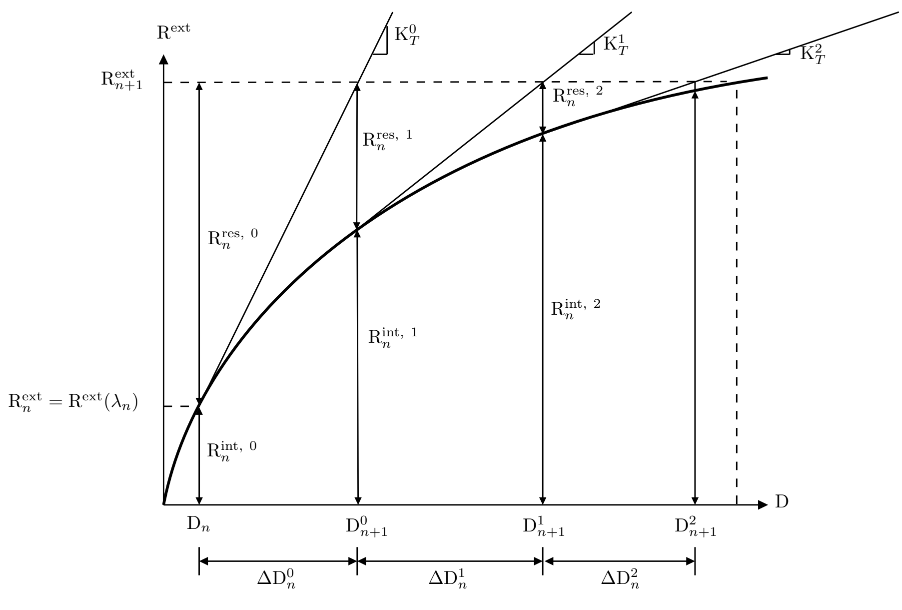


Figure 2.5: Visualization of the Newton Raphson method. External force not dependent on displacement for simplicity.

As the main purpose is to trace the fundamental equilibrium path, challenges occurs when the paths become more complex, inhabiting types of critical points, see figure 2.4. The main challenge to ensure the possibility for convergence along the path, lays in the choice of load incrementation size. Some paths even requires negative load incrementation to be able to traverse some of the critical points. The remedy is to use adaptive solution algorithms, that on basis of certain user prescribed inputs, and the degree of non-linearity of the path can adjust the size of the load incrementation. The use of *arc length methods* provide this kind of behaviour [20].

Another important aspect of the equation solving, is the choice of convergence criterion, i.e., the measure of how well the obtained solution satisfies equilibrium. The criterion is usually in some form based on displacements, residuals or energy (product of residual and displacements). Choosing the type of criteria and tolerance level should be done carefully to provide both accurate and computational economical solutions.

The approach on non-linear dynamic equations is somewhat similar as for the static case presented above. However, solving the equation of motion by explicit direct integration methods does not require iterations and convergence checks. That is because the next time step conditions purely are approximated by the previous conditions, demanding very small time steps for the solution to be numerically stable. For implicit methods, enforcing of equilibrium becomes necessary at each time step, and the incremental-iterative solution algorithms apply. The application is made possible as the approach for the static case includes the time equivalent load step (pseudo-time). The difference is that the right hand expression of equation (2.1.49) get the damping force and inertia terms in addition;

$$\{\mathbf{R}^{\text{ext}}\} = [\mathbf{M}]\{\ddot{\mathbf{D}}\} + \{\mathbf{R}^{\text{dmp}}\} + \{\mathbf{R}^{\text{int}}\} \tag{2.1.56}$$

The common way is then to linearize the internal force instead of the residual force. Giving the incremental equation

$$\{\mathbf{R}^{\text{int}}\}_{n+1} = \{\mathbf{R}^{\text{int}}\}_n + \left[\frac{\partial \mathbf{R}^{\text{int}}}{\partial \mathbf{D}}\right]\{\Delta\mathbf{D}\}_n = \{\mathbf{R}^{\text{int}}\}_n + [\mathbf{K}_T]_n\{\Delta\mathbf{D}\}_n \tag{2.1.57}$$

Substituting into the equation of motion gives

$$[\mathbf{M}]\{\ddot{\mathbf{D}}\}_{n+1} + [\mathbf{C}]\{\dot{\mathbf{D}}\}_{n+1} + [\mathbf{K}_T]_n\{\Delta\mathbf{D}\}_n = \{\mathbf{R}^{\text{ext}}\}_{n+1} - \{\mathbf{R}^{\text{int}}\}_n \tag{2.1.58}$$

where some approximation for the velocity and the acceleration are needed, e.g., by Newmark approximations. The equation of motion on incremental form then becomes

$$[\mathbf{K}^{\text{eff}}]_n\{\Delta\mathbf{D}\}_n = \{\Delta\mathbf{R}^{\text{eff}}\}_{n+1} \tag{2.1.59}$$

where $[\mathbf{K}^{\text{eff}}]$ gathers all the terms associated with $\{\Delta\mathbf{D}\}$ and $\{\Delta\mathbf{R}^{\text{eff}}\}_{n+1}$ gathers the approximated terms. These expressions depend on the choice of velocity and acceleration approximation. Equation (2.1.59) may then be solved by the same incremental-iterative methods as for the static case.

There are several important considerations when embarking on non-linear analysis. As the principle of superpositioning does not apply, results are not proportional to the load, and different load cases cannot be combined. The sequence of applying the loads may also become relevant, meaning that reversing the sequence may produce different results. Also, initial state of stress, like residual stresses from welding, temperature or prestressing of concrete, may be of great importance for the overall response.

2.1.7 Craig-Bampton reduction

As previously stated, the Craig-Bampton method is one of the most commonly used dynamic substructuring techniques in engineering practice [21]. The number of DOFs in a typical beam frame structure can easily grow to thousands. This combined with time-domain simulations for turbine dynamics can slow down the efficiency of aero-hydro-servo-elasto-dynamic codes such as OpenFAST. With Craig-Bampton reduction the number of DOFs will be reduced to increase the efficiency [22]. This method was invented by Roy Craig and Mervyn Bampton [23].

The FE undamped equation of motion for a substructure is written as

$$[\mathbf{M}]_s\{\ddot{\mathbf{D}}\}_s + [\mathbf{K}]_s\{\mathbf{D}\}_s = \{\mathbf{R}^{\text{ext}}\}_s \tag{2.1.60}$$

where the subscript s indicates the s-th substructure. The equation of motion is then divided into interior and boundary DOFs indicated with subscript i and b, respectively:

$$\begin{bmatrix} \mathbf{M}_{bb} & \mathbf{M}_{bi} \\ \mathbf{M}_{ib} & \mathbf{M}_{ii} \end{bmatrix}_s \begin{Bmatrix} \ddot{\mathbf{D}}_b \\ \ddot{\mathbf{D}}_i \end{Bmatrix}_s + \begin{bmatrix} \mathbf{K}_{bb} & \mathbf{K}_{bi} \\ \mathbf{K}_{ib} & \mathbf{K}_{ii} \end{bmatrix}_s \begin{Bmatrix} \mathbf{D}_b \\ \mathbf{D}_i \end{Bmatrix}_s = \begin{Bmatrix} \mathbf{R}_b^{\text{ext}} \\ \mathbf{R}_i^{\text{ext}} \end{Bmatrix}_s \tag{2.1.61}$$

To statically eliminate all the interior DOFs from the model, the static response of the interior DOFs is calculated when one boundary DOF is given a unit displacement, while the response for the rest of the boundary DOFs are held fixed. This retains only the boundary DOFs, and the reduced system is of small size since the boundary DOFs are the only unknowns. The constraint modes are given by

$$[\mathbf{\Psi}]_s = \begin{bmatrix} \mathbf{I} \\ \mathbf{\Phi}_b \end{bmatrix}_s \tag{2.1.62}$$

where $[\mathbf{\Psi}]_s$ is called the Guyan modes, $[\mathbf{I}]$ is the identity matrix of size $b \times b$ and describes the unit displacements of each boundary DOF, and $[\mathbf{\Phi}_b]_s$ is calculated by considering the second row of the static part of equation (2.1.61).

$$[\mathbf{K}_{ib}]_s\{\mathbf{D}_b\}_s + [\mathbf{K}_{ii}]_s\{\mathbf{D}_i\}_s = \{\mathbf{0}\} \tag{2.1.63}$$

Rearranging yields:

$$\{\mathbf{D}_i\}_s = -[\mathbf{K}_{ii}]_s^{-1}[\mathbf{K}_{ib}]_s\{\mathbf{D}_b\} = [\mathbf{\Phi}_b]_s\{\mathbf{D}_b\} \tag{2.1.64}$$

$[\mathbf{\Phi}_b]_s$ then corresponds to the rigid body modes for an unconstrained structure. These are important to ensure that the interior nodes follow the boundary nodes for rigid body motion. To capture the dynamics of the system, the retained modes of the system are expanded to also include the dynamic modes. The dynamic modes are obtained by fixing the boundary DOFs and solving the eigenvalue problem:

$$([\mathbf{K}_{ii}]_s - \omega_i^2[\mathbf{M}_{ii}]_s)\{\boldsymbol{\phi}_i\}_s = \mathbf{0} \tag{2.1.65}$$

Where $\{\boldsymbol{\phi}_i\}_s$ is the basis for the reduced generalized modal DOFs $\{\mathbf{q}_m\}_s$. $\{\boldsymbol{\phi}_i\}_s$ is assumed mass normalized.

By then reducing the number of generalized DOFs to $m < i$ and sort the modes by increasing natural frequency, $\{\boldsymbol{\phi}_m\}_s$ becomes the truncated set of $\{\boldsymbol{\phi}_i\}_s$. The modes are collected into a retained fixed boundary mode matrix:

$$[\mathbf{\Theta}]_s = \begin{bmatrix} \mathbf{0} \\ \mathbf{\Phi}_m \end{bmatrix}_s \tag{2.1.66}$$

The Guyan modes and the retained fixed boundary modes are collected to form the C-B reduction matrix as:

$$[\mathbf{T}]_s = \begin{bmatrix} \mathbf{\Psi} & \mathbf{\Theta} \end{bmatrix}_s = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{\Phi}_i & \mathbf{\Phi}_m \end{bmatrix} \tag{2.1.67}$$

The C-B reduction matrix provides a transformation from the physical DOFs $\{\mathbf{D}\}_s$ to the C-B generalized modal DOFs $\{\mathbf{q}_m\}_s$:

$$\begin{Bmatrix} \mathbf{D}_b \\ \mathbf{D}_i \end{Bmatrix}_s \approx \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{\Phi}_i & \mathbf{\Phi}_m \end{bmatrix}_s \begin{Bmatrix} \mathbf{D}_b \\ \mathbf{q}_m \end{Bmatrix}_s \tag{2.1.68}$$

By applying the C-B transformation matrix to the mass and stiffness matrices the reduced mass and stiffness matrices are found:

$$\begin{aligned} [\mathbf{M}_{\text{red}}]_s &= [\mathbf{T}]_s^T[\mathbf{M}]_s[\mathbf{T}]_s \\ [\mathbf{K}_{\text{red}}]_s &= [\mathbf{T}]_s^T[\mathbf{K}]_s[\mathbf{T}]_s \end{aligned} \tag{2.1.69}$$

Which yields:

$$[\mathbf{K}_{\text{red}}]_s = \begin{bmatrix} \tilde{\mathbf{K}}_{bb} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}_m \end{bmatrix}_s$$
$$[\mathbf{M}_{\text{red}}]_s = \begin{bmatrix} \tilde{\mathbf{M}}_{bb} & \tilde{\mathbf{M}}_{mb} \\ \tilde{\mathbf{M}}_{bm} & \mathbf{I} \end{bmatrix}_s \tag{2.1.70}$$

Where $[\boldsymbol{\Omega}_m]_s$ is the diagonal matrix containing the squared natural frequencies of the fixed boundary modes. The sub-matrices $[\boldsymbol{\Omega}_m]_s$, $[\tilde{\mathbf{K}}_{bb}]_s$, $[\tilde{\mathbf{M}}_{bb}]_s$ and $[\tilde{\mathbf{M}}_{ib}]_s$ are calculated as (s subscript ignored for convenience):

$$[\boldsymbol{\Omega}_m] = [\boldsymbol{\Phi}_m]^T[\mathbf{K}_{ii}][\boldsymbol{\Phi}_m] \tag{2.1.71}$$

$$[\tilde{\mathbf{K}}_{bb}] = [\mathbf{K}_{bb}] - [\mathbf{K}_{bi}][\mathbf{K}_{bb}]^{-1}[\mathbf{K}_{bi}] \tag{2.1.72}$$

$$[\tilde{\mathbf{M}}_{bb}] = [\mathbf{M}_{bb}] - [\mathbf{M}_{bi}][\mathbf{K}_{ii}]^{-1}[\mathbf{K}_{ib}] - [\mathbf{K}_{bi}][\mathbf{K}_{ii}]^{-1}[\mathbf{M}_{ib}] + [\mathbf{K}_{bi}][\mathbf{K}_{ii}]^{-1}[\mathbf{M}_{ii}][\mathbf{K}_{ii}]^{-1}[\mathbf{K}_{ib}] \tag{2.1.73}$$

$$[\tilde{\mathbf{M}}_{mb}] = [\boldsymbol{\Phi}_m]^T([\mathbf{M}_{ib}] - [\mathbf{M}_{ii}][\mathbf{K}_{ii}]^{-1}[\mathbf{K}_{ib}]) = [\tilde{\mathbf{M}}_{bm}]^T \tag{2.1.74}$$

The C-B equation of motion now gets:

$$\begin{bmatrix} \tilde{\mathbf{M}}_{bb} & \tilde{\mathbf{M}}_{mb} \\ \tilde{\mathbf{M}}_{bm} & \mathbf{I} \end{bmatrix}_s \begin{Bmatrix} \ddot{\mathbf{D}}_b \\ \ddot{\mathbf{q}}_m \end{Bmatrix}_s + \begin{bmatrix} \tilde{\mathbf{K}}_{bb} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Omega}_m^2 \end{bmatrix}_s \begin{Bmatrix} \mathbf{D}_b \\ \mathbf{q}_m \end{Bmatrix}_s = \begin{Bmatrix} \mathbf{R}_b^{\text{ext}} \\ \mathbf{R}_m^{\text{ext}} \end{Bmatrix}_s \tag{2.1.75}$$

The equation (2.1.75) represents the equation of motion for the substructure after the C-B reduction where the number of DOFs are reduced from six times the number of nodes ($6 \times (i + b)$) to only the number of boundary nodes plus the number of generalized DOFs ($b + m$).

2.2 Earthquake

This section presents relevant theory of earthquakes and the effect of earthquakes on soil and structures. The theory is based on Kramer's book *Geotechnical earthquake engineering* [24].

2.2.1 Seismic action

Earthquakes are a naturally occurring phenomenon potentially leading to tremendous infrastructural damage and deadly outcome. Hazards associated with earthquake are commonly referred to as seismic hazards, and the most dramatic kind is those of structural collapse. This led to the importance of earthquake engineering; how to minimize material and human impact during earthquake excitation.

An earthquake produces seismic waves that radiate away from the source, traveling rapidly through the earth's crust. Waves reaching the ground surface give a shaking response of the ground, lasting from milliseconds to days with amplitudes ranging from nanometers to meters. This happens almost continuously, and the great majority of these vibrations are of a character that need specialized equipment to be measured. Such micro-seismic activity is of greater importance to seismologists than engineers. From an earthquake engineer perspective, so-called *strong ground motion* is the important activity. That means motion with sufficient strength to affect people and their environment. Strong ground motion usually last from 15 to 30 seconds [25].

Seismic waves travel most of the time through rigid rock, but often through softer soil layers before reaching the ground surface. Soil layers might both attenuate and amplify the ground motion, depending on the local soil deposit combined with the characteristics of the seismic waves reaching the site. Strong ground motion causing minimally response at one site, might produce devastating response at another. These local site effects as a result of the local geologic and soil conditions were described already in 1824 by J. MacMurdo's paper "Papers relating to the Earthquake which occurred in India in 1819" [26]. He writes inter alia "buildings situated upon rock were not by any means so much affected by the earthquake as those whose foundations did not reach the bottom of the soil".

The most important characteristics of strong ground motion is amplitude, frequency content and duration. As no earthquakes produces exactly the same motion, analysis of empirical data is an important study in order to develop a design ground motion for different areas. I.e., motions that reflect the levels of strong ground motion amplitude, frequency content and duration that a structure or facility at a particular site should be designed for. Thus, over the years considerable advances have been made to the earthquake-resistant design and the seismic design code requirements. The design focus has also changed from an emphasis on structural strength to emphases on both strength and ductility as the accuracy of the ground motion predictions have become better.

2.2.2 Seismic waves

Earthquakes produce different types of waves, where the two main types are body waves and surface waves. Body waves are those traveling through the interior of the earth, and surface waves obviously those traveling on the ground surface and in superficial layers of the earth. Surface waves arises from body waves interacting with surfaces. Of the two main types, there are also some sub types. Body waves are divided into two types; p- and s-waves, and the two most important surface waves are Rayleigh and Love waves.

P-waves propagates by successive compression and expansion of the traveling medium parallel to the traveling direction. P-waves may then also propagate through fluids, unlike the s-waves propagating by shear deformation normal to the traveling direction. This is due to the lack of shear strength in fluids, like water. The traveling velocity of body waves depends on the stiffness of the material, and since the geological materials are stiffest in compression, p-waves travel faster than other seismic waves and will arrive first at the site. This is also noticed by MacMurdo in his 1824 paper, where he describes the chairs lifting from the ground before the rest of the tremendous vibrations starts.

Rayleigh and Love waves are also propagating by shear deformation, Rayleigh waves with vertical or both vertical and horizontal particle movement, and Love waves with only horizontal particle movement of the surface. They travel along the surface with amplitudes decreasing roughly exponentially with depth.

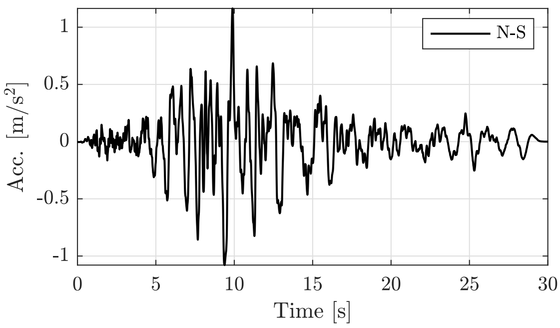
2.2.3 1989 Loma Prieta earthquake

Figure 2.6: Earthquake location. Map from Google Earth ©.

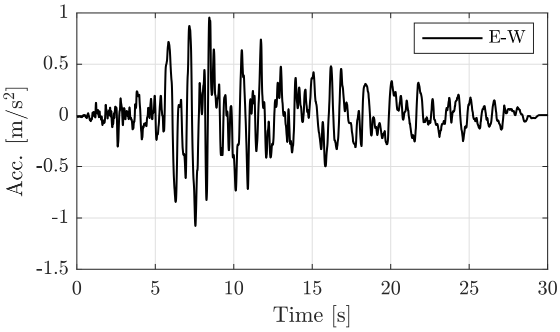
In October 1989, an earthquake occurred in the San Francisco area. The epicenter was located near

Mt. Loma Prieta, about 100 km south of San Francisco, see figure 2.6. In the epicentral area, an MMI VIII categorized earthquake intensity was registered, but in some areas in San Francisco even higher intensities (MMI IX) was registered. See table 2.1 for the relevant intensity descriptions. The San Francisco site is partly located on mud and partly on rock ground, and the effect of local site effects were clearly materialized in the different grade of structural damage. The earthquake caused extensive damage in certain areas and almost none in other neighbouring areas.

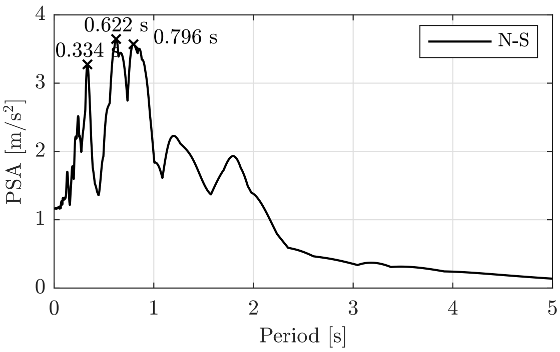
Both the epicentral area and the San Francisco area were well instrumented, and data from this earthquake is therefore well suited for analysis of different earthquake effects. In this project, recordings from the Menhaden Court, Foster City, is used. The recordings are downloaded from the PEER Strong Motion Database [27] and figure 2.7 shows the horizontal acceleration of the north-south (N-S) and the east-west (E-W) motion. The peak ground acceleration (PGA) is 1.164m/s^2 for the N-S direction and 0.955m/s^2 in the E-W direction.



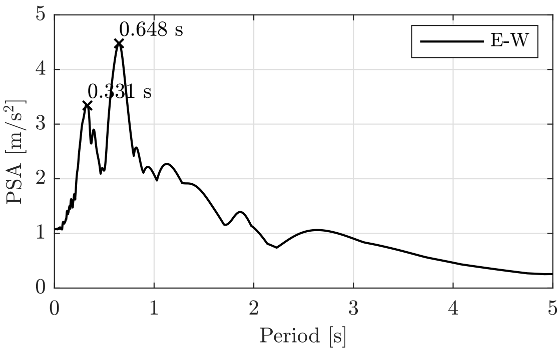
(a) N-S accelerations. PGA = 1.1640m/s^2



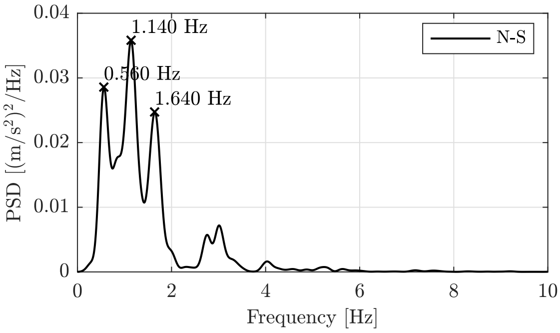
(b) E-W accelerations. PGA = 0.9550m/s^2



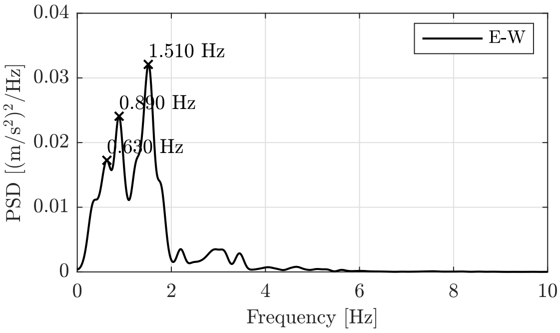
(c) N-S PSA for 5% damping ratio



(d) E-W PSA for 5% damping ratio



(e) N-S PSD



(f) E-W PSD

Figure 2.7: Loma Prieta earthquake recorded ground acceleration at Menhaden Court, Foster City, 18th of October 1989. (a) and (b) showing acceleration time series, (c) and (d) showing Pseudo-acceleration response spectra, (e) and (f) showing Power spectral density of the acceleration computed using Welch method and Hamming window.

The frequency content of the accelerations, shown in figure 2.7 (e) and (f), is of importance when conducting the FE analysis. The wavelength of the seismic waves propagating through soil layers is related to the frequency of the earthquake accelerations, and the highest frequency determines the needed FE mesh size. This is discussed further in section 2.2.5. For this earthquake, 6 Hz is defined as the highest frequency of interest for both directions.

Table 2.1: Earthquake intensity scale description

Modified Mercalli Intensity Scale of 1931 - MMI

Grade	Description
VIII	Damage slight in specially design structures, considerable in ordinary substantial buildings, with partial collapse, great in poorly built structures; panel walls thrown out of frame structures; fall of chimneys, factory stacks, columns, monuments, walls; heavy furniture overturned; sand and mud ejected in small amounts; changes in well water; persons driving motor cars disturbed.
IX	Damage considerable in specially designed structures; well-designed frame structures thrown out of plumb; great in substantial buildings, with partial collapse; buildings shifted off foundations; ground cracked conspicuously; underground pipes broken.

2.2.4 Systems subjected to seismic loading

Structures exposed to ground motion will experience inertia as the ground accelerates the structure mass. The acceleration of the structure mass will then induce a structural displacement, u, if the mass is able to move relative to the ground. The total displacement, u_t of the mass then becomes

$$u_t = u_g + u \tag{2.2.1}$$

where u_g denotes the ground displacement. The relative displacement will then lead to internal forces, and the total acceleration $\ddot{u}_t = \ddot{u}_g + \ddot{u}$ will give the total inertia. With a classical interpretation of damping, the damping forces will relate to the relative velocity, $\dot{u}$. By Newton's second law of motion, the equation of motion of a SDOF system then becomes

$$c\dot{u} + ku = m(\ddot{u} + \ddot{u}_g) \Rightarrow m\ddot{u}_g + c\dot{u} + ku = -m\ddot{u}_g \tag{2.2.2}$$

A simple SDOF system exposed to ground motion is visualized in figure 2.8.

The same yields for a MDOF system, and the equation of motion then becomes

$$[\mathbf{M}]\{\ddot{\mathbf{u}}\} + [\mathbf{C}]\{\dot{\mathbf{u}}\} + [\mathbf{K}]\{\mathbf{u}\} = -[\mathbf{M}]\{\boldsymbol{\iota}\}\ddot{u}_g \tag{2.2.3}$$

where $\{\boldsymbol{\iota}\}$ is an influence vector, describing how the DOFs are influenced by a unit displacement of the ground.

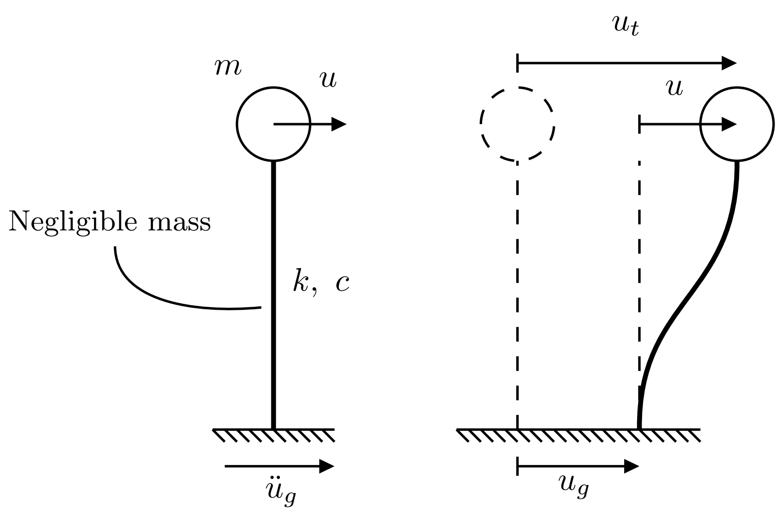


Figure 2.8: A SDOF system subjected to earthquake

2.2.5 Wave propagation

For the case of seismic waves propagating in soil layers, the horizontal shear waves propagating vertically from the bedrock will now be discussed. Thus, considering a one-dimensional analysis of a soil layer extending infinitely in the horizontal direction, with all boundaries assumed horizontal. The system is shown in figure 2.9. To obtain different response quantities, the use of transfer function may apply. This relies on the principles of superposition and is therefore limited to linear analysis. The approach is, however, useful for verifying the behaviour of FE models of a soil deposit.

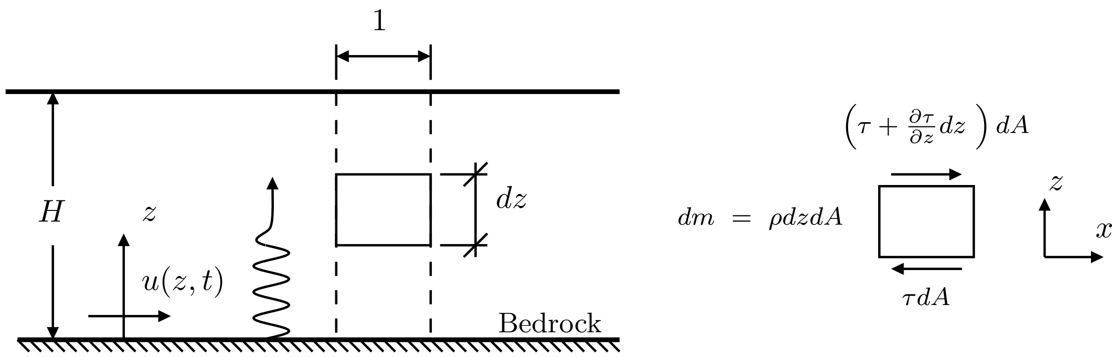


Figure 2.9: Vertically propagating shear waves in uniform soil medium.

For a linear elastic soil deposit situated up on bedrock, horizontal harmonic motion of the bedrock will result in vertically propagating shear waves throughout the soil layer. The horizontal motion in the soil layer can easily be expressed by the horizontal resultant force on the infinitesimal element shown in figure 2.9 and Newton's second law of motion, neglecting damping forces:

$$\sum F_x = m\ddot{u} \Rightarrow \frac{\partial \tau}{\partial z} = \rho \ddot{u} \tag{2.2.4}$$

Introducing the relations

$$\tau = G\gamma = G\frac{\partial u}{\partial z} \quad \text{and} \quad v_s = \sqrt{\frac{G}{\rho}} \tag{2.2.5}$$

where G is the shear modulus, γ is shear strain and ρ is the density, gives the one dimensional wave equation

$$\frac{\partial^2 u}{\partial z^2} - \frac{1}{v_s^2}\ddot{u} = 0 \tag{2.2.6}$$

where v_s denotes the shear wave velocity. The response is assumed on the form

$$u(z,t) = \bar{u}(z)\cos(\omega t) \tag{2.2.7}$$

where ω is the excitation circular frequency. Substituting into equation (2.2.6) gives

$$\frac{\partial^2 \bar{u}}{\partial z^2}\cos(\omega t) + \left(\frac{\omega}{v_s}\right)^2 \cos(\omega t)\bar{u} = 0 \Rightarrow \frac{\partial^2 \bar{u}}{\partial z^2} + \left(\frac{\omega}{v_s}\right)^2 \bar{u} = 0 \tag{2.2.8}$$

Such a second order differential equation has the solution

$$\bar{u}(z) = A\cos\left(\frac{\omega}{v_s}z\right) + B\sin\left(\frac{\omega}{v_s}z\right) \tag{2.2.9}$$

Introducing the boundary conditions

$$\bar{u}(0) = u_0 \quad \text{and} \quad \tau(H) = G\frac{\partial \bar{u}}{\partial z}\Big|_{z=H}\cos(\omega t) = 0 \tag{2.2.10}$$

then gives

$$\bar{u}(z) = u_0\cos\left(\frac{\omega}{v_s}z\right) + u_0\tan\left(\frac{\omega H}{v_s}\right)\sin\left(\frac{\omega}{v_s}z\right) \tag{2.2.11}$$

To quantify the amplification of the free surface displacement relative to the bedrock displacement, equation (2.2.11) can be used to produce a transfer function, $H(\omega)$:

$$H(\omega) = \frac{|u(H,t)|}{|u(0,t)|} = \frac{|\bar{u}(H)|}{|\bar{u}(0)|} = \frac{1}{|\cos(\omega H/v_s)|} \tag{2.2.12}$$

This equation shows that the displacement amplitude of the free surface always is equal to or greater than the displacement amplitude of the bedrock. As illustrated in figure 2.10, the denominator of equation (2.2.12) approaches zero when $\omega H/v_s$ goes towards $\pi/2 \cdot (2n-1)$ for $n = 1,\ 2,\ 3,\ ...$, and the amplification then goes towards infinity. This emphasises the importance of the site effects mentioned in section 2.2.1; excitation frequency at the base combined with the given soil properties have a major effect on the resulting surface response.

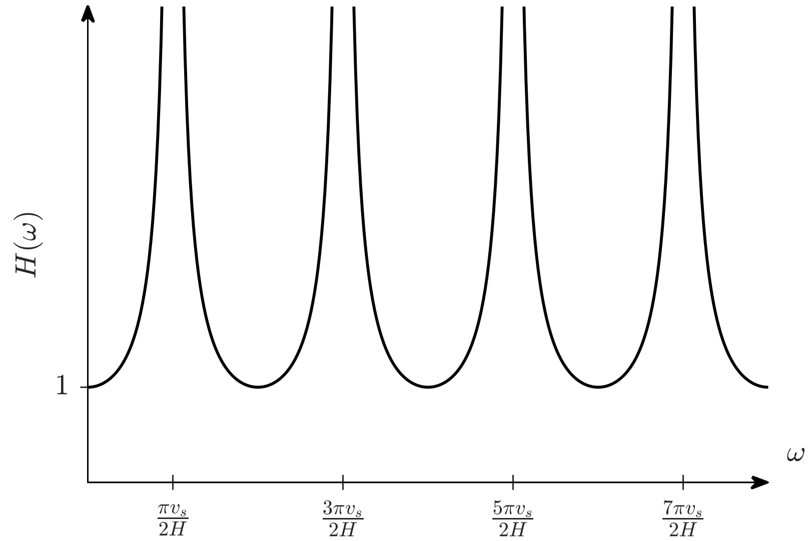


Figure 2.10: Amplification of harmonic base motion for undamped soil.

The transfer function indirectly gives the soil layer natural frequencies, ω_n, by considering the resonance frequencies. The infinite amplification frequencies indicating resonance, thus, the natural frequencies of the soil:

$$\omega_n = \frac{\pi v_s}{2H}(2n-1), \quad n \in \mathbb{N} \tag{2.2.13}$$

The mode shapes, ϕ_n, is then found by considering zero ground displacement, $\bar{u}(z=0) = 0$, giving $A = 0$ in equation (2.2.9). Evaluating at the natural formally reveals the mode shape:

$$\bar{u}(z) = B\sin(\frac{\omega_n}{v_s}z) = B\sin(\frac{\pi z}{2H}(2n-1)) \Rightarrow \phi_n = \sin(\frac{\pi z}{2H}(2n-1)) \tag{2.2.14}$$

The infinite amplification is of course not realistic as there always exist some damping which will reduce it. Repeating the procedure above assuming Kelvin-Voigt shearing characteristics of the soil, gives the damped wave equation:

$$\rho\frac{\partial^2 u}{\partial t^2} = G\frac{\partial^2 u}{\partial z^2} + \eta\frac{\partial^3 u}{\partial z^2 \partial t} \tag{2.2.15}$$

The solution of this equation is on the form

$$u(z,t) = Ae^{i(\omega t + k^* z)} + Be^{i(\omega t - k^* z)} \tag{2.2.16}$$

where k^* is a complex wave number. From this, the transfer function for an undamped soil layer becomes

$$H(\omega) = \frac{1}{\cos(k^* H)} = \frac{1}{\cos(\omega H/v_s^*)} \tag{2.2.17}$$

where the complex shear wave velocity, v_s^*, for small damping ratios, ζ, can be expressed as

$$v_s^* = \sqrt{\frac{G^*}{\rho}} = \sqrt{\frac{G(1+i2\zeta)}{\rho}} \approx \sqrt{\frac{G}{\rho}}(1+i\zeta) = v_s(1+i\zeta) \tag{2.2.18}$$

Applying this to the transfer function and utilizing the $|\cos(x+iy)| = \sqrt{\cos^2 x + \sinh^2 y}$ identity and $\sinh^2 y \approx y^2$ for small values of y gives

$$H(\omega) = \frac{1}{\sqrt{\cos^2(\omega H/v_s) + [\zeta(\omega H/v_s)]^2}} \tag{2.2.19}$$

The amplification from this transfer function is shown in figure 2.11 for several damping values.

Worth noticing is that the wavelength of the propagating shear wave, λ, becomes

$$\lambda = \frac{2\pi}{\omega} v_s \tag{2.2.20}$$

I.e., the wave length is related to the soil properties by the shear wave velocity, v_s, and to the earthquake loading by the loading frequency, ω. In a FE discretization of a soil deposit, the FE mesh then needs to be fine enough to capture the wavelength. The maximum characteristic element length, L_e, should therefore be limited to 1/8 of shortest wavelength, determined by the highest frequency:

$$L_{\mathrm{e,max}} = \frac{2\pi v_s}{8\omega_{\max}} \tag{2.2.21}$$

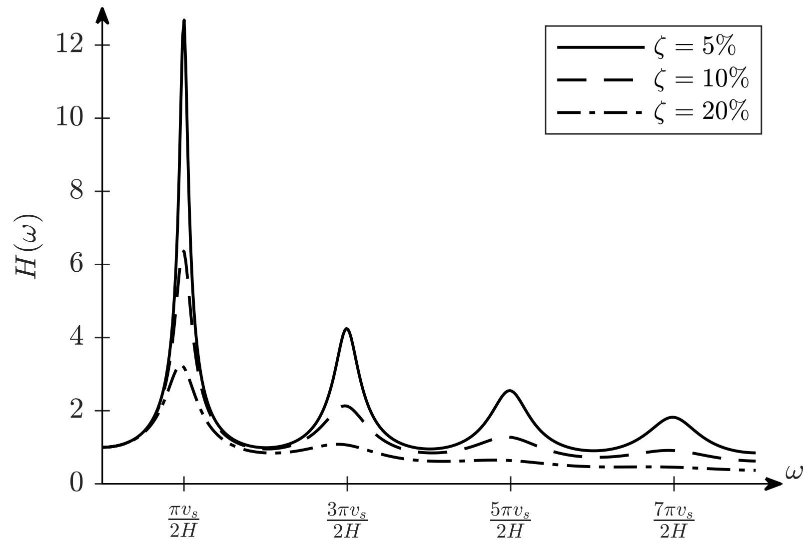
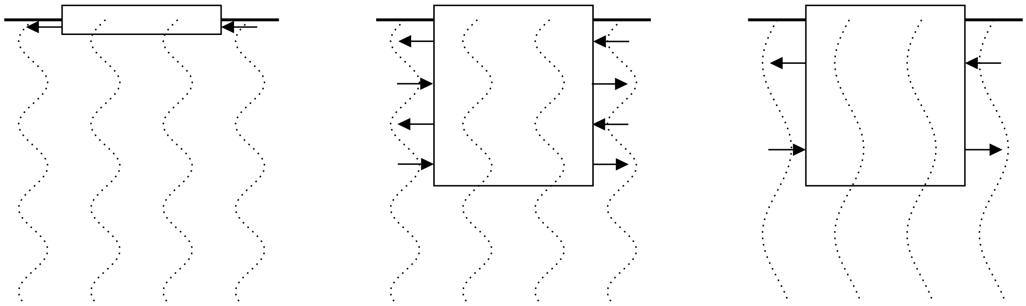


Figure 2.11: Amplification of harmonic base motion for damped soil

2.2.6 Soil-structure interaction

Ground motion not influenced by the presence of structures are referred to as *free field* motion. A structure situated up on solid rock will experience ground motion very close to the free field motion due to the extremely high stiffness of rock. The structure may then be treated with a fixed base, and response computations becomes relatively simple. The same structure situated up on a soft soil deposit, on the other hand, would respond differently. The inability of the foundation to conform to the free field deformations of the soil will make the actual ground motion to vary from the free field motion and the dynamic response of the structure will affect the motion of the soil foundation. This two-way interaction is referred to as soil-structure interaction (SSI). Computation including this phenomenon becomes quite complicated compared to a fixed base structure case.

The SSI effect on dynamic response of structures and foundations may be little for some systems, and severe for others. Whether neglecting the SSI effect is conservative or non-conservative must be evaluated on a case-by-case basis. The stiff structure foundation impeding the free field motion of the soil is referred to as *kinematic interaction*. The embedded depth of the foundation in combination with the wavelength, is important for the extent to which the kinematic interaction leads to rocking of the structure. The latter is illustrated in figure 2.12: Foundation (a) is rather shallow, experiencing a very small resulting moment from the horizontal forces. Foundation (b) is exposed to a shorter wavelength than foundation (c), and the horizontal forces will in a greater extent outbalance each other, resulting in a smaller resulting moment than foundation (c).



(a) Shallow foundation **(b)** Short wave length **(c)** Long wave length

Figure 2.12: Kinematic interaction rocking effect from vertical propagating shear waves with different wave lengths on foundations with different embedment depths.

As a soft soil layer will decrease the overall stiffness of the soil-structure system, the structure will experience a reduction of the natural frequency, ω_n. If an SDOF system excited by horizontal ground motion is considered, the reduced natural frequency becomes

$$\frac{1}{\omega_{eq}^2} = \frac{1}{\omega_n^2} + \frac{1}{\omega^2} \tag{2.2.22}$$

where ω_{eq} denotes the combined system natural frequency and ω is the horizontal excitation frequency.

Methods for including the SSI effects are divided into two main categories: *direct methods* and *multi-step methods*. Multi-step methods relay on the principles of superpositioning, and are therefore limited to linear systems, where direct methods consider a one step analysis of an *integrated model*. An integrated model means a FE model including both the structure and a soil deposit. Apparently, it is not possible to discretize the semi-infinite soil domain with a finite number of elements. The soil domain is therefore truncated by introducing artificial boundaries. This leads to the main challenge of the FE SSI analysis; how to model the boundaries to represent an adequate behaviour of the response. The boundaries must have the ability of transmitting energy of waves approaching and leaving the finite domain. Especially, if the boundaries are not able to transmit energy out of the model, accumulation of energy in the model may significantly disturb the results.

Several methods are used for this purpose, and a common approach is assuming the bedrock boundary less important, modelling it as either elastic, represented by linear springs, or rigid. The earthquake motion may then be applied directly to the bedrock boundary as accelerations or displacements. For the lateral boundaries, a larger selection of methods applies [28][29]:

- Free boundary
- Viscous-spring (VS) boundary
- Free field loading combined with viscous dashpots-spring (FFL-VS) boundary
- Tied degree of freedom (TDOF) boundary
- Perfeclty-matched-layers (PML) boundary
- Domain reduction (DMR)

How the methods are applied is also a concern. E.g., the VS and FFL-VS boundaries includes the application of springs and dashpots, and for what directions these are applied will affect how waves propagating in different directions are handled. An appropriate location of the boundaries is, obviously, of great importance as well, and tends to be a question of computational capacity/-efficiency.

In the case of SSI and OWT structures, one of the concerns is that plastic strains in the soil, due to interaction with the OWT substructure foundation, may lead to settlements and permanent tilting of the structure [10]. To evaluate such problems, a direct method including soil non-linearity needs to be run. One approach is then to implementing *Mohr-Coulomb plasticity* in the soil model.

2.2.7 Mohr-Coulomb placticity

In order to include plastic behaviour of a material, a plasticity model needs to be defined. For materials which the compressive strength far exceeds the tensile strength, such as typical soil materials, the *Mohr-Coulomb plasticity criterion* model is suited. The model is a non-associative, elastic-perfectly plastic model, where the elastic behaviour follows Hooke's law defined by Young's modulus, E, and the Poisson's ratio, ν. The plastic behaviour follows a given flow rule, Q, and the yield criterion, F, is linearly dependent on the normal stress in the same plane.

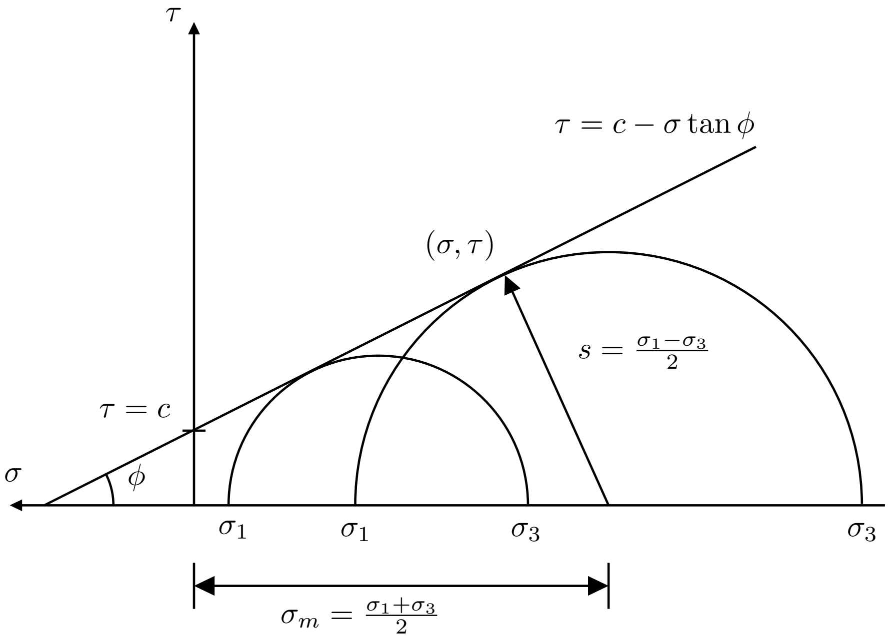


Figure 2.13: Illustrating basis of the Mohr-Coulomb criterion. σ is negative in compression.

The yield criterion is based on plotting Mohr's circle for states of stress where yielding occurs in the plane of the maximum and minimum principal stresses. The yield line is then the line best fitted to touch the circles. This is illustrated in figure 2.13 and gives the yield line defined as

$$\tau = c - \sigma \tan\phi \tag{2.2.23}$$

where τ is the shear stress, σ is the normal stress defined as negative in compression, c is the cohesion of the material, and ϕ is the material angle of friction. From Mohr's circle;

$$\tau = s\cos\phi \tag{2.2.24}$$

$$\sigma = \sigma_m + s\sin\phi \tag{2.2.25}$$

Substituting this into equation (2.2.23), the Mohr-Coulomb criterion can be rewritten as

$$F = s + \sigma_m \sin\phi - c\cos\phi = 0 \tag{2.2.26}$$

The principle of a yield criterion is as follows:

$$\begin{aligned} F < 0: &\quad \text{No yielding - elastic behaviour} \\ F = 0: &\quad \text{Yielding} \\ F > 0: &\quad \text{Physically impossible} \end{aligned}$$

I.e., during plastic flow, stresses remain on the yield surface. How the material behaves when yielding, is then determined by a flow rule [30]. The flow rule describes how plastic strains develop in terms of a *plastic potential* or *flow potential*, Q. For associative models, the flow potential is the same as the yield criterion, but for non-associative models, an own potential is defined. Associative models relates to ductile materials, while non-associative relates to granular materials. The Mohr-Coulomb flow potential includes the ψ-parameter, referred to as the *angle of dilation*, and controls the plastic volumetric strain rate during yielding. The expression describing the flow potential can be found in [31].

2.3 Environmental Conditions

The offshore wind turbine is located far out in the ocean where there are several environmental conditions due to waves, hydrodynamics and wind. The theory behind these conditions is presented in the forthcoming subsections.

2.3.1 Random waves and wave spectra

Ocean waves are best described by a random wave model due to their random and irregular behaviour. One such model is the linear random wave model which is a sum of a finite number of linear wave components. The simplest wave theory is the linear wave theory, also known as the Airy theory. In linear wave theory the wave height is much smaller than the wavelength and the water depth, and the wave shape is assumed to be sinusoidal [32]. The surface elevation in linear wave theory is given as:

$$\eta(x, y, t) = \frac{H}{2} \cdot \cos\theta \tag{2.3.1}$$

where

$$\theta = k(x\cos\beta + y\sin\beta) - \omega t \tag{2.3.2}$$

is the phase angle and β is the direction of propagation measured from the positive x-axis.

The random and irregular ocean sea states can be modelled as a summation of a finite number of linear waves. This is the simplest random wave model and is given by:

$$\eta(t) = \sum_{k=1}^{N} A_k \cos(\omega_k t + \epsilon_k) \tag{2.3.3}$$

where ϵ_k are random phases uniformly distributed between 0 and 2π. A_k are the random amplitudes and are Rayleigh distributed with mean square given as:

$$E[A_k^2] = 2S(\omega_k)\delta\omega_k \tag{2.3.4}$$

$S(\omega)$ is the wave spectrum and $\delta\omega_k$ is the difference between successive frequencies. The wave spectrum describes how the energy of the waves are distributed among different frequencies for a sea state. The Pierson-Moskowitz (PM) spectrum is one such spectrum and is used to describing fully developed sea states[32]. The PM spectrum is given by:

$$S_{PM} = \frac{5}{16} H_s^2 \omega_p^4 \omega^{-5} exp\Big(-\frac{5}{4}\Big(\frac{\omega}{\omega_p}\Big)^{-4}\Big) \tag{2.3.5}$$

where H_s is the significant wave height and ω_p is the angular spectral peak frequency. For developing sea states the PM spectrum is modified to form the JONSWAP spectrum defined as:

$$S_J(\omega) = A_\gamma S_{PM}(\omega)\gamma^{exp\left(-\frac{1}{2}\left(\frac{\omega-\omega_p}{\sigma\omega_p}\right)^2\right)} \tag{2.3.6}$$

where $A_\gamma = 1 - 0.287\ln(\gamma)$ is a normalizing factor, γ is a non-dimensional peak shape parameter and σ is the spectral width parameter. For γ equal to 1 the JONSWAP spectrum will be equal to the Pierson-Moskowitz spectrum. The spectral width parameter σ is equal to 0.07 when $\omega \leq \omega_p$ and σ is equal to 0.09 when $\omega > \omega_p$. The effect of the peak shape parameter γ is shown in figure 2.14.

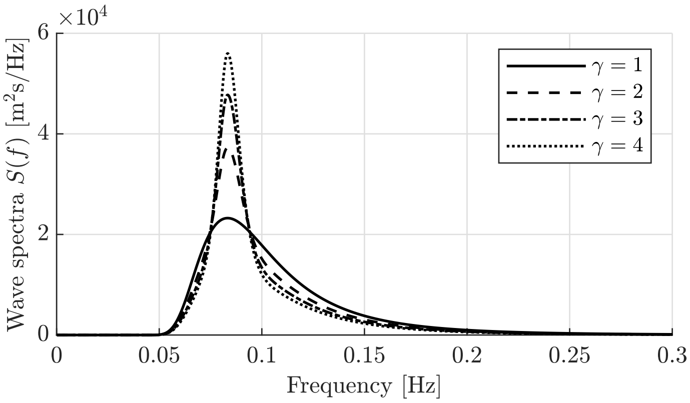


Figure 2.14: The effect from the values of the peak shape parameter γ

2.3.2 Fluid-structure interaction

Fluid-structure interaction (FSI) is important for structures placed in a fluid, such as offshore structures. The path of the fluid flow field is altered by the presence of the structure in the fluid. For offshore structures in a real sea state; the structures experience large oscillating forces in the direction of the flow. The forces on structures in a fluid flow field are characterised as drag and lift. Drag forces are acting in line of the direction of the flow while the lift forces are transverse to the direction of the flow [25].

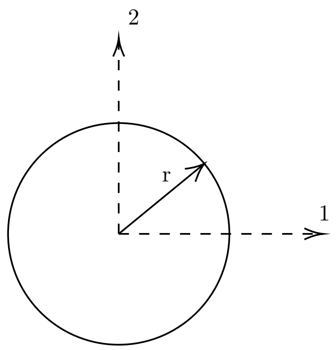
Added mass
The inertia forces for accelerated bodies surrounded by a fluid are larger than for bodies in vacuum. The reason for the larger inertia forces is due to that the surrounding fluid is accelerated along with the body, which effectively adds to the total mass of the system. The added mass can be interpreted as a volume of fluid particles that is accelerated together with the body it surrounds. The particles will however be accelerating with a varying degree, which depends on the distance relative to the body. The added mass is a weighted integration of the mass of all these fluid particles [33].

When the body is an elongated cylinder with a simple geometrical shape, the added mass coefficients can be approximated by a strip theory synthesis, where the flow at each section is assumed to be locally two dimensional. Some underlying assumptions is that the dimension out of plane is large compared to the in-plane dimensions, the body is stiff and that the body is in an approximately infinite fluid. The last assumption requires only that the body is relatively small compared to the fluid, as no fluid is truly infinite.

The simplest example of added mass is for circular sections. It can be shown that the added mass for a circular section is equal to the mass of the displaced fluid as can be seen in figure 2.15.

Since the added mass increases the total mass of the body, it will also change the natural frequencies of the body located in the fluid, which for a SDOF system is given as:

$$\omega_n = \sqrt{\frac{k_{tot}}{m_{tot}}} \quad \implies \quad \omega_n = \sqrt{\frac{k_{tot}}{m_{\text{structure}} + m_{\text{added mass}}}} \tag{2.3.7}$$



$m_{11} = \pi\rho r^2$

$m_{22} = \pi\rho r^2$

Figure 2.15: 2D added mass for a circle

Hydrodynamic damping

Hydrodynamic damping in civil engineering structures comes from three different contributions; viscous damping, radiation damping and skin friction damping [34]. The three types have different contributions for different situations.

The viscous damping take place whenever a body vibrates in a viscous medium. A viscous medium is any fluid or gas that offers a resistance to the motion of the different layers in the medium. This property is referred to as viscosity and is a measure of a fluids resistance to the shear strain rate. The viscous damping applies to any vibrating civil engineering structure, such as an oil platform, a bridge, wind turbine etc. The damping is caused by vortex shedding and this contribution can be found in the drag term of Morison's equation:

$$F_d = \frac{1}{2}\rho C_d A v^2 \tag{2.3.8}$$

where ρ is the mass density of the fluid, C_d is the drag coefficient, A is the reference area and v is the flow velocity relative to the structure. The drag force on a structure could also be an exciting force. This happens when the water particle motions are large, such as near the surface when there are waves. Deeper into the ocean where the wave effects are small, due to the decay function of waves, the drag force will be a damping force.

The radiation damping is caused by waves radiating out from an oscillating structure. The damping occurs when vibration energy dissipates into the fluid when the structure vibrates and creates waves.

The skin friction damping is caused by shear forces between the fluid and the surface of the structure. This damping is of different nature than the drag and radiation damping, and the skin friction damping is usually very small. For long vertical cylinders oscillating with a vertical small amplitude the skin friction damping could be significant.

2.3.3 Wind and wind spectra

Wind is generally caused by three things; the heating of the atmosphere by the sun, the earth's rotation and the irregularities of the earth's surface like mountains, forests, etc. [35]. The temperature gradient causes a difference in atmospheric pressure where the warmer air gets a lower density, which results in a lower pressure. This causes the warm, low density air to rise and the cold, high density and high pressure air to travel horizontally underneath the hot air. The greater the difference in atmospheric pressure is, the stronger the wind gets. The earth's rotation causes the wind to deflect and not blow directly into low pressure zone and this effect is called the Coriolis effect. The fact that the earth's surface is irregular exerts a horizontal friction force on the moving air. This forms the atmospheric boundary layer where the flow is irregular and turbulent. Within this boundary layer the wind speed increases with elevation, and the shape of the velocity profile is decided by the roughness of the earth's surface [36].

The turbulence in the wind causes the wind speeds to be random in time. The total wind speed can be divided into a sum of a mean component and a fluctuating turbulence component in three orthogonal directions.

$$U(t) = \bar{U} + u(t) \tag{2.3.9}$$

where $U(t)$ is the wind speed, $\bar{U}$ is the mean velocity and $u(t)$ is the fluctuating turbulence as can be seen in figure 2.16.

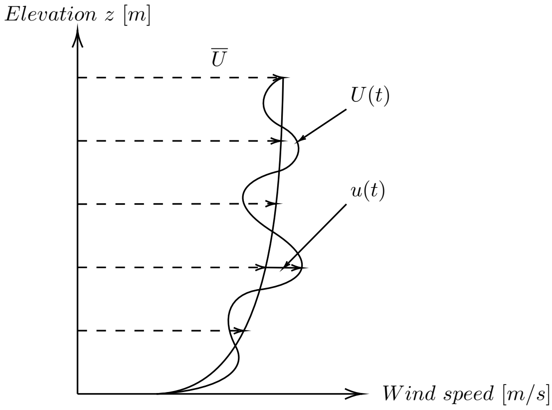


Figure 2.16: Wind speed profile for the boundary layer showing the total wind speed profile $U(t)$, the mean speed profile $\bar{U}$ and the turbulence speed profile $u(t)$

The mean wind speed profile shown in figure 2.16 follows the logarithmic law given as:

$$\bar{U} = \frac{u_*}{k} \ln\left(\frac{z}{z_0}\right) \tag{2.3.10}$$

where the u_* is the friction velocity, k is the von Karman constant, z is the elevation and z_0 is the surface roughness height.

The wind turbulence fluctuates with several frequencies, and these are shown in the Van der Hoven spectrum shown in figure 2.17.

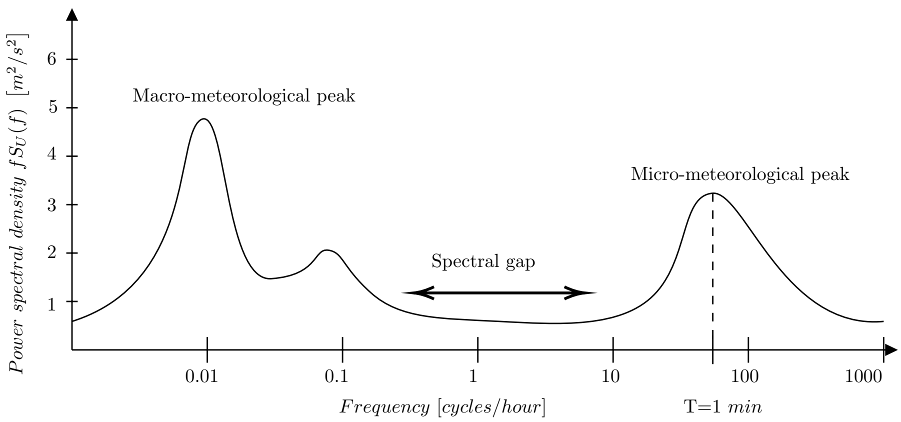


Figure 2.17: Van der Hoven spectrum of wind speeds in a wide frequency range. The micro-meteorological peak has a period of around 1 minute.

The macro-meteorological peak represents the large-scale global wind moments while the micro-meteorological peak represents the turbulence of the wind speed, which depends on topology, terrain surface and obstacles. For civil engineering structures, it is the fluctuations in the micro-meteorological peak that is of interest. This peak is represented by several expressions for the spectral density. The spectral density explained here are the Kaimal type and the others are deemed outside the scope of this thesis.

The IEC Kaimal spectra for the three components (u, v, w) used by TurbSim to generate a full wind field is given as [37]:

$$S_k(f) = \sigma_k^2 \frac{4\frac{L_k}{\bar{u}_{hub}}}{\left(1 + 6\frac{f \cdot L_k}{\bar{u}_{hub}}\right)^{\frac{5}{3}}} \tag{2.3.11}$$

where f is the frequency, σ_k is the standard deviation about the mean velocity in each direction, L_k is an integral length scale, $\bar{u}_{hub}$ is the mean velocity at the hub. The IEC defines the standard deviation as:

$$\begin{aligned} L_u &= 8.10\Lambda_U \\ L_v &= 2.70\Lambda_U \\ L_w &= 0.66\Lambda_U \end{aligned} \tag{2.3.12}$$

where Λ_U is the turbulence scale parameter defined in the third edition of the IEC 61400-1 as:

$$\Lambda_U = 0.7 \cdot z_{hub} \tag{2.3.13}$$

where z_{hub} is the elevation of the turbine hub above sea level. The relationship between the different standard deviations used in (2.3.11) are:

$$\begin{aligned} \sigma_v &= 0.8\sigma_u \\ \sigma_w &= 0.5\sigma_u \end{aligned} \tag{2.3.14}$$

3 Abaqus model

This section describes the FE model of the reference OWT modelled in the FE program Abaqus [12]. The model is intended to handle SSI effects, also non-linear soil dynamics, and is also used to verify the OpenFAST model described in section 4. The model consists of five main parts; (1) tower and RNA, (2) transition piece, (3) jacket, (4) piles and (5) soil, and the full assembly is referred to as an *integrated model*. To avoid complex operations in the graphical user interface, and to make the model, in some extent parametric, Python-scripts are made for building the model [38][39]. To keep an organized overview of the model parameters, an Excel spreadsheet is used. Including the parameters into the Python-scripts utilizes csv-files. The python-scripts and the csv-files are added to appendix C, and all the parameters used are described in the further sections. The model global coordinate system has its z-axis along the tower center axis and the x-y-plane lies at the mean sea level.

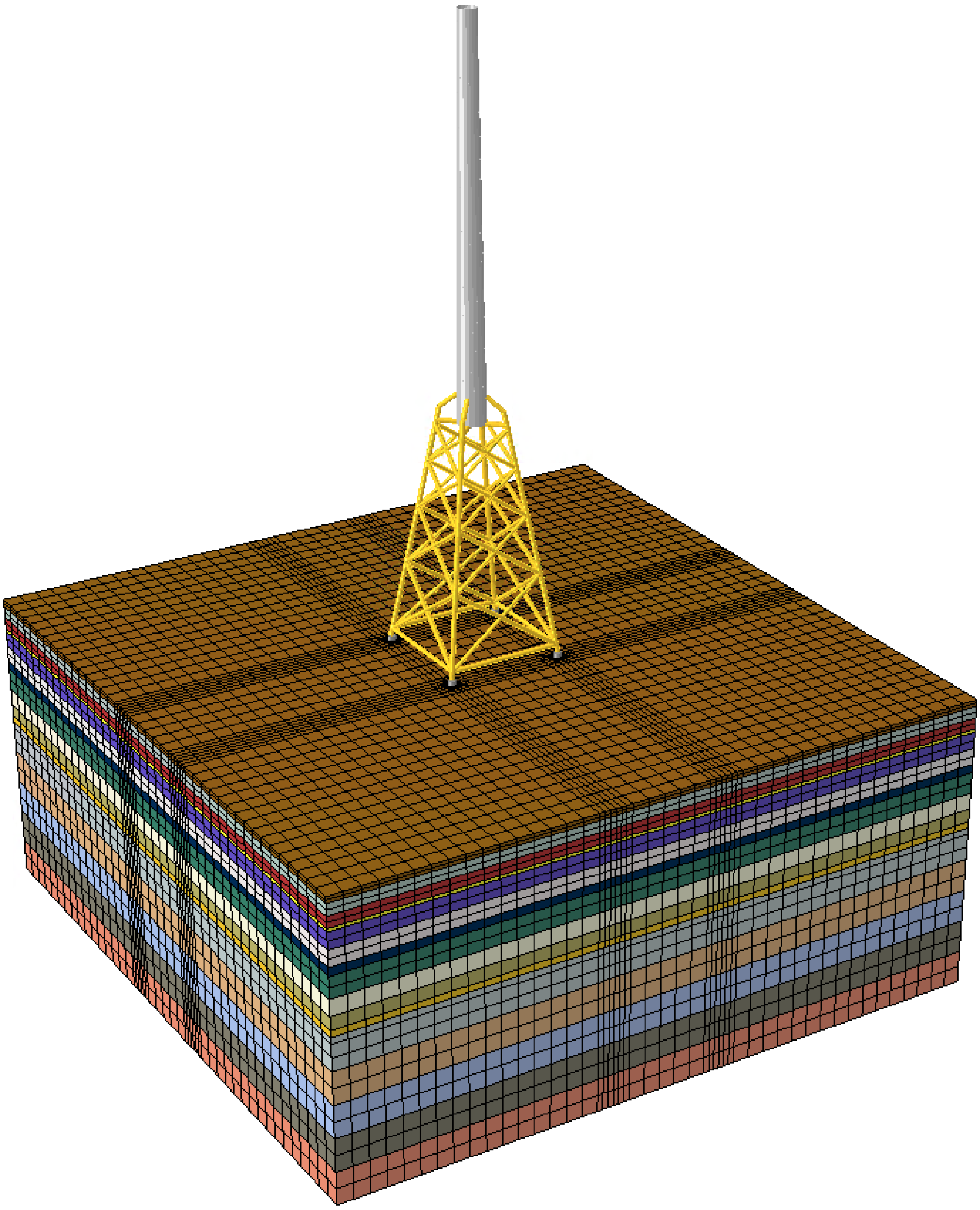


Figure 3.1: Visualization of the integrated model. Different colors in the soil showing different soil layers.

Abaqus gives a lot of opportunities to include different kind of loads and behaviour. E.g., implementation of environmental loads is highly possible through the *Abaqus/Aqua* toolbox [40]. Tentatively implementation of this toolbox is made and found working, but no results or methodology of the application will be presented in this thesis. This remains for further work. However, the main goal of investigating the effect of soil non-linearities is adapted by including *Mohr-Coulomb plasticity* in the soil material. The next sections will describe the modeling of each part, the assembly technique, damping assessment, choice of boundary conditions, verification of the soil part, and presentation of the model's dynamic properties.

3.1 Tower and RNA

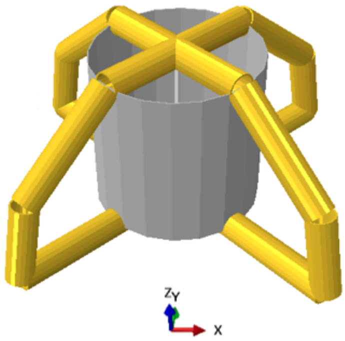
The tower is modelled with a tapered pipe section, with a linear varying outer diameter and wall-thickness. Linear interpolated Timoshenko beam elements (B31) are used, and due to that Abaqus can not handle a tapered pipe cross-section along with the chosen beam element, a modelling technique with a discretization of the tower in n sections with decreasing (from bottom to top) profile radius and wall-thickness is used. The middle values for each section is chosen. By choosing an appropriate discretization number ($n \geq 100$) the geometry is well represented. The RNA is handled as a point mass at the tower top. The RNA shape and eccentricity is taken care of by applying its mass moment of inertia to the point mass. The inertias are with respect to the global axis-directions at the tower top. The tower material's mass proportional damping coefficient, α, is also applied to the RNA point mass. The bottom tower piece surrounded by the transition piece is excluded from the tower model, and included in the transition piece model. Table 3.1 summarizes the parameters used for the tower and RNA model, appendix A.2 summarizes the different RNA components mass and inertia, and a detailed description of the RNA can be found at the IEA report [6].

Table 3.1: Summary of the tower and RNA model parameters. Note that the tower bottom is at the intersection with the transition piece level.

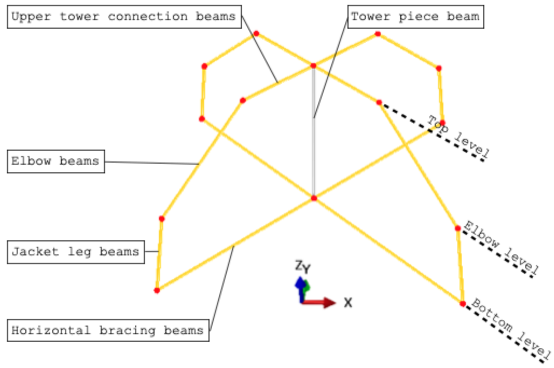
Parameter	Value
Tower top z-coordinate [m]	131.63
Tower bottom z-coordinate [m]	26.00
Tower top diameter [m]	5.50
Tower bottom diameter [m]	8.30
Tower top wall-thickness [m]	0.03
Tower bottom wall-thickness [m]	0.07
Discretization number, n [-]	160
Young's modulus [GPa]	210.0
Poisson's ratio [-]	0.3
Tower mass density [kg/m^3]	8500.0
Rayleigh α coefficient [-]	0.159
Rayleigh β coefficient [-]	0.006
RNA total mass [kg]	866 555
RNA total mass moment of inertia, I_{xx} [kg m^2]	240 016 659
RNA total mass moment of inertia, I_{yy} [kg m^2]	142 102 115
RNA total mass moment of inertia, I_{zz} [kg m^2]	111 846 413

3.2 Transition piece

The transition piece is a simple strutted beam design, basically elongating the jacket legs for connection to the tower. A transition piece is a critical part of such a construction due to the transmission of large forces and moments, and should be like a fixed joint between the jacket and the tower. To achieve the wanted stiff behaviour with this design, Rambøll suggests to model it with a material with Young's modulus five times the jacket material's. In addition, the different transition piece beams are chosen to be modelled with one element each (B31 elements), also increasing the stiffness. This also includes the bottom tower piece. The latter piece has regular cross section properties in contrast to the rest of the tapered tower section. The tower piece cross section properties corresponds to the tower model bottom properties. See figure 3.2 for an illustration of the transition piece and table 3.2 and 3.3 for a summary of the model parameters. Note that the *upper tower connection beams* is a modelling idealization for connecting the transition piece beams to the tower beam.



(a) FE model with rendered beam profiles



(b) Beam and level descriptions

Figure 3.2: Transition piece overview

Table 3.2: Summary of transition piece parameters, see table 3.3 for beam section properties.

Parameter	Value
Top level z-coordinate [m]	26.00
Elbow level z-coordinate [m]	22.00
Bottom level z-coordinate [m]	18.00
Young's modulus [GPa]	1050.00
Poisson's ratio [-]	0.3
Mass density [kg/m^3]	7850.00
Rayleigh α coefficient [-]	0.159
Rayleigh β coefficient [-]	0.006

Table 3.3: Summary of the transition piece beam section properties.

Beam type	Profile radius [m]	Profile wall-thickness [m]	Elements per beam [-]
Upper tower connection beams	0.70	0.08	1
Elbow beams	0.70	0.08	1
Jacket leg beams	0.70	0.08	1
Horizontal bracing beams	0.70	0.08	1
Tower piece beam	4.15	0.07	1

3.3 Jacket

The jacket is modelled with beam elements (B31) with geometry following the Reference Jacket design drawings. However, the coordinate system used by Rambøll is rotated 45 degrees, see figure 3.3. Also, the optimized cross section properties for the structure joints are neglected, meaning that the main part properties of the different beams are chosen. The figure in appendix A.3 shows how the design drawings are interpreted and what dimensions that are chosen. For the connection between jacket and piles, a grouted connection with the jacket legs located inside the piles are presented by Rambøll. This lower part of the jacket legs are neglected as the piles are modelled with beam elements as well.

The jacket is a four legged design with four levels of X-bracing. The main parameters are therefore the bottom and top width, and the height of each bracing level. All jacket components are pipe section beams, where legs and bracing associated with the same level have the same cross section properties. The different cross sections also have different number of elements. Table 3.4 and 3.5 summarizes the jacket parameters and figure 3.3 gives an overview of different parameters.

Table 3.4: Summary of jacket parameters, see table 3.5 for beam section properties.

Parameter	Value
Top width [m]	14.00
Bottom width [m]	38.00
Level T z-coordinate [m]	18.00
Level 4 z-coordinate [m]	16.80
Level 3 z-coordinate [m]	5.676
Level 2 z-coordinate [m]	-8.084
Level 1 z-coordinate [m]	-25.124
Level 0 z-coordinate [m]	-46.214
Level B z-coordinate [m]	-47.50
Level Bi z-coordinate [m]	-48.50
Young's modulus [GPa]	210
Poisson's ratio [-]	0.3
Mass density [kg/m^3]	7850
Rayleigh α coefficient [-]	0.159
Rayleigh β coefficient [-]	0.006

Table 3.5: Summary of the jacket beam section properties.

Beam type	Profile radius [m]	Profile wall-thickness [m]	Elements per beam [-]
Leg level T beams	0.70	0.066	2
Leg level 4 beams	0.70	0.042	10
Leg level 3 beams	0.70	0.042	10
Leg level 2 beams	0.70	0.042	10
Leg level 1 beams	0.70	0.07	10
Leg level 0 beams	0.70	0.12	2
Leg level B beams	0.70	0.12	2
Bracing level 4 beams	0.52	0.02	10
Bracing level 3 beams	0.416	0.016	5
Bracing level 2 beams	0.42	0.02	5
Bracing level 1 beams	0.468	0.018	5
H bar beams	0.53	0.03	5

3.4 Piles

The piles are modelled as pipe profiled beam elements (B31) with a given length and five different cross sections. The outer diameter is the same for the whole pile, but the different sections have different wall-thickness. This is an optimization made by Rambøll for a preliminary jacket design, but is here chosen to accompany the final reference jacket as well. The different sections have different length, but are chosen to be modelled with 10 elements each, which is a rather fine mesh for the shortest sections. However, as a pragmatic choice the top section is modelled with 14 elements, to ensure that the piles get a node exactly at the specified mudline (seabed level) for the soil part. This makes the tie constraint used to attach the piles to the soil more realistic. Table 3.6 and 3.7 summarizes the pile parameters and figure 3.4 gives a overview of different parameters.

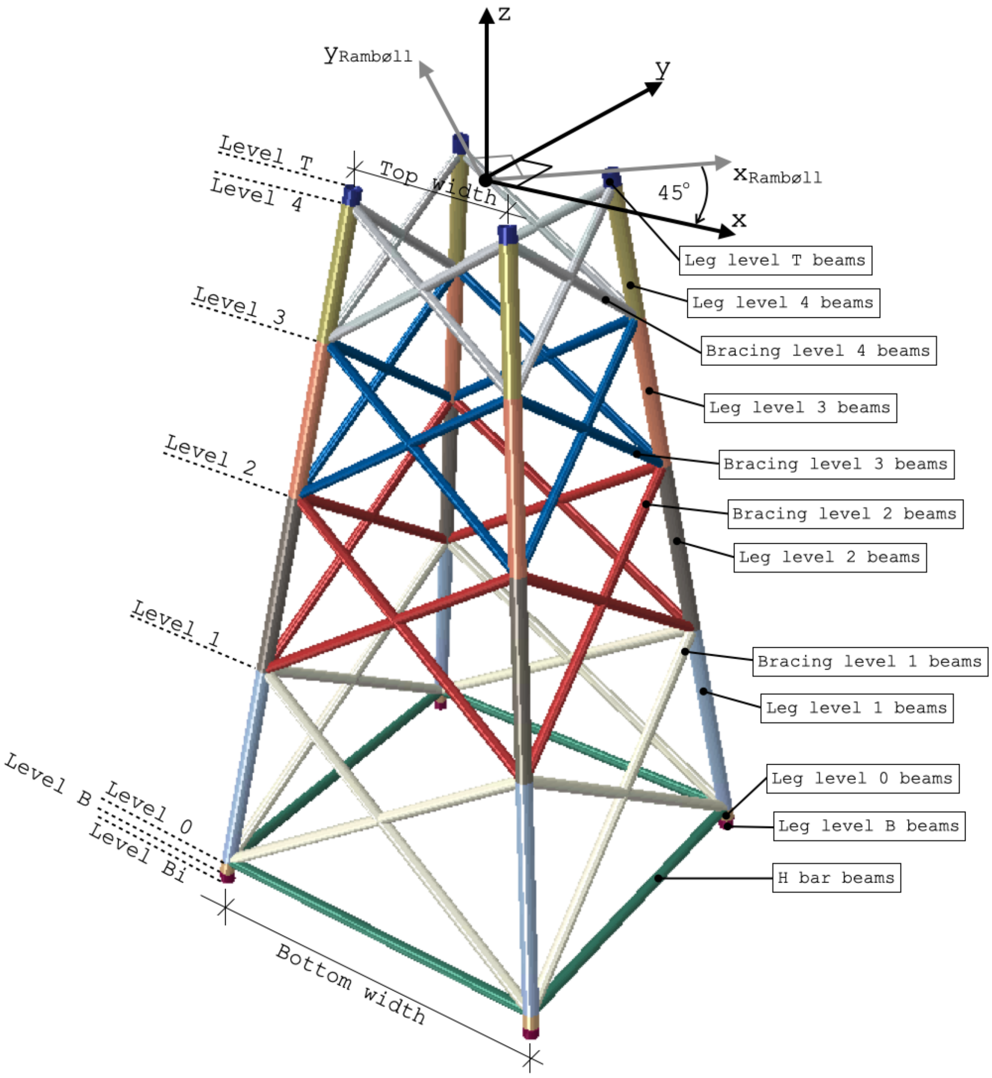


Figure 3.3: Description of jacket parameters on FE model with rendered beam profiles. Same color beams indicating same cross section type. Note that each X-bracing consists of four beams, one beam starts from K-joint and ends at X-joint. Axis origin do not coincide with the modelled z-level. *Rambøll axis* shows design drawings axis-orientation.

Table 3.6: Summary of the pile parameters. See Table 3.7 for beam section properties.

Parameter	Value
Level 5 z-coordinate [m]	-48.50
Level 4 z-coordinate [m]	-59.00
Level 3 z-coordinate [m]	-65.00
Level 2 z-coordinate [m]	-70.00
Level 1 z-coordinate [m]	-86.00
Level 0 z-coordinate [m]	-92.00
Pile outer diameter [m]	2.438
Young's modulus [GPa]	210.00
Poisson's ratio [-]	0.3
Mass density [kg/m^3]	7850.00
Rayleigh α coefficient	0.00
Rayleigh β coefficient	0.00

Table 3.7: Summary of the pile beam section properties.

Beam type	Profile wall-thickness [m]	Elements per section [-]
Level 5 beams	0.052	14
Level 4 beams	0.050	10
Level 3 beams	0.036	10
Level 2 beams	0.028	10
Level 1 beams	0.030	10

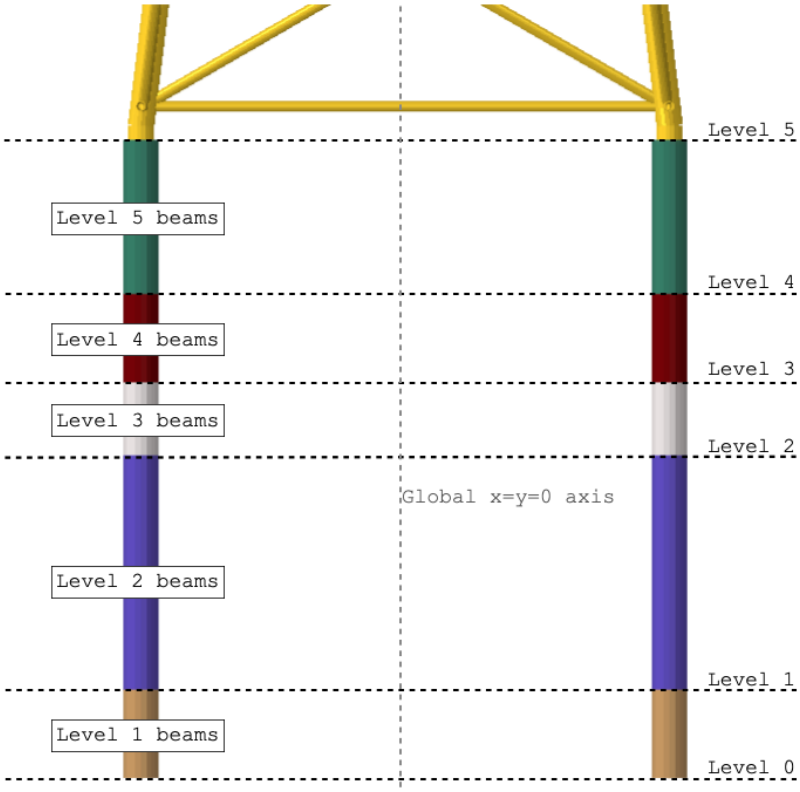


Figure 3.4: Description of pile parameters. Same color beams indicating same cross section type.

3.5 Soil

The soil part is modelled with 8-node linear interpolated brick elements with reduced integration and hourglass control (C3D8R). The part has extruded holes for the piles, with hole diameter corresponding to the pile profile outer diameter. The chosen horizontal mesh consists of a course main mesh, and a finer mesh around the piles as these areas are assumed most important. The horizontal mesh is controlled by three parameters; (1) course element length, (2) fine mesh offset from piles, and (3) number of fine elements per fine mesh offset. The parameters are set to 5m for the course element length, 5m for the fine mesh offset, and 3 elements per fine mesh offset. The latter gives a fine element length of 5/3m. When Abaqus seeds by element size, as for the course mesh, it makes a seeding with approximate the specified length, to make the element size evenly distributed on the specified lengths. The vertical mesh is seeded individually for each layer, where the layer vertical element length, L_v, is set according to

$$L_\mathrm{v} < \frac{v_s}{8f_\mathrm{max}} \tag{3.5.1}$$

where v_s is the layer shear wave velocity and f_max is the highest frequency of interest in the earthquake loading. For this project f_max is assumed 6Hz, see section 2.2.3 for earthquake load properties and section 2.2.5 for the theory behind equation (3.5.1). The seeding procedure is done for each layer by the Python-scripts, assuring a number of elements in the vertical direction of the layer giving element lengths satisfying equation (3.5.1).

The dimensions of the soil part is given by *total width* in x- and y-direction, and by *depth* in z-direction. The specified mudline level indicates the z-coordinate of the top of the soil part. Each soil layer and accompanying material is implemented according to the given soil profile. Table 3.8 summarizes the soil part parameters used and table A.1 shows the given soil profile. When applying Mohr-Coulomb plasticity, cohesion yield strength, friction angle and dilation angle have to be specified. The cohesion (yield strength) shown in table A.1 is calculated as 0.2 times the effective vertical stress based on the reported drained friction angle of the soil (35 degrees). The dilation is assumed zero for simplicity and the friction angle is also set to zero because the material behaves undrained under earthquake loading due to its high strain rate. Zero friction angle actually makes the Mohr-Coulomb criterion reduce to the pressure-independent *Tresca criterion* [31], and zero dilation angle gives no plastic volumetric strain rate. The Tresca criterion is then:

$$F = T/2 - c = 0 \tag{3.5.2}$$

where $T = \sigma_{\max} - \sigma_{\min}$ and c is the cohesion.

Table 3.8: Summary of soil part parameters.

Parameter	Value
Total x-width [m]	200.00
Total y-width [m]	200.00
Depth [m]	92.00
Mudline z-coordinate [m]	-50.00
Fine mesh offset [m]	5.00
Fine mesh horizontal elements per offset [-]	3
Coarse mesh horizontal element length [m]	5.00
Rayleigh α coefficient [-]	0.429
Rayleigh β coefficient [-]	0.003

3.6 Damping

For this model, Rayleigh damping is applied. The Rayleigh coefficients are tuned individually for the soil part and the OWT (not including the piles). The piles are for convince not given any damping. The tuning frequencies for the soil are first and fourth horizontal mode. The OWT coefficients are tuned after first and third for-aft mode when the OWT is clamped. Typical damping ratio for the OWT structure is around 2-3%, but for easier interpretation of the results, both parts are tuned for 5% damping ratio. Table 3.9 summarizes the tuning frequencies, damping ratio and the resulting coefficients.

Table 3.9: Model Rayleigh damping tuning modes and coefficients.

Soil. Damping ratio; $\zeta = 5\%$		
Tuning mode 1 [Hz]	0.876	- Soil first horizontal mode
Tuning mode 2 [Hz]	4.451	- Soil fourth horizontal mode
Alpha coefficient [-]	0.429	
Beta coefficient [-]	0.003	
OWT. Damping ratio; $\zeta = 5\%$		
Tuning mode 1 [Hz]	0.281	- Clamped OWT first for-aft mode
Tuning mode 2 [Hz]	0.255	- Clamped OWT third for-aft mode
Alpha coefficient [-]	0.159	
Beta coefficient [-]	0.006	

3.7 Soil boundaries and verification

For the artificial boundaries, i.e., the boundaries truncating the soil domain, fixed boundary (or in reality pinned as the C3D8R element has no rotational DOFs) is chosen for the bedrock boundary, and tied degree of freedoms (TDOF) boundary is chosen for the lateral boundaries. The TDOF boundary is chosen as it is shown to represent free-field motion exact, and to perform good in SSI analysis with horizontal earthquake excitation [28]. The TDOF implementation is done with multi point constraints (MPC) tie in Abaqus [41], and is illustrated in figure 3.5. The chosen MPC tie system allows for bi-directional horizontal earthquake excitation. None of the boundaries have the possibility of transmitting energy from scattered waves (waves radiating away from the structure, see figure 3.6), thus, the location of the boundaries have to be at a sufficient long distance from the piles, such that scattered waves are sufficiently damped when reaching the boundaries. The bedrock boundary is assumed less important in this model, and is simply enough matching the depth of the given soil profile. The x- and y-width of the soil part is chosen to be 200m each, making the lateral boundaries 83m away from the pile center.

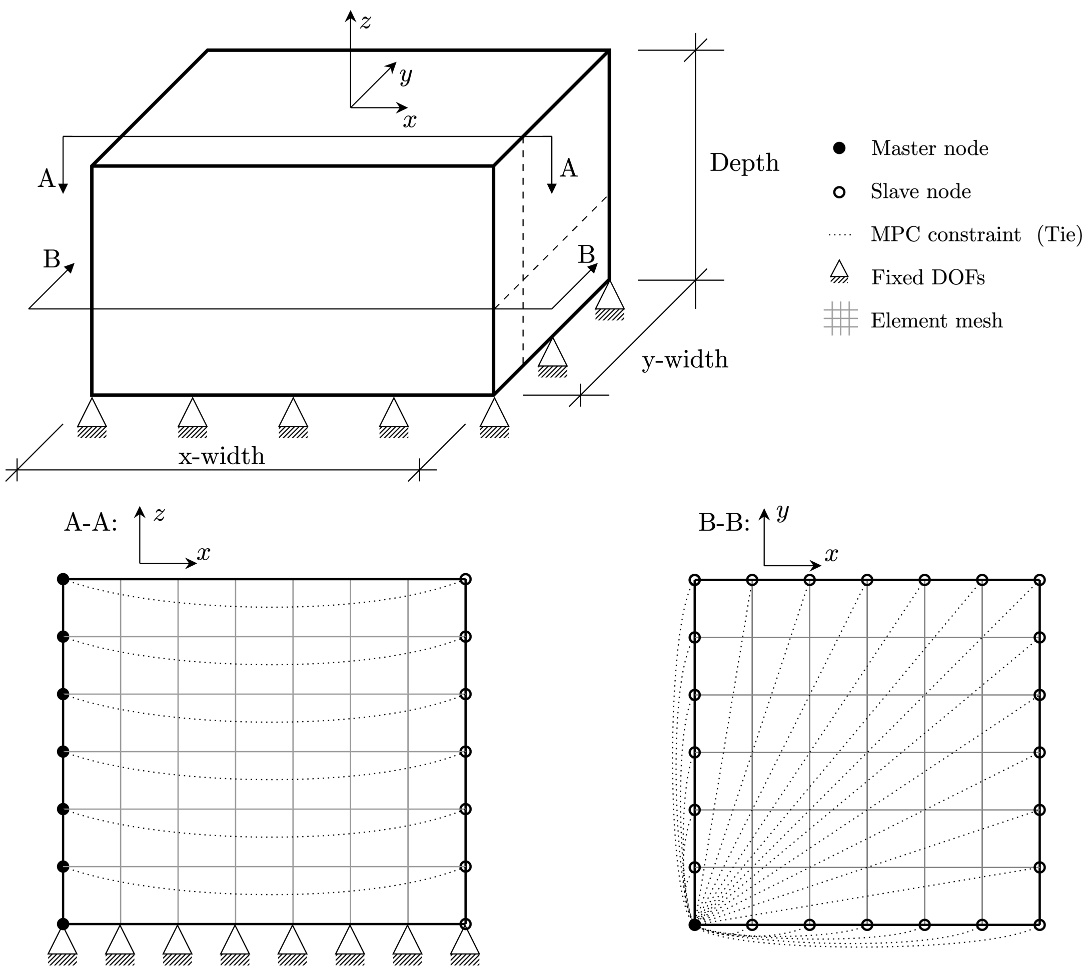


Figure 3.5: Schematic illustration of the chosen artificial boundaries.

The chosen boundaries are verified by; (1) comparing natural frequencies to the analytical solution, (2) comparing measured free field amplification to analytical solution, and (3) comparing the boundary motion (for the integrated model) during earthquake with the free field motion, to verify the location of the boundaries. The verification is presented in the following sections. Note that (1) and (2) was carried out before the final soil profile and boundary location was established, thus, different parameters are used, but the same boundaries and element type yields.

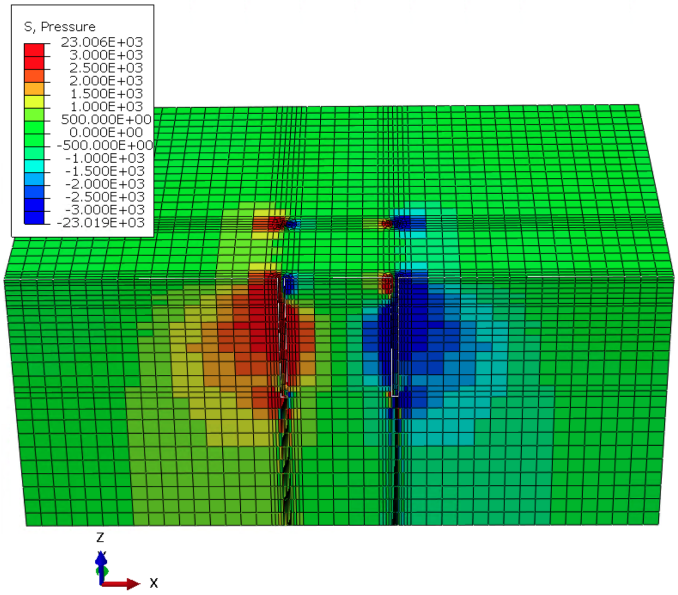


Figure 3.6: Visualization of pressure waves radiating away from the piles during Loma Prieta N-S earthquake excitation. Piles and OWT structure not rendered for visualization purposes. The shown frame is from $t = 6.6$s. The result is obtained from a dynamic implicit analysis without soil non-linearities considered.

3.7.1 Natural frequencies

To obtain the natural frequencies, a *soil slice* is considered. The soil slice is homogeneous and has the dimensions width×height×thickness = 100m × 60m × 1m, see figure 3.7. The chosen TDOF boundaries and an element mesh with characteristic element length 2m and C3D8R elements are applied. Material properties are

$$E = 540 \text{ MPa}$$
$$\nu = 0.3$$
$$\rho = 1700 \text{kg/m}^3$$

which yields a shear wave velocity of

$$v_s = \sqrt{\frac{G}{\rho}} = \sqrt{\frac{E}{2(1+\nu)\rho}} \approx 350 \text{m/s} \tag{3.7.1}$$

If assuming 10Hz is the max frequency of interest;

$$L_\text{v} \leq \frac{v_s}{8 f_\text{max}} = 4.4\text{m} \tag{3.7.2}$$

Thus, the chosen vertical element length should be small enough. The theoretical natural frequencies are given by equation (2.2.13) and are compared to the soil slice frequencies extracted by Abaqus in table 3.10.

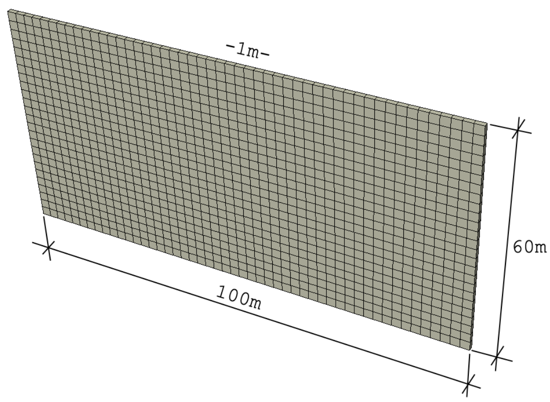


Figure 3.7: Visualization of the soil slice used for natural frequency and free field amplification verification.

Table 3.10: Numerical natural frequencies compared to theoretical.

Mode	Theoretical [Hz]	Numerical [Hz]	Error [%]
First horizontal	1.456	1.456	0.00
Second horizontal	4.369	4.365	−0.09
Third horizontal	7.282	7.262	−0.27
Fourth horizontal	10.195	10.139	−0.55
Fifth horizontal	13.107	12.989	−0.90

Table 3.10 shows that the natural frequencies corresponds well to the exact solution, which assumes an infinite soil domain in the horizontal direction. The slightly lower frequencies obtained by the FE model is expected, as reduced integration gives a softer system. Abaqus also uses a lumped mass representation for the chosen element, which also *in general* yields a lower natural frequency [14]. The observed deviation is therefore not to blame on the boundaries, and the chosen boundary representations are assumed to give a satisfying modal representation of the soil part.

3.7.2 Amplification

The same soil slice as in the previous section is still considered. From the extracted natural frequencies, the soil material is given Rayleigh damping tuned for 5 % damping ratio of the first and fifth horizontal mode. The soil slice is excited at the base by a sinusoidal unit-displacement in different frequencies, and the free-field amplification is measured from the steady-state response amplitude. The results are obtained from an implicit dynamic analysis. The measured amplifications are presented in table 3.11 and compared against the exact solution obtained from equation (2.2.19). Figure 3.8 illustrates the measured results on the theoretical amplification-curve. The results show that the inaccuracy is largest for the highest resonance frequency tested. The rest of the results are quite accurate, but worth noticing is that they essentially give a too low amplification factor, which is non-conservative. Although, except of the highest frequency, the measured behaviour is assumed as a good representation.

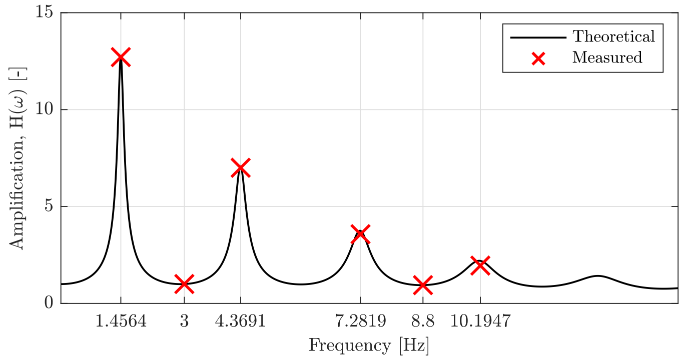


Figure 3.8: Soil slice free field amplification plotted against the theoretical.

Table 3.11: Measured amplification factor compared against theoretical.

Load frequency [Hz]	Theoretical amp. [-]	Measured amp. [-]	Error [%]
1.456	12.733	12.710	-0.18
3.000	0.999	1.001	0.20
4.369	7.072	6.999	-1.03
7.282	3.746	3.572	-4.64
8.800	0.944	0.944	0.00
10.195	2.195	1.951	-11.12

3.7.3 Boundary location

Now considering a soil slice from the actual model, also including the actual soil layer and mesh properties, see figure 3.9. Both the soil slice and the integrated model is excited in x-direction at bedrock by the Loma Prieta N-S accelerations, and response is obtained by a dynamic implicit analysis. The soil slice response is used as a *free field motion reference* for the integrated model. In figure 3.10 the free field reference acceleration is plotted against the mudline acceleration measured at the lateral boundary of the integrated model. The boundary acceleration is measured at a point with the same y-coordinate as two of the piles.

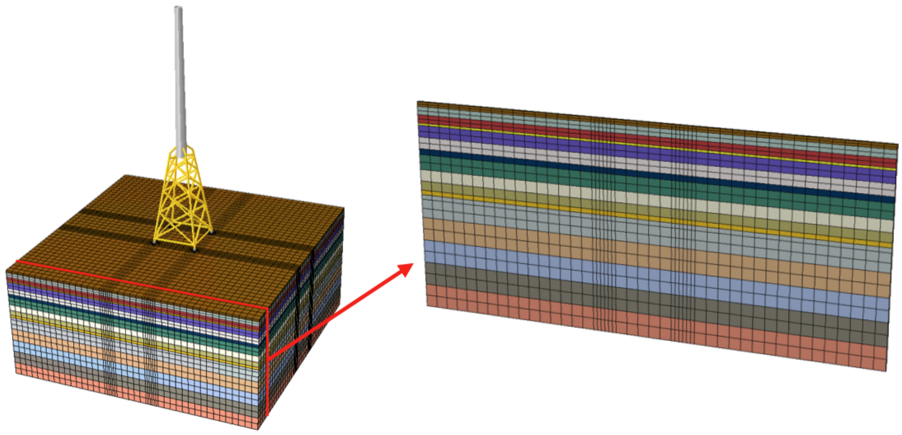


Figure 3.9: Illustration of soil slice from the actual model. Different colors representing different soil layers.

Figure 3.10 shows that the integrated model boundary motion represents the free field reference motion almost exact. An error evaluation method, based on the Euclidean norm (the l^2-norm), is given as

$$R = \frac{\sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}}{\sqrt{\sum_{i=1}^{n} |y_i|^2}} \times 100\% \tag{3.7.3}$$

where y_i is the reference values, and $x_i - y_i$ is the deviation values. As only real numbers are considered, the absolute value signs have no effect, but the interpretation of the method is that the length of the deviation vector is compared against the length of the reference vector. The time series in figure 3.10 yields a value of $R = 1.42\%$. Normally, $R \leq 5\%$ is specified as the range of acceptable error tolerance [28], thus, approving the location of the model boundaries. This means that scattered waves get damped enough before reaching the boundaries, and does not accumulate energy in the system.

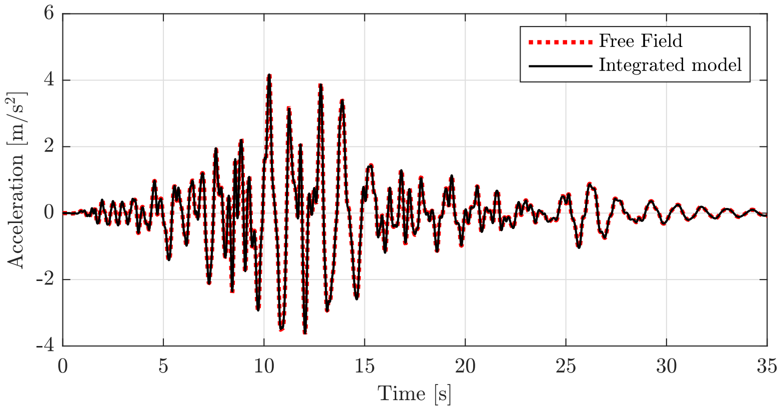


Figure 3.10: Verification of the boundary location.

3.8 Model assembly

Except for the connection between soil and piles, the parts are tied together with an MPC constraint of the intersecting nodes. This practically makes the intersection nodes merge together. For the soil/pile connection, a surface-based tie constraint using *node-to-surface* tie formulation is used [42]. This makes all the nodes on the slave surface have the same motion as the closest point on the master surface. As the piles are represented by beam elements, an offset between the pile and the hole surface exists. Abaqus is specified to not adjust this, but handles it differently based on the choice of master/slave surface. The reason is that only one of the surfaces has rotational degree of freedoms (the pile surface). If the slave surface has rotational DOFs (soil master): The translational motion is constrained, keeping the same offset between the slave node and the initial closest point on the master surface, and a moment based on the constraint force times the offset distance is applied to each slave node. If the master surface has rotational DOFs (pile master): The translational motion is constrained, and a moment is applied to the master DOFs if relevant. The effect of master/slave choice in this soil/pile connection case is illustrated in figure 3.11. The figure illustrates how the hole diameter is not retained when soil is master, and how the soil not follows the rotation of the pile. Pile is therefor chosen as master in this project. Others have made the same master/slave choice as well, e.g., M. Mucciacciaro and S. Sica in their soil/pile interaction article [43].

The soil/pile interaction has also been tried to modelled as a contact problem, but the attempt resulted in far more computational expense, and a better implementation of such a feature remains for further work. The main advantages of modeling it as a contact problem are; (1) frictional behaviour of the soil/pile interaction is represented, and (2) different movements of the pile elements and soil elements (slip) are allowed. Anyways, the soil/pile interaction will in reality be close to rigid and the choice of method should not yield large differences, at least not for the linear analysis.

3.9 Dynamic properties

Running a natural frequency extraction analysis (Abaqus Frequency step) on the integrated model fills the results with internal modes of the soil part. And only the first side-side and for-aft mode is practically identifiable, as they have the lowest frequencies. To investigate the higher

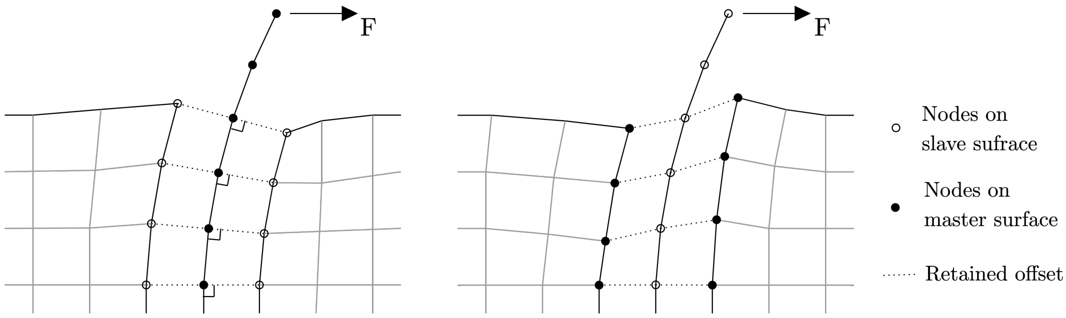


Figure 3.11: Illustrating the effect of using soil or pile as master.

modes, a substructuring of the soil and piles are done. As stated in the theory section; while substructuring in static analysis do not include any further approximations to the linear elastic model, dynamic substructuring does. And in addition, the Abaqus substructuring method is the so-called *Guyan reduction* [44]. Guyan reduction includes only static modes, and not any dynamic modes as *Craig-Bampton reduction*, see section 2.1.7. However, if the modes from the substructure is not of great importance, the results might be a good approximation. The frequencies from the substructuring approach must therefore be interpreted as indicative values. Table 3.12 summarizes these frequencies, and it can be seen that the substructure approach gives the correct natural frequencies for the first modes, indicating little influence from the soil's dynamic modes.

In addition, the tower and RNA clamped at bottom, the OWT (not including piles) clamped at bottom, and the soil part frequencies are presented in table 3.13 to 3.15. The soil part frequencies are extracted from a soil slice representing the actual soil part, see figure 3.9.

Table 3.12: Full system natural frequencies extracted from integrated model and model with soil and piles as substructure.

Integrated model freq. [Hz]	Substructuring freq. [Hz]	Mode description
0.272	0.272	First side-side
0.274	0.274	First for-aft
	1.178	Second side-side
	1.185	First torsional
	1.301	Second for-aft
	2.007	Third side-side
	2.221	Third for-aft
	3.441	Fourth side-side
	3.529	Fourth for-aft
	3.581	First internal jacket
	3.738	First vertical

Table 3.13: Clamped tower and RNA natural frequencies

Frequency [Hz]	Mode description
0.340	First side-side
0.345	First for-aft
1.315	First torsional
1.524	Second side-side
1.850	Second for-aft
4.002	Third side-side
4.188	Third for-aft
6.894	First vertical

Table 3.14: Clamped OWT natural frequencies

Frequency [Hz]	Mode description
0.278	First side-side
0.281	First for-aft
1.198	First torsional
1.259	Second side-side
1.432	Second for-aft
2.341	Third side-side
2.553	Third for-aft
3.791	First jacket internal
4.079	First vertical

Table 3.15: Soil part natural frequencies extracted from soil slice

Frequency [Hz]	Mode description
0.808	First horizontal
1.511	First vertical
1.990	Second horizontal
3.168	Third horizontal
3.722	Second vertical
4.451	Fourth horizontal
5.644	Fifth horizontal

To identify the system total damping, a free decay test, by applying an initial unit displacement at the tower top (for-aft direction), and LSF damping estimation (introduced in section 2.1.3), is conducted. The free decay time series is shown in figure 3.12 and the measured damping is 5.02%. Due to computational time purposes, the free decay test is run on the substructure model. As the substructuring approach have shown to give a correct representation of the first modes, it is assumed to reveal a sufficient good estimate of the damping.

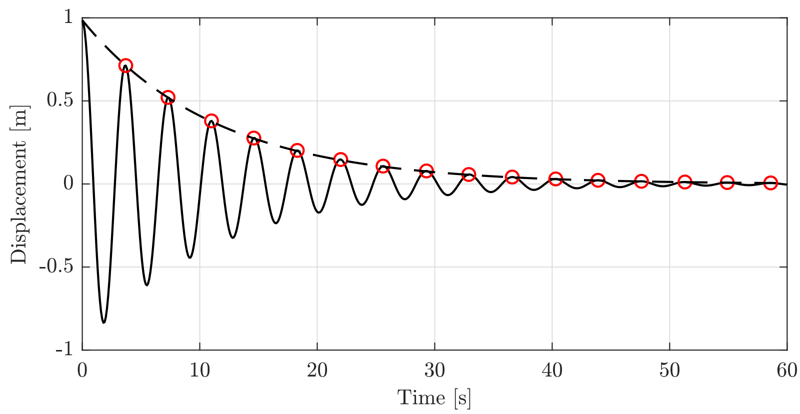


Figure 3.12: Free decay analysis using substructure. Estimated damping ratio: 5.02%

4 OpenFAST

OpenFAST is an open-source, nonlinear, multi-physics tool for simulating coupled dynamic response of wind turbines [22]. It is developed and managed by a team at the National Renewable Energy Laboratory (NREL) through the US Department of Energy. OpenFAST is made with the goal of being community developed and used by research laboratories, academia and industry. NREL's objective is that OpenFAST is a self-sustaining, well tested and well documented software. OpenFAST is written in Fortran 90 and new modules may also be written in C or C++. The OpenFAST community is based on GitHub [11], where the code for compiling your own release is distributed along with a forum for developing and debugging of the code. OpenFAST is the latest version of FAST, and the transition from FASTv8 to OpenFAST represented a transition to an open-source community for better support, developed across research laboratories, industry and academia. There is an old forum on NREL's own website that has more than 15 years of questions and solutions to different errors and problems in earlier versions of the software [45]. This forum is still in use today with new questions and answers every day.

OpenFAST is the framework that couples the calculation from the different computational *modules*. These modules enable nonlinear aero-hydro-servo-elastic simulation in time domain by OpenFAST. NREL has also developed other small programs like *BModes* and *TurbSim* to accompany OpenFAST simulation. The version of OpenFAST used in this project is compiled from the *development branch*. This means that the *main branch* at this time would not be able to run the OpenFAST input files used in this project. This is because the main branch in January 2021 were not able to use soil-structure interaction springs at the reaction nodes of the jacket. A later merge into the development branch allowed the user to apply a time series of forces and moments in all three directions at a given coordinate in the model. This feature allows the model to be excited by an earthquake load further explained in section 4.2.1. This merge, however, had a bug which did not allow the user to run the *SubDyn module* without using the *HydroDyn module* while the time series of forces and moments were applied. NREL fixed the bug and merged pull request number 739 [46] into the *rc-v3.0.0 branch* on the OpenFAST GitHub.

Worth noticing is that an older version of OpenFAST were able to run a module called *SoilDyn*, which included ground motion into the model. This module was branched off and does not work with the current version, thus, the spring approach is needed for the soil-structure interaction.

The modelling approach in OpenFast has to a great extent been a sensitivity study of understanding the nature of the program. Different settings have been implemented and verified against the complementary Abaqus model. The forum communities have been a great guidance, and even some of the developers has been helping by direct communication through e-mailing and video meetings. Our overview of the computational method is even tough still rather modest, and OpenFast has been kind of a black box system during this analysis. The next sections will describe the OpenFAST framework and give a short introduction to each computational modules, the model used in this project with the changes made to input files and the method used to apply the earthquake loading. The last section of the chapter presents the verification results of the model.

4.1 The OpenFAST modelling framework

OpenFAST is the framework that couples the modules used in the calculation. OpenFAST is called the glue-code which glues the modules together. The modules used in this project are ElastoDyn, InflowWind, AeroDyn, ServoDyn, HydroDyn and SubDyn. What each module calculates and how they work are further explained in the next sections. The coupling of these modules as well as the input from BModes and TurbSim are shown in figure 4.1. The modelling in OpenFAST is done with the use of multiple input-files; one for each module and one for OpenFAST to couple the modules. The program starts with the OpenFAST ".fst"-input file where all the other input files are given.

The complete documentation for OpenFAST can be found online at OpenFAST's *readthedocs* page [22]. There are some of the modules that has little information in the documentation, but more

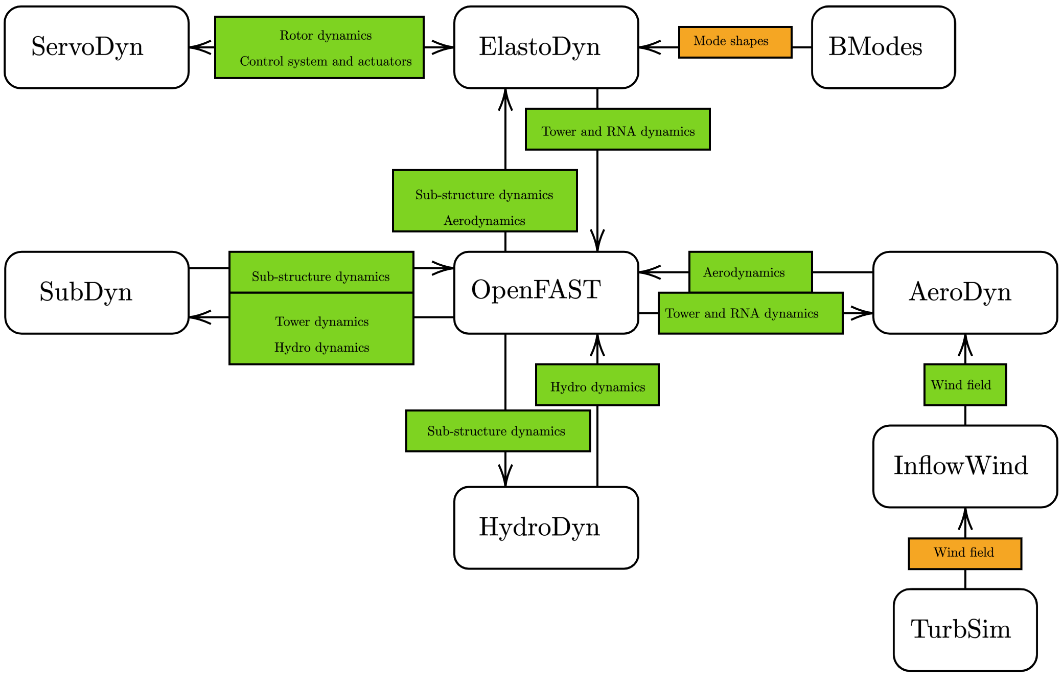


Figure 4.1: Flowchart of the coupling of OpenFAST

information can be found on the forum [45] or on the GitHub [11]. More information can also be found in the development version of the readthedocs page [47]. In the next sections there are given short summaries of all the modules used in this thesis. This is only meant to give a short introduction to what each module does and does not give a complete understanding of the modules.

4.1.1 ElastoDyn

ElastoDyn is the structural dynamics module that models the tower, platform and rotor-nacelle assembly. The primary ElastoDyn input file defines the parameters for the parts of the offshore wind turbine, with regards to degrees of freedom, initial conditions, turbine configuration, mass and inertia, blade file and tower file. The blade and tower files contain the distributed properties along the blade and tower. ElastoDyn requires an input of four tower mode shapes; the two first in each direction, specified as polynomial coefficients. These must be obtained in advance with the use of BModes, see section 4.1.7.

ElastoDyn uses linear Euler-Bernouli beams which implies no axial or torsional DOFs and no shear deformation. The mode shapes from BModes are used as shape functions in the non-linear model using Rayleigh Ritz method.

4.1.2 SubDyn

SubDyn is the structural dynamics module for modelling multi-member, bottom-supported substructures. The module supports jackets, tripods, monopile and other non-floating lattice-type substructures for offshore wind turbines.

The substructure in SubDyn is either clamped or supported by springs at the seabed, and rigidly connected to the transition piece. The spring stiffness at the seabed is provided in its own input file to consider soil structure interaction.

When SubDyn is coupled through the OpenFAST framework, loads and responses are transferred between SubDyn, HydroDyn and ElastoDyn to enable hydro-elastic interaction. The inputs to SubDyn from the other modules at every time step during the simulation are displacements, velocities and accelerations at the interface node from ElastoDyn and hydrodynamic loads from HydroDyn. The outputs from SubDyn to the other modules are reaction loads at the interface node to ElastoDyn and displacements, velocities and accelerations for the substructure to HydroDyn.

SubDyn uses a linear frame finite-element model with either Euler-Bernoulli beam elements or Timoshenko beam elements. In a finite-element analysis of a typical multi member structure the number of degrees of freedom could seriously slow down the dynamic computation. Therefor a Craig-Bampton (C-B) systems reduction is used, as explained in section 2.1.7. C-B reduction may lead to the exclusion of axial modes, which are important to capture the effects from gravity and buoyancy. The static improvement method (SIM) is implemented into SubDyn to mitigate this problem. SIM allows for the model to only use the modes needed to capture the highest frequency relevant to the model. Such as the highest frequency relevant in an earthquake.

4.1.3 HydroDyn

HydroDyn is the hydrodynamics module for calculating the hydrodynamic forces on multi-member substructures. The module supports the same type of substructures as SubDyn. The mapping between HydroDyn and SubDyn means that the nodes entered in the HydroDyn input-file does not have to correspond one to one with the nodes in SubDyn, but it is advised to have some consistency between HydroDyn and SubDyn.

HydroDyn can compute different types of waves; regular waves, irregular waves from JONSWAP/ Pierson-Moskowitz spectrum and irregular waves from white noise spectrum.

HydroDyn does not only calculate waves, but also current, added mass effect and floating platform forces. The added mass effects are applied by the ability to *fill* the members. The added mass effect is equal to the displaced area for circular sections, as explained in section 2.3.2. This is why HydroDyn and SubDyn for now only allow circular sections.

4.1.4 AeroDyn

AeroDyn is a time-domain aerodynamics module that calculates the aerodynamic loading (lift, drag and pitching moments) on both the blades and the tower. AeroDyn uses the wind field processed by InflowWind and generated by TurbSim, see section 4.1.6 and 4.1.8. AeroDyn calculates the aerodynamic loads on both the blades and the tower based on the principle of *actuator lines*. This means that the flow around a 3D object is approximated by a local 2D flow around a cross section. The lift forces, drag forces and pitching moments are used to approximate the distributed pressure and shear stresses along the length of the blade. The total 3D aerodynamic loads are found by integrating the 2D distributed loads along the length of the blade [22]. A further explanation is deemed out of scope for this project but can be found on the website for the documentation of OpenFAST.

4.1.5 ServoDyn

ServoDyn is the control and electrical drive dynamics module of OpenFAST. It includes models to control blade pitch, nacelle yaw, shaft brake, blade-tip brakes and generator torque. One of the pitch control modes in OpenFAST is the bladed-style DLL which is used in this project. The controller used is the NREL's Reference OpenSource Controller (ROSCO) [48].

The ServoDyn module also has the capability of applying a time series of loads and moments at several given coordinates in the model. This capability is called structural control and allows the user to apply an earthquake to the reaction nodes of the model as a time series of loads and moments. The calculation of loads and moments are further explained in section 4.2.1.

4.1.6 InflowWind

InflowWind is the OpenFAST module for processing wind-inflow data generated by TurbSim, as explained in section 4.1.8. InflowWind can process different type of wind fields; uniform, binary TurbSim full-field, binary Bladed-style full-field or HAWC format. It can also internally calculate steady wind field. InflowWind receives the coordinate position of various points from the driver code and returns the undisturbed wind-inflow velocities at these coordinates.

4.1.7 BModes

BModes is a finite-element code that calculates the natural frequency as well as the mode shape for either a blade or a tower [49]. Both the blade and tower may have a tip attachment, which is assumed to be a rigid body with mass, six moments of inertia and a mass centroid that can be offset from blade or tower axis. The elements used by BModes is an element with 15 DOFs. The DOFs are divided into three torsional, four for axial and one for each of the tower bending direction; fore-aft and side-side as shown in figure 4.2.

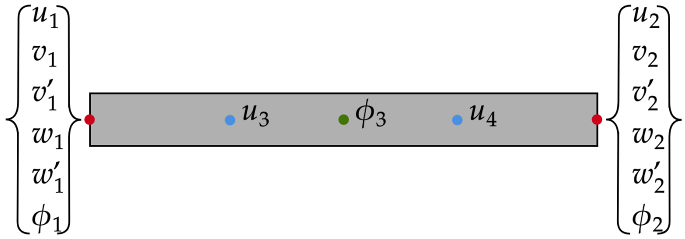


Figure 4.2: 15-DOFs element used in BModes

4.1.8 TurbSim

TurbSim is a turbulent-wind simulator that uses a statistical model to numerically simulate stochastic, full-field, time series of three-component wind-speed vectors. These vectors correspond to wind-speeds at points in a two-dimensional vertical rectangular grid, fixed in space. TurbSim calculates spectra of velocity components and spatial coherence which are defined in the frequency domain and the time series are produced with the use of an inverse Fourier transform [37].

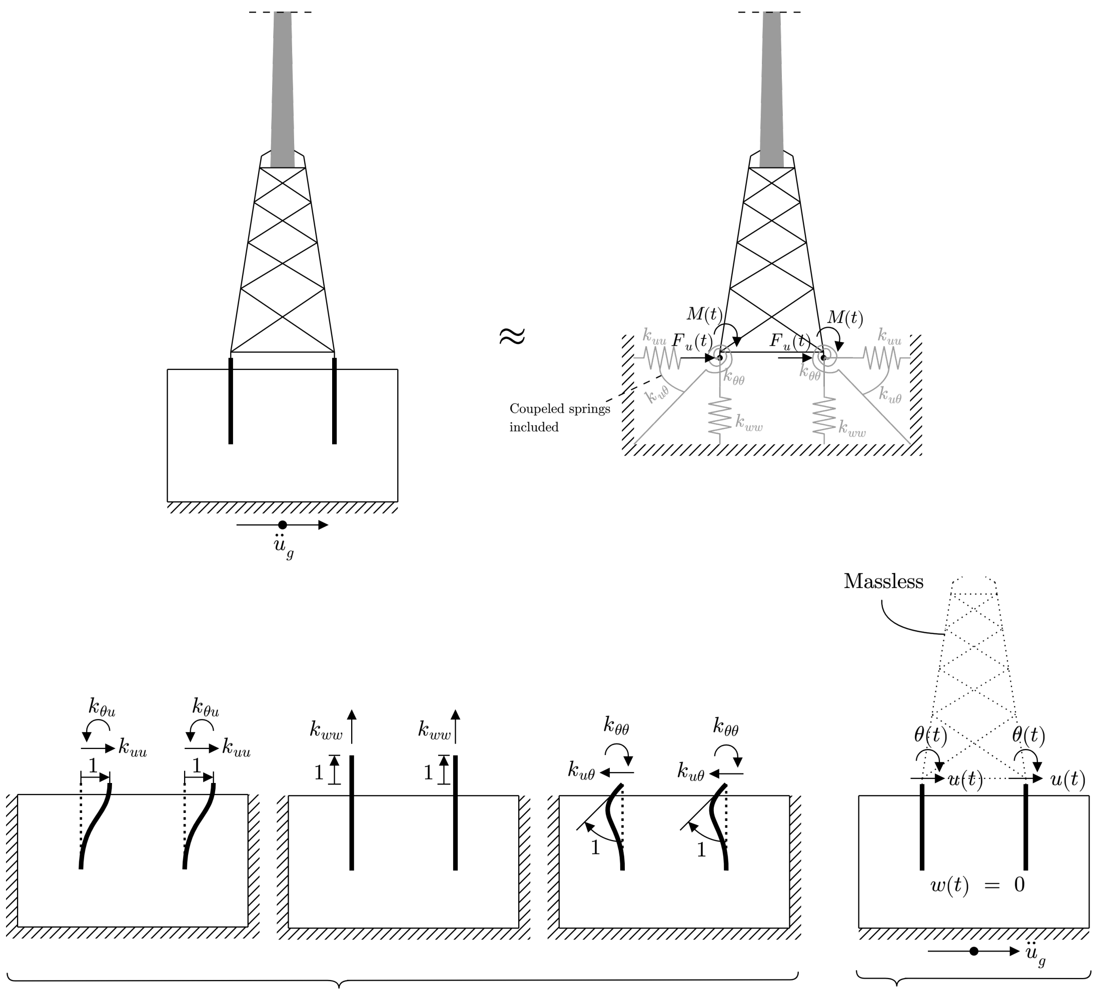
4.2 The reference OWT OpenFAST model

As described in section 1.3, the OpenFAST model used in this project is obtained by downloading complete OpenFAST input files for the IEA 10MW RWT. All the input files are downloaded from the IEA GitHub repository [50]. The files from IEA are made according to the IEA task 37 project report [6], and since the model from IEA models a turbine on a monopile, the SubDyn and HydroDyn files is rewritten to incorporate the reference jacket. Some changes are made to the ElastoDyn files and the AeroDyn files, to accommodate the new substructure. The file for ServoDyn is left untouched except of applying the earthquake load. The changes and verification of the model is further described in the sections below, and the input files used to model the OWT in this project can be found in appendix D. The input files for all the different cases are not attached since the different modules are only turned on or off.

4.2.1 Multi-step method to include soil-structure interaction

As previous stated, the OpenFAST software does not give the opportunity to include a soil foundation. The soil-structure interaction in the OpenFAST model is therefore represented by a multi-step approach. The pile and soil foundation used in this project is then represented by linear elastic springs attached to the bottom of the jacket legs. The effects of radiation damping and added mass from the foundation will then not be included in the dynamics. To apply earthquake load in OpenFAST, the load must be applied as a time series of forces, and not as an acceleration time series, like in Abaqus. The forces are obtained using the same linear springs and a time series of pile top displacements including the kinematic interaction.

Combining the steps of a multi-step method to represent the SSI effects relies upon the principle of superposition, and would not yield if non-linear soil effects were included in the different steps. Thus, even though OpenFAST has the capacity of utilizing a non-linear solver, soil non-linearities cannot be included by this method.



$$[\mathbf{K}] = \begin{bmatrix} k_{uu} & 0 & k_{u\theta} \\ 0 & k_{ww} & 0 \\ k_{\theta u} & 0 & k_{\theta\theta} \end{bmatrix} \qquad \{\mathbf{F}\} = \begin{Bmatrix} F_u(t) \\ F_w(t) \\ M(t) \end{Bmatrix} = \begin{bmatrix} k_{uu} & 0 & k_{u\theta} \\ 0 & k_{ww} & 0 \\ k_{\theta u} & 0 & k_{\theta\theta} \end{bmatrix} \begin{Bmatrix} u(t) \\ w(t) \\ \theta(t) \end{Bmatrix} = [\mathbf{K}]\{\mathbf{u}\}$$

Figure 4.3: The applied multi-step method shown for a 2D case

The multi-step procedure used is illustrated in figure 4.3 and described in the list below, where the Abaqus model is utilized for the first two steps:

- On basis of the principle of virtual displacement, extract the pile top stiffness matrix: Apply a unit displacement/rotation at one DOF, restraining motion at the other DOFs, one at a time. For each iteration, the given reaction forces corresponds to the stiffness terms in one column of the stiffness matrix; the active DOF reaction force gives the diagonal term, and the other give the respective coupled (off-diagonal) terms. The boundaries of the soil profile are fixed for this step. The step is illustrated for a 2D system down left in figure 4.3.
- To obtain the kinematic interaction motion at the top of the piles, the horizontal earthquake

load is applied to the foundation with preferred boundaries, and including a massless jacket to obtain the correct behaviour of the foundation while excited by an earthquake. The earthquake forces are then obtained by multiplying the time series with the already obtained stiffness. This step is illustrated for a 2D system down right in figure 4.3.

- Each jacket leg is *attached* to the ground with the obtained stiffness matrix, and the earthquake load is applied by the force time series at the bottom of each jacket leg. This step is illustrated for a 2D system up right in figure 4.3.

Figure 4.3 shows the procedure for a 2D system, and as the real model is in 3D, with 6 DOFs at each node, the stiffness matrix attached to each jacket leg get the size 6×6. The obtained stiffness matrix, $[\mathbf{K}_{\mathrm{SSI}}]$ used in this project is:

$$[\mathbf{K}_{\mathrm{SSI}}] = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} & k_{x\theta_x} & k_{x\theta_y} & k_{x\theta_z} \\ k_{yx} & k_{yy} & k_{yz} & k_{y\theta_x} & k_{y\theta_y} & k_{y\theta_z} \\ k_{zx} & k_{zy} & k_{zz} & k_{z\theta_x} & k_{z\theta_y} & k_{z\theta_z} \\ k_{\theta_x x} & k_{\theta_x y} & k_{\theta_x z} & k_{\theta_x\theta_x} & k_{\theta_x\theta_y} & k_{\theta_x\theta_z} \\ k_{\theta_y x} & k_{\theta_y y} & k_{\theta_y z} & k_{\theta_y\theta_x} & k_{\theta_y\theta_y} & k_{\theta_y\theta_z} \\ k_{\theta_z x} & k_{\theta_z y} & k_{\theta_z z} & k_{\theta_z\theta_x} & k_{\theta_z\theta_y} & k_{\theta_z\theta_z} \end{bmatrix} \tag{4.2.1}$$

$$= \begin{bmatrix} 4.69 & 0.0 & 0.0 & 0.0 & -19.35 & 0.0 \\ 0.0 & 4.69 & 0.0 & 19.35 & 0.0 & 0.0 \\ 0.0 & 0.0 & 24.45 & 0.0 & 0.0 & 0.0 \\ 0.0 & 19.35 & 0.0 & 152.45 & 0.0 & 0.0 \\ -19.35 & 0.0 & 0.0 & 0.0 & 152.45 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 39.68 \end{bmatrix} \cdot 10^8$$

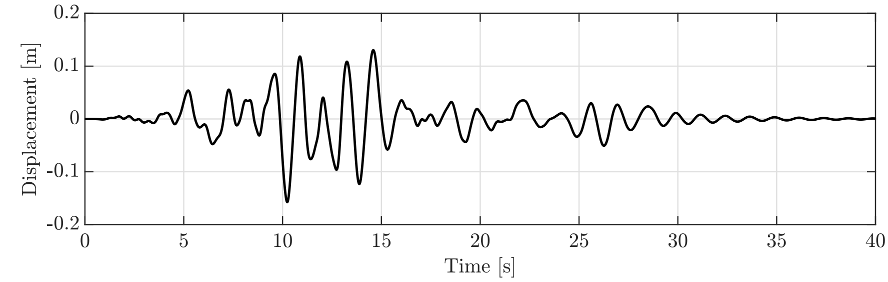
If considering only horizontal earthquake loading in the x-direction; applying N-S acceleration of the Loma Prieta earthquake (see section 2.2.3) gives kinematic interacted pile top displacements in x-direction and rotations about the y-direction:

$$\{\mathbf{F}\} = \begin{Bmatrix} F_x(t) \\ F_y(t) \\ F_z(t) \\ M_x(t) \\ M_y(t) \\ M_z(t) \end{Bmatrix} = \begin{bmatrix} k_{xx} & 0 & 0 & 0 & k_{x\theta_y} & 0 \\ 0 & k_{yy} & 0 & k_{y\theta_x} & 0 & 0 \\ 0 & 0 & k_{zz} & 0 & 0 & 0 \\ 0 & k_{\theta_x y} & 0 & k_{\theta_x\theta_x} & 0 & 0 \\ k_{\theta_y x} & 0 & 0 & 0 & k_{\theta_y\theta_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & k_{\theta_z\theta_z} \end{bmatrix} \begin{Bmatrix} u_x(t) \\ u_y(t) \\ u_z(t) \\ \theta_x(t) \\ \theta_y(t) \\ \theta_z(t) \end{Bmatrix} = [\mathbf{K}_{SSI}]\{\mathbf{u}\} \tag{4.2.2}$$

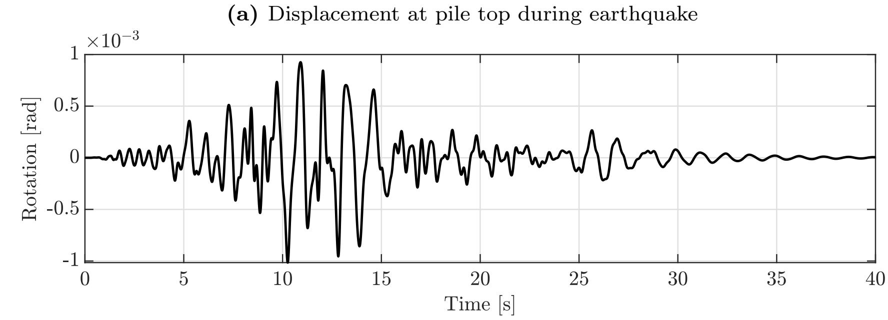
$$u_y(t) = u_z(t) = \theta_x(t) = \theta_z(t) = 0$$

$$\Rightarrow F_x(t) = k_{xx} \cdot u(t) + k_{x\theta_y} \cdot \theta_y(t), \quad M_y(t) = k_{\theta_y x} \cdot u(t) + k_{\theta_y\theta_y} \cdot \theta_y(t) \tag{4.2.3}$$

The displacement and rotation time series used are shown in figure 4.4. Figure 4.5 shows the Power Spectral density (PSD) of the displacement and rotation time series. The force and moment calculated by equation (4.2.3) are shown in figure 4.6.

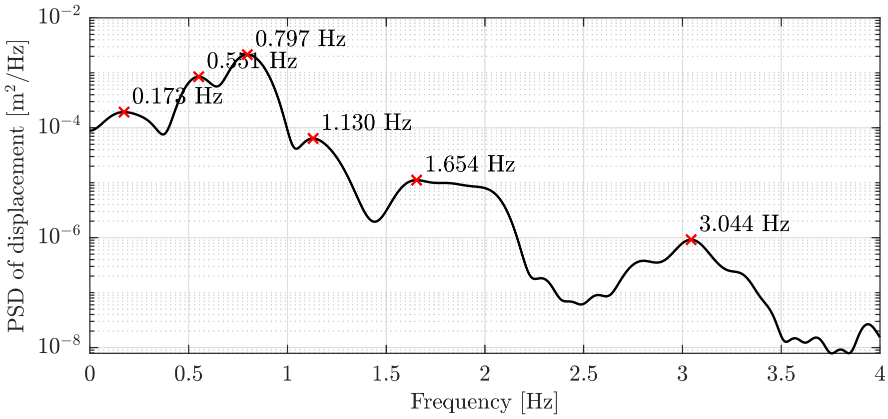


(a) Displacement at pile top during earthquake

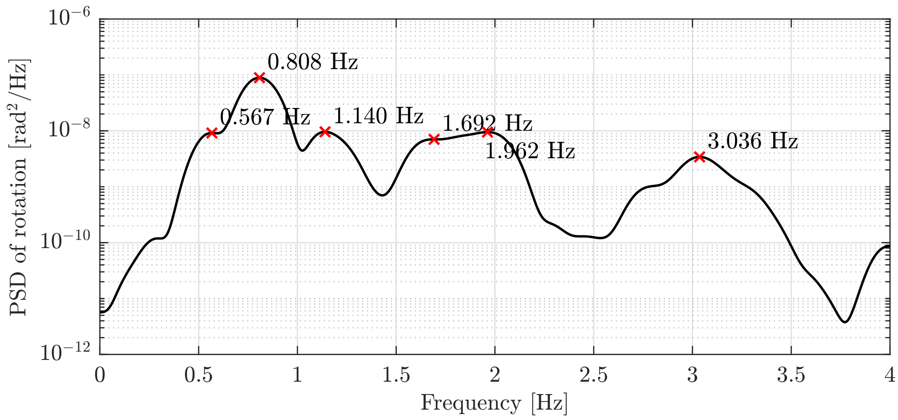


(b) Rotation of pile top during earthquake

Figure 4.4: Displacement and rotation at pile top during a 30 seconds earthquake with free decay



(a) Power spectral density of the displacement



(b) Power spectral density of the rotation

Figure 4.5: Power spectral density of displacement and rotation for the applied earthquake

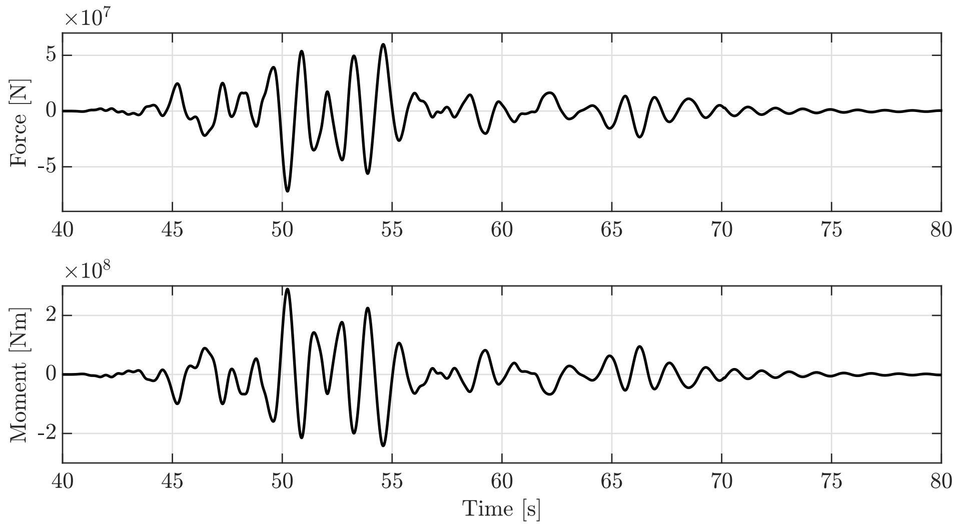


Figure 4.6: Force and moment time series applied at the reaction nodes. Peak force: 71951kN and peak moment: 289487kNm. Earthquake starts at time=40s

4.2.2 Environmental loading

The wind loads in OpenFAST are calculated with the module AeroDyn. This module uses a binary file as an input for the full field wind data which is made by the NREL written program TurbSim. The full field wind data is made with the use of turbulence spectral models and the model used in this project is the *Kaimal spectral model*, which is further explained in section 2.3.3. The wind field made by TurbSim is set to have a mean wind speed of 11m/s at the reference height. This wind speed was chosen since it is the rated wind speed for the turbine used [6]. The reference height is set at the hub at 131.63m above mean sea level. Figure 4.7 shows the wind speed in all three directions, where U-, V- and W-component is parallel to respectively the global x-, y- and z-axis. The wind direction is set to be along the x-axis such that the motion mainly is in the direction of the earthquake. Figure 4.8 shows the PSD of the U-component, and table 4.1 shows the mean wind velocities. The aerodynamic loads on the blades and the tower are calculated by AeroDyn as explained in section 4.1.4.

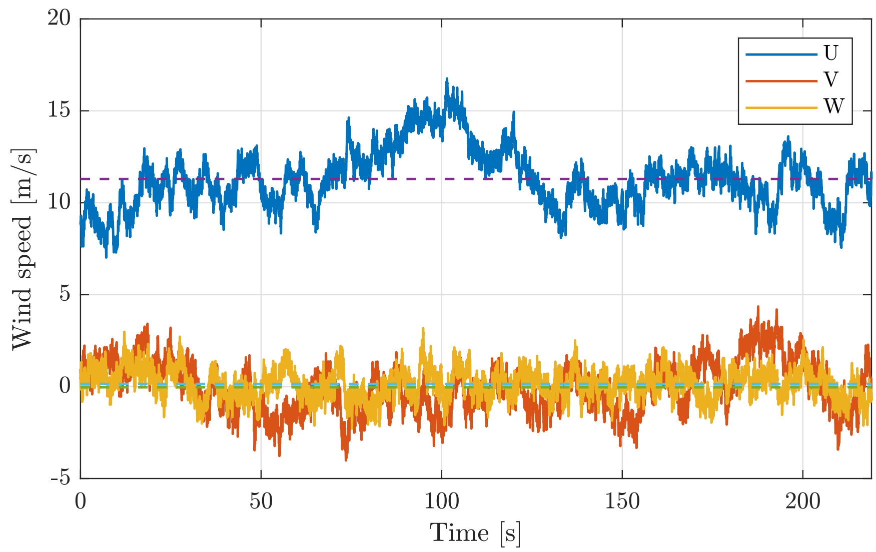


Figure 4.7: Wind speeds at the rotor hub in all three direction. The dashed lines indicates the mean wind speed for the whole time series in each directions.

Table 4.1: Mean wind velocity at hub at 131.63m above mean sea level

Direction	Mean wind speed
U-component	11.2964m/s
V-component	-0.0357m/s
W-component	0.1455m/s

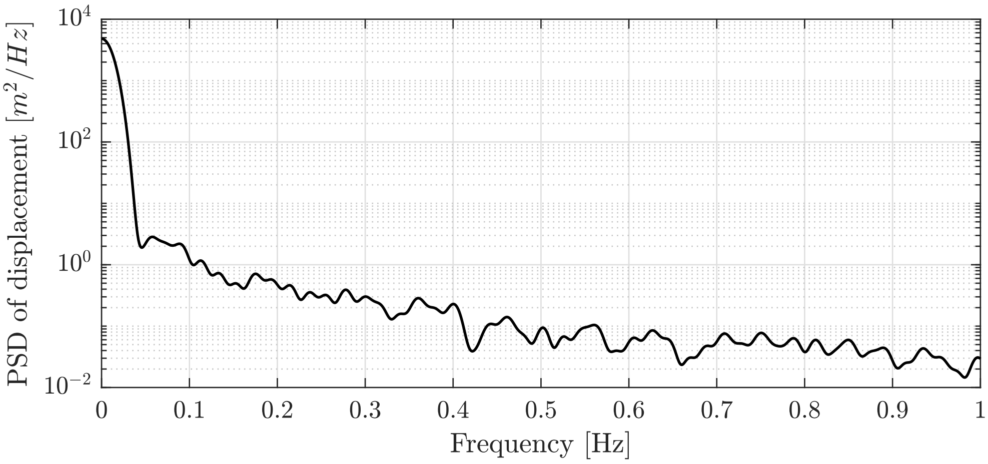


Figure 4.8: Wind spectrum for the wind shown in figure 4.7 in the U-direction which corresponds to the global x-direction of the OpenFAST model

The wave loads in OpenFAST are given through the module HydroDyn where the wave kinematics model is chosen and defined. For the irregular waves, the JONSWAP spectrum are used since the sea state is considered developing when there are high wind speeds blowing on the wind turbine. The JONSWAP spectrum is defined as stated in section 2.3.1 and equation (2.3.6), and the spectrum used is shown in figure 4.9. The value for the peak shape parameter is calculated according to the recommended practice DNV-RP-C205 [32]:

$$\gamma = \exp\left(5.75 - 1.15\frac{T_p}{\sqrt{H_s}}\right) \qquad \text{for} \qquad 3.6 < \frac{T_p}{\sqrt{H_s}} < 5 \tag{4.2.4}$$

Where H_s is the significant wave height and T_p is the peak spectral period. Since design of a wind turbine is not the scope of this project; $H_s = 8$m and $T_p = 12$s are chosen. Based on these values the peak shape parameter γ is:

$$\frac{T_p}{\sqrt{H_s}} = \frac{12}{\sqrt{8}} = 4.243 \qquad \Longrightarrow \qquad \gamma = \exp\left(5.75 - 1.15 \cdot \frac{12}{\sqrt{8}}\right) = 2.3892 \tag{4.2.5}$$

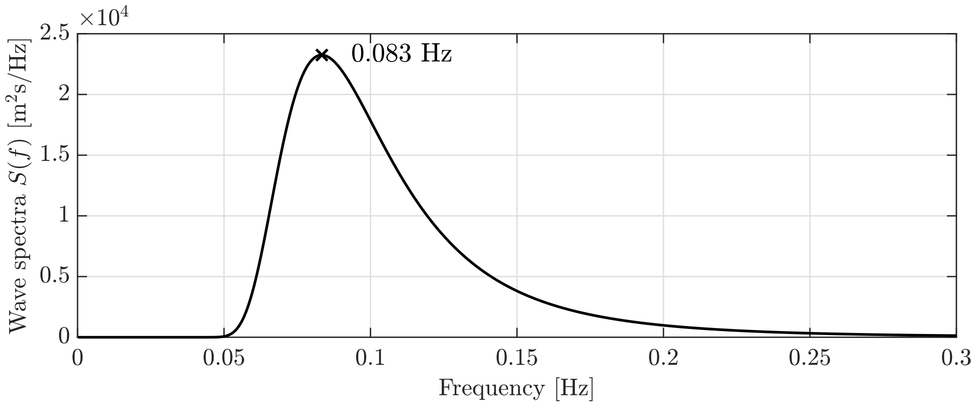


Figure 4.9: Shape of the JONSWAP spectrum used to model the waves in OpenFAST

4.2.3 ElastoDyn configuration

The changes made to the primary input file for ElastoDyn is given in table 4.2. The platform yaw inertia is set equal to the total tower rotational inertia about its center line, due to avoid a potential division by zero. The value used is extracted from the Abaqus model. The change of substructure also raises the need for new mode shapes in the ElastoDyn file. The mode shapes are found with BModes and includes the first and second fore-aft and side-side modes in form of a sextic polynomial obtained with the Excel spreadsheet "ModeShapePolyFitting.xls" provided by NREAL [51]. The input into BModes includes the substructure mass and stiffness matrices, referred to as *hydro_M* and *hydro_K*, tip mass corresponding to the RNA total mass, and the RNA mass moment of inertias. The latter values corresponds to those used in the Abaqus model, see table 3.1, and the matrices is obtained from the SubDyn module.

Parameter	New value
TowerHt [m]	131.63
TowerBsHt [m]	26.0
PtfmCMzt [m]	26.0
PtfmRefzt [m]	26.0
PtfmYIner [$\mathrm{kg\,m^2}$]	40 513 389

Table 4.2: Changes made to the primary ElastoDyn input file

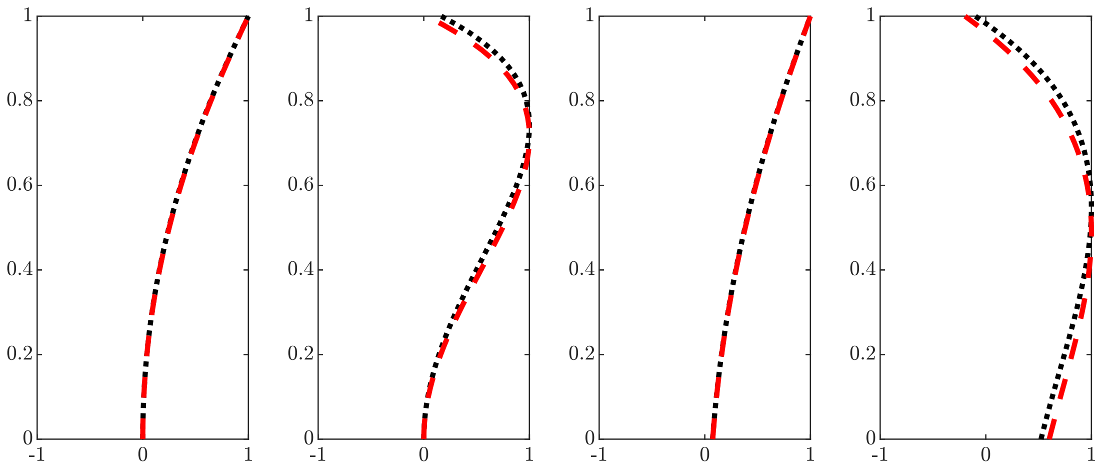
The mode shape calculation procedure is described below:

- Write the SubDyn input file with pile stiffness extracted from Abaqus.
- Run the SubDyn module with the summary switch turned on to acquire the stiffness- and mass-matrices of the jacket.
- Copy these into the BModes input file as the hydro_M and hydro_K. This is done to allow BModes to take into account the mass and stiffness of the substructure situated on piles and soil.
- Run BModes to acquire the correct tower mode shapes with the settings for tower base connection switch set to 2.
- Copy these deflection outputs one by one into the Excel spreadsheet "ModeShapePolyFitting.xls".
- Copy the polynomial coefficients from the Excel spreadsheet of each mode shape into the ElastoDyn tower file (`10MW_ElastoDyn_Tower.dat`).

The modes calculated for this project is shown in figure 4.10. To emphasis the effect of the substructure, figure (a) and (b) shows the corresponding mode shapes if the tower is assumed clamped at the bottom. The natural frequencies for the two systems are presented in table 4.3.

Table 4.3: Natural frequencies calculated with BModes for released and fixed tower end

Mode	Frequency [Hz]
First released side-side	0.2780
First released fore-aft	0.2805
First clamped side-side	0.3516
First clamped fore-aft	0.3569
Second released side-side	1.1625
Second released fore-aft	1.2887
Second clamped side-side	1.5086
Second clamped fore-aft	1.8365



(a) First clamped mode **(b)** Second clamped mode **(c)** First released mode **(d)** Second released mode

Figure 4.10: Clamped and released tower mode shapes in side-side and fore-aft direction. Dashed red line indicates fore-aft and dotted black line indicates side-side.

4.2.4 SubDyn and HydroDyn configuration

The SubDyn and HydroDyn files is rewritten since there is no jacket design available. The coordinates for the nodes and cross-sectional properties are taken from the Abaqus model. This ensures that the geometry is the same for the two models, which is important for verifying of the OpenFAST model establishing. The reaction nodes in both of the modules are set at the top of each pile. The piles is actually sticking 1.5m up above the mudline, but the water depth is set to -48.5m instead of -50m as in the Abaqus model. This is due to the choice of modelling the piles as a part of the soil instead of a part of the substructure. The HydroDyn parameters used for the wave generation is given in table 4.4.

Table 4.4: Parameters used in HydroDyn input file

Parameter	Value
H_s	8m
T_p	12s
γ	2.3892

4.2.5 ServoDyn configuration

To apply the earthquake load to the structure, the ServoDyn input file is changed to allow for the structural control to apply a time series of loads and moments. The switch for number of substructure structural control is set to *four* since there are four reaction nodes in our model. The list of names of the files for substructure structural controllers were entered with the name of the files containing the time series for loads and moments.

4.2.6 AeroDyn configurations

The AeroDyn input file is changed to accommodate the new total height of the structure and the tower properties.

4.2.7 Damping configurations

The damping of the total structure is calibrated to 5% in OpenFAST since it is the chosen damping ratio. There are three places to change the damping of the structure in OpenFAST; (1) Rayleigh damping of the Guyan modes in SubDyn, (2) damping of the retained CB-modes in SubDyn, and (3) damping of the mode shapes in ElastoDyn. The Rayleigh damping is set equal to the Rayleigh damping of the Abaqus model, the retained CB-modes damping is set to 5% and the mode shape damping is set to 35.5%. This yields a measured total damping of the first mode of 5.01% in the fore-aft direction and 5.03% in the side-side direction. The high mode shape damping in ElastoDyn is to enforce the same total system damping as in the Abaqus model. The Abaqus model also applies the Rayleigh Damping to the RNA mass, while no damping of the RNA is included in OpenFAST. Thus, the ElastoDyn damping is used to enforce the same total structural damping. The damping is found by executing a free decay test and using the method of least square fitting further explained in section 2.1.3.

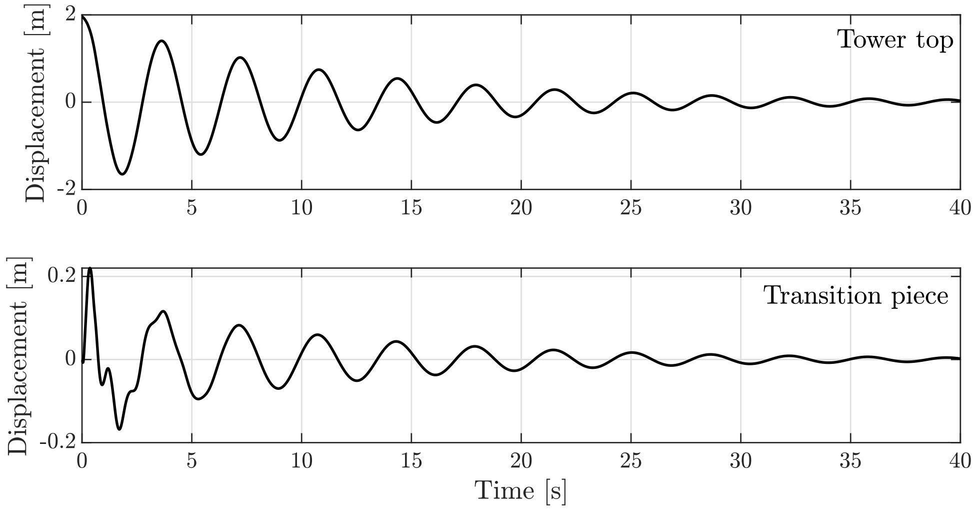
4.2.8 Initial conditions

Since OpenFAST does not calculate a static steady state due to gravity prior to the simulation, there is performed a free-decay analysis with no other loads than gravity. After 200 seconds, it shows that the displacements stabilize at; -0.225m for the top of tower in fore-aft direction, -0.006m at the transition piece in fore-aft direction and -0.0156m at the transition piece in the vertical direction. The negative values in the fore-aft direction tells that the turbine tower bends forewords due to the heavy RNA and the negative value in the vertical direction means that the structure becomes compressed. In analysis where gravity is turned on, these displacements are used as initial conditions.

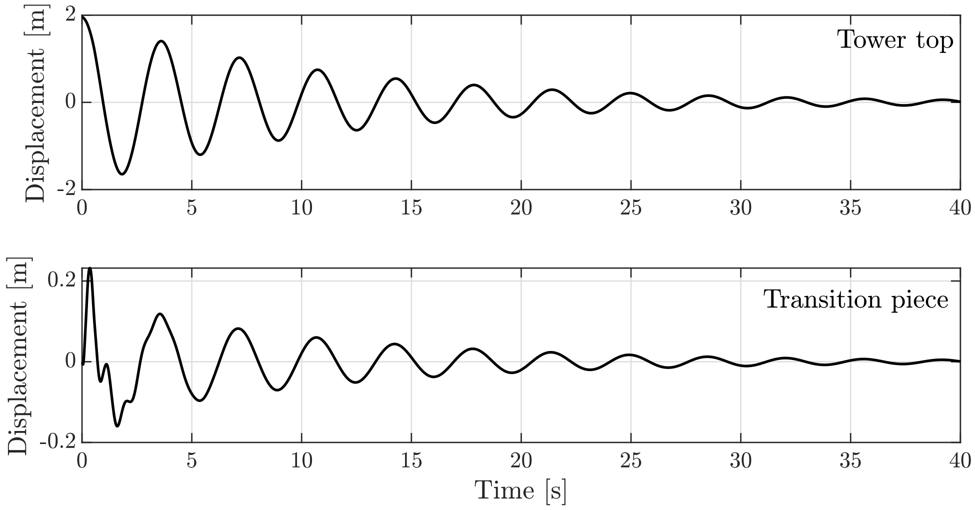
4.3 Model verification

4.3.1 Natural frequencies

The mode shapes calculated with BModes when the bottom of the tower is released with the mass and stiffness from the substructure should be like the natural frequencies of the total structure modelled in OpenFAST and to the Abaqus model. This is controlled by performing two free decay analysis where the top of tower is given an initial displacement of 2m in fore-aft and side-side by turns. During the free decay analysis, the blade bending DOFs are turned off. This is done to ensure that the model behaves in the same manner as in Abaqus and BModes. Turning off all the bending DOFs in the blades means that the RNA will behave like a rigid body, and therefore no disturbance from the blades during the free decay analysis. The gravity is also set to 0m/s^2 because the centre of mass of the RNA has an offset from the tower axis which will give a displacement and make it difficult to compare it to BModes and Abaqus. Figure 4.11 shows the free decay displacement of the tower top and transition piece when the top of tower is given an initial displacement in respectively side-side and fore-aft direction. A PSD analysis of the response time series reveals the frequencies of the first modes shown in figure 4.12. Figure 4.12a and figure 4.12b shows that the first natural frequency in side-side and fore-aft direction are 0.280Hz and 0.281Hz, respectively. The figures also shows that the second side-side and fore-aft natural frequency are respectively 1.174Hz and 1.256Hz. The second natural frequency is only present in the PSD of displacement for the transition piece and not at the top of tower. This is due to that throughout vibration in the second mode, the top of tower does not have any displacement, as shown in figure 4.10d. Table 4.5 shows the results as well as the results from BModes and the Abaqus model.

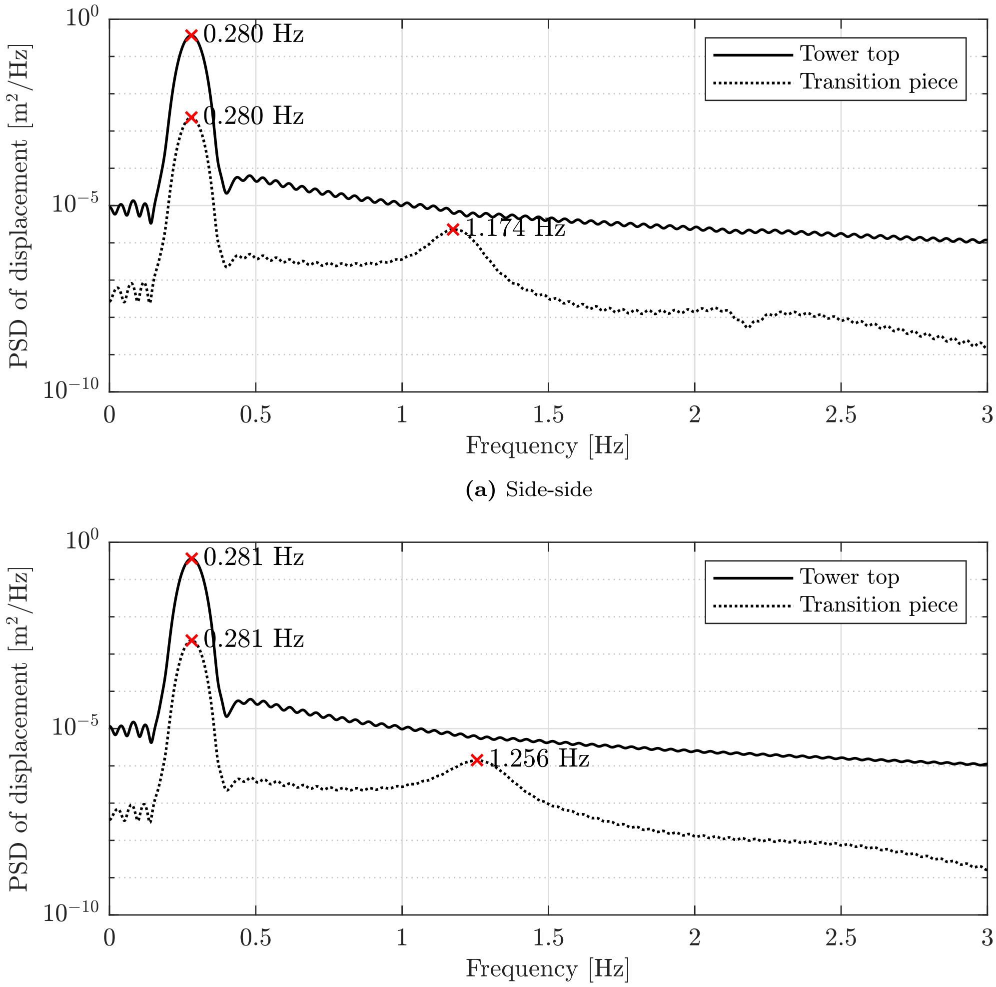


(a) Side-side



(b) Fore-aft

Figure 4.11: Tower top and transition piece displacements in respectively side-side and fore-aft direction for stiff blades



(a) Side-side

(b) Fore-aft

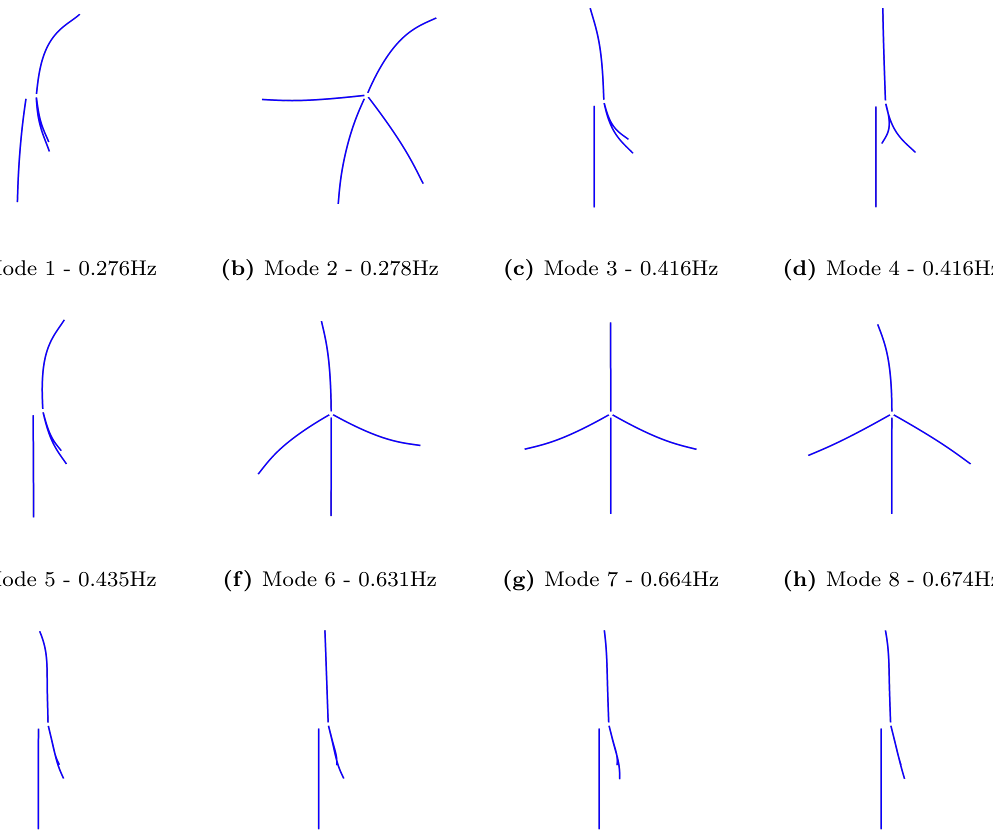
Figure 4.12: PSD of displacement in side-side and fore-aft direction for transition piece and tower top with stiff blades

Table 4.5: Natural frequencies calculated with BModes and natural frequencies measured with free decay analysis

Mode	BModes freq. [Hz]	OpenFAST freq. [Hz]	Abaqus freq. [Hz]
First side-side	0.278	0.280	0.272
First fore-aft	0.281	0.281	0.274
Second side-side	1.162	1.174	1.178
Second fore-aft	1.289	1.256	1.301

The results presented in table 4.5 shows a good match between the BModes and OpenFAST, indicating a correct implementation of the mode shapes. The results also shows a good match with the Abaqus model, indicating a correct modelled system. The differences; however, may be from numerical errors, and the fact that the different models have different discretizations of the system. The applied free decay test gives a clear identification of the second mode shape because the initial conditions does not replicate the first mode shape perfectly. The tower is given an initial displacement of 2m and the jacket is undisturbed at start, while figure 4.10 shows that the first mode also includes a jacket displacement. The initial response of the jacket, clearly visualized in figure 4.11, then gives raise to the identification of the second mode. Figure 4.10 also explains why the second mode is not able to be identified by the tower top PSD; the tower top has barely any displacement during vibration in the second mode.

The natural frequencies of the total structure with the effects from the blades (soft blades) included is extracted by utilizing the *linearization* capability of OpenFAST. The linearization analysis is performed by following the step-by-step description at one of the OpenFAST readthedocs pages [52]. Figure 4.13 shows the first 12 tower and blade mode shapes found. The rest of the modes, as well as mode 10, are only a combination of different Craig-Bampton modes, and the SubDyn module is not capable of visualizing these modes. The linearization analysis of the model does not yield any pure 2nd tower bending mode, of unknown reasons. Most likely is this due to errors while performing the linearization, as the model shows adequate behaviour outside of this analysis.



(a) Mode 1 - 0.276Hz **(b)** Mode 2 - 0.278Hz **(c)** Mode 3 - 0.416Hz **(d)** Mode 4 - 0.416Hz

(e) Mode 5 - 0.435Hz **(f)** Mode 6 - 0.631Hz **(g)** Mode 7 - 0.664Hz **(h)** Mode 8 - 0.674Hz

(i) Mode 9 - 1.349Hz **(j)** Mode 11 - 1.504Hz **(k)** Mode 12 - 1.546Hz **(l)** Mode 13 - 1.585Hz

Figure 4.13: The 12 first mode shapes and it's natural frequencies

4.3.2 Earthquake response

The earthquake response of the OpenFAST model is verified against the Abaqus model to ensure that OpenFAST handles the SSI-effects as wanted. During these calculations, earthquake is the only load applied to the models. Figure 4.14 presents the comparison between the Abaqus model and the OpenFAST model with both stiff and soft blade behaviour. The stiff blades are intended to represent the rigid body assumption used in Abaqus, where the RNA is interpreted as a rigid body and represented by a point mass and mass moment of inertia. Although, the results shows best match with the soft blade behaviour. The two blade behaviours differs especially at the peak around 10 seconds and the first peak after 15 seconds. This is due to energy taken up by the soft blades damps out some of the tower top response. The effect may be somewhat similar to the radiation damping effect of the soil in the Abaqus model.

From the PSD of the response for the three models shown in figure 4.15, the Abaqus model has a peak at the second tower fore-aft mode (1.179 Hz, see table 4.5), while the OpenFAST simulations does not contain this frequency at the top of tower. This is just like for the free decay analysis shown earlier. The reason for the identification of the second natural frequency in the Abaqus tower top response, is that the mode shape has a significant displacement at the top of tower, see figure 4.16, in contrast to the OpenFAST mode. The mode shapes from Abaqus could have been implemented to get a greater correlation with Abaqus. The soft blade response has a peak corresponding to the second mode of the blades (1.616Hz), which is of course not present in the stiff blade response. Although, the response from the OpenFAST model is very similar in both cases.

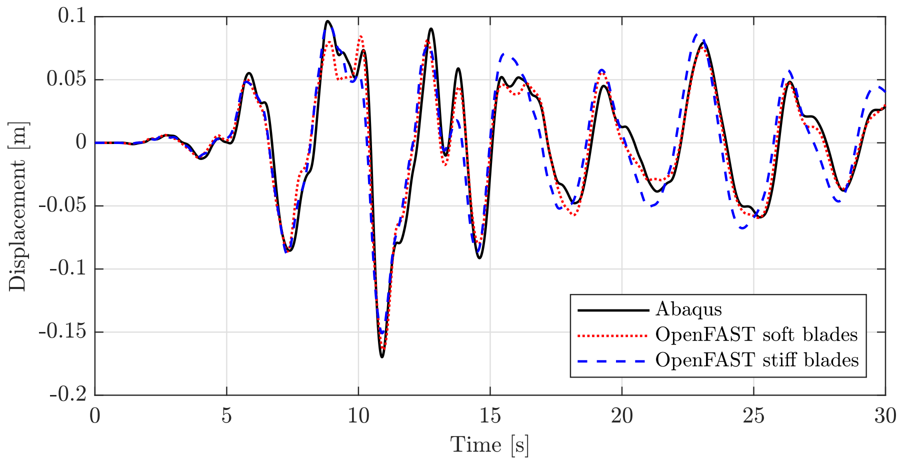


Figure 4.14: Earthquake response of Abaqus full model compared to OpenFAST model with both soft and stiff blades measured at the tower top

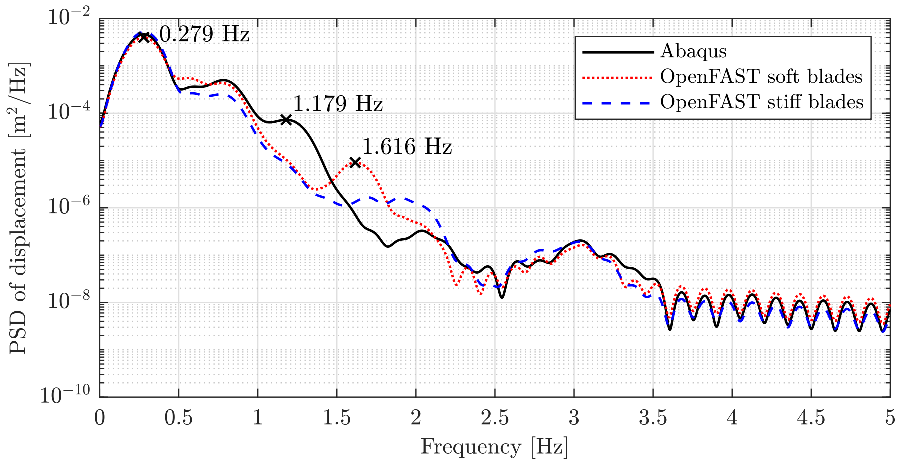


Figure 4.15: Power spectral density of displacement for the earthquake response of Abaqus full model compared to OpenFAST with both soft and stiff blades

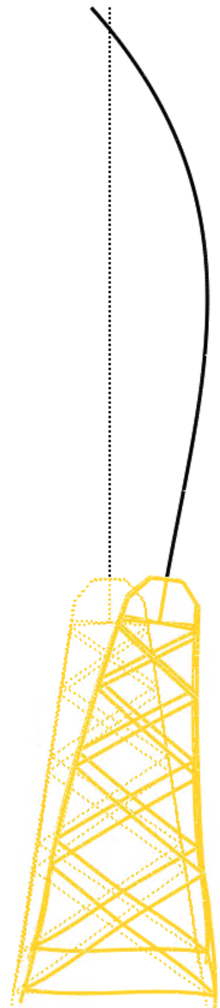
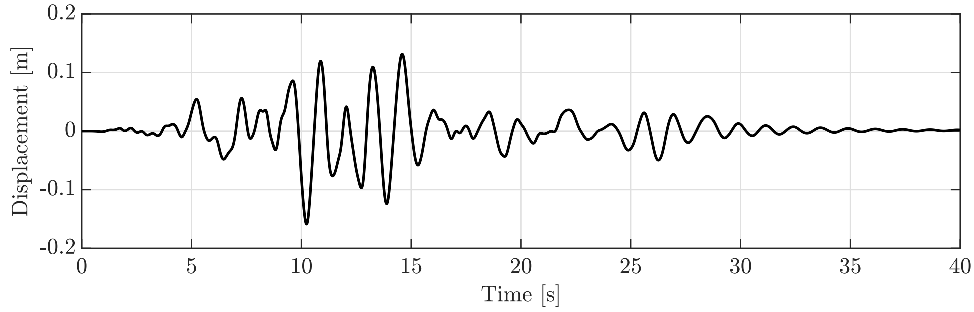


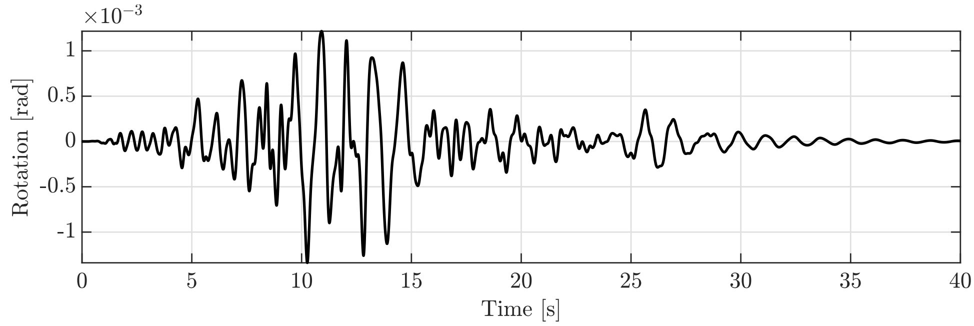
Figure 4.16: Abaqus 2nd fore-aft mode shape

4.3.3 Effect of jacket in multi-step method

To investigate the effect of the massless jacket in the multi-step method, it is performed an extraction of the kinematic interaction time series without the massless jacket. The resulting pile top motion and calculated forces are shown in figure 4.17 and 4.18. Comparing these results to those obtained with the massless jacket, see figure 4.6 and 4.4, shows that the change in peak displacement is small compared to the change in peak rotation. The stiffness of the jacket clearly counteracts the rotation of the pile head. However, the coupling of the displacement and the rotation springs gives a very small difference in the applied load. The difference of the peak load is 0.05 % less horizontal force and 0.7 % more moment without the jacket. This indicates that the use of a massless jacket to obtain the kinematic interaction response could be disregarded if the geotechnical model does not have access to the jacket geometry.



(a) Displacement at pile top during earthquake for only soil and piles



(b) Rotation of pile top during earthquake for only soil and piles

Figure 4.17: Displacement and rotation at pile top during a 30 seconds earthquake with free decay

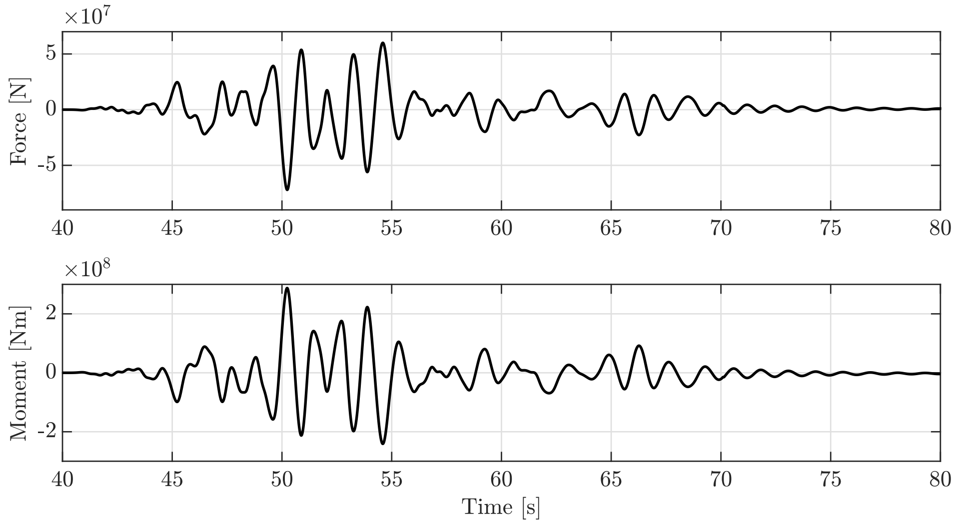


Figure 4.18: Force and moment time series applied at reaction nodes for only soil and piles. Peak force: 71981KN and peak moment: 287357kNm. Earthquake starts at time=40s

5 Results and Discussion

5.1 Case simulations

The OpenFAST model is used to simulate the following cases:

- **Production** - This case simulates normal production where the wind turbine is excited by both wind and waves. In this case the rotor is spinning and the OWT produces around 10MW of electricity.
- **Parked** - This case simulates a parked turbine on a sunny day with no wind. The OWT is only excited by the wave loads, as explained in section 4.2.2, and the rotor is not spinning. It could be argued that the wave spectrum should be changed to a Pierson-Moskowitz spectrum since it is not a situation with developing sea state. Although the focus of this project is on the response due to earthquake during different cases and for the ease of comparison; the same waves has been used. The rotational DOF controlling the rotation of the rotor is fixed to prevent the rotor from rotating.
- **Maintenance** - This case simulates a situation where the OWT has been stopped due to maintenance. In this case the OWT is excited by both wind and waves, but the rotational DOF for the rotor is the same as during the Parked case.

All the aforementioned cases have also been simulated with earthquake applied after 40 seconds. If something other than explained in the bullet points above is applied to the case, it is indicated; e.g., with *Production+EQ* to highlight that it is the production case with earthquake load applied. Some special cases are also simulated:

- **Increased earthquake** - The earthquake has been multiplied by a factor of three to simulate an increased earthquake intensity during the Production case.
- **Bi-directional earthquake loading** - Earthquake is applied in both directions with the same conditions as for the Parked case.

The actual response values is not the most interesting in these results, since the structural design is not fully adequate, as discussed in section 1.3. The focus of the results is the comparison of the load cases with and without earthquake load, highlighting the earthquake load effect on the structure. The considered results are; displacements at top of tower and transition piece, accelerations at top of tower, and overturning moment at the jacket base. The displacements are interesting both for investigation of the dynamic behaviour and the response magnitude. The tower top accelerations are interesting since it affects the workload on the RNA components, potentially leading to serviceability limitations contributing to a shorter operational lifetime of the turbine. Masses experience the total acceleration, while elastic forces only raise from the relative displacements, see section 2.2.4. The overturning moment is an important parameter of the pile design for offshore jacket structures and gives an indication of the stresses needed to be handled by the soil.

5.1.1 Production

Figure 5.1 shows the fore-aft displacement of the top of tower and transition piece, respectively. Looking at the displacements; the earthquake does not have a big impact on the magnitude of the response at the top of tower, namely that the response only vibrates around the reference production response. This means that the wind has a larger impact on the tower displacement compared to the earthquake. This is due to the large RNA mass counteracts some of the motion from the substructure, and that the aerodynamic force from the rotor pre-stresses the tower structure. The response at the transition piece; however, is much larger throughout the earthquake compared to the reference production case. Figure 5.2 shows the fore-aft displacement at the transition piece and the applied displacement at the base of the jacket. This shows how the jacket moves with the ground and that there is some amplification of the earthquake motion in the jacket. The effect from the wind on the jacket displacement is also clear, namely that the transition piece has a downwind displacement at the beginning of the earthquake (40 seconds).

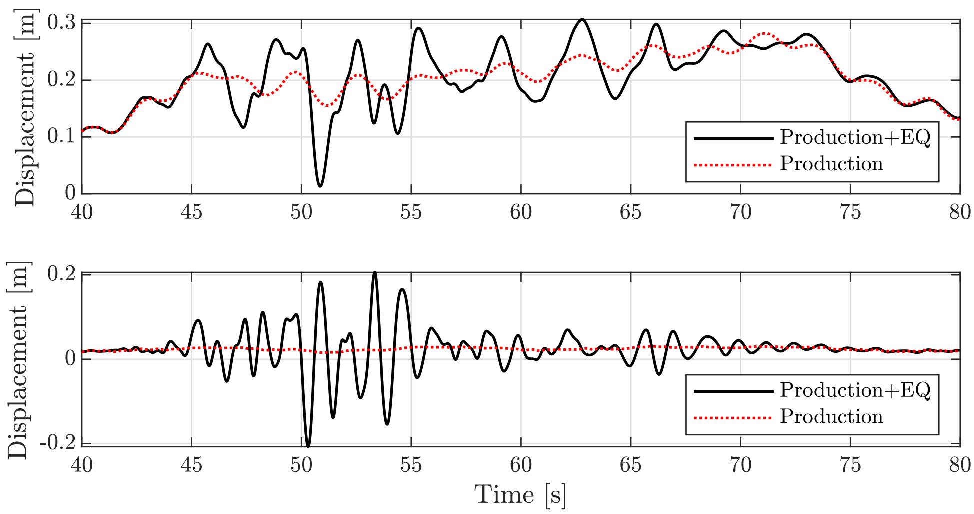


Figure 5.1: Downwind displacement for top of tower (upper plot) and transition piece (lower plot) for a production case

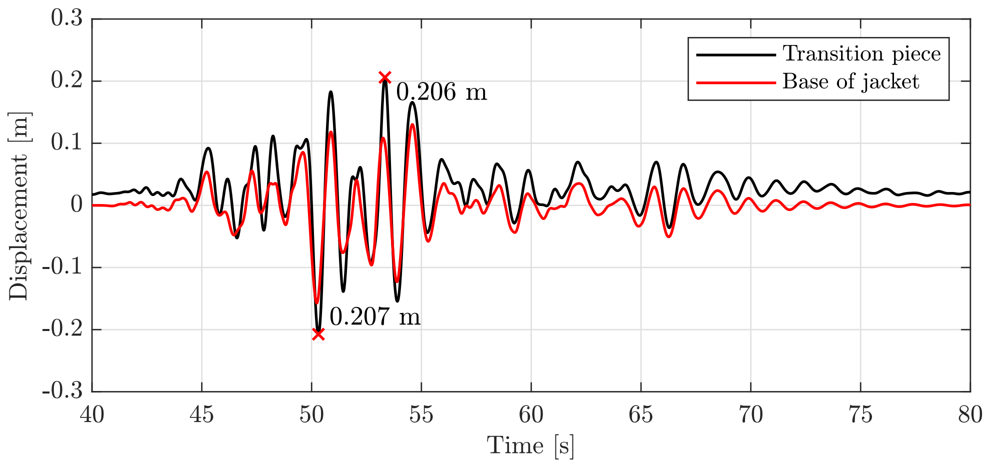


Figure 5.2: Transition piece displacement compared to the displacement of the base of the jacket

From figure 5.3, it is clear that the earthquake has a larger impact on the acceleration compared to the displacement at the top of tower. The acceleration shows 10 times larger magnitude during earthquake excitation in contrast to the reference case, while the displacement magnitude barely changes. Figure 5.4 shows the total overturning moment at the reaction nodes prior to and throughout the earthquake. From the figure it is evident that the earthquake has a large influence on the total loads acting on the structure compared to the reference production case. This should especially be investigated in a structural design setting.

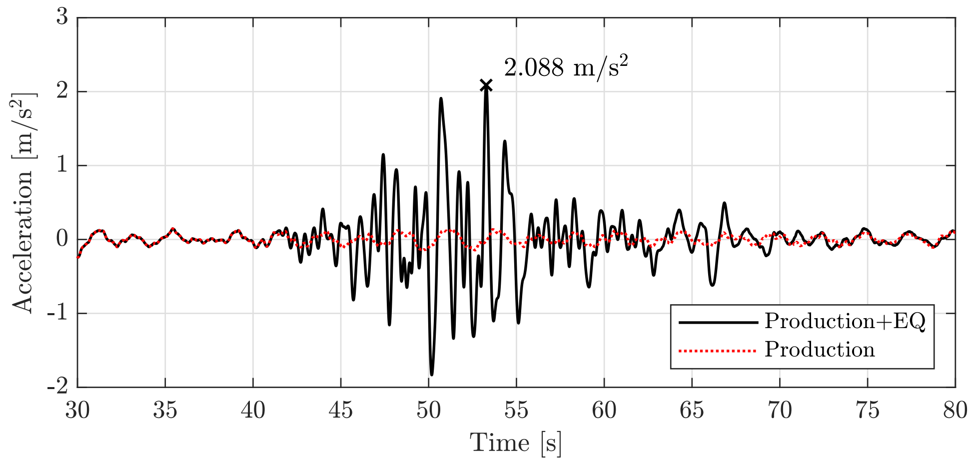


Figure 5.3: Fore-aft acceleration of the top of tower during earthquake

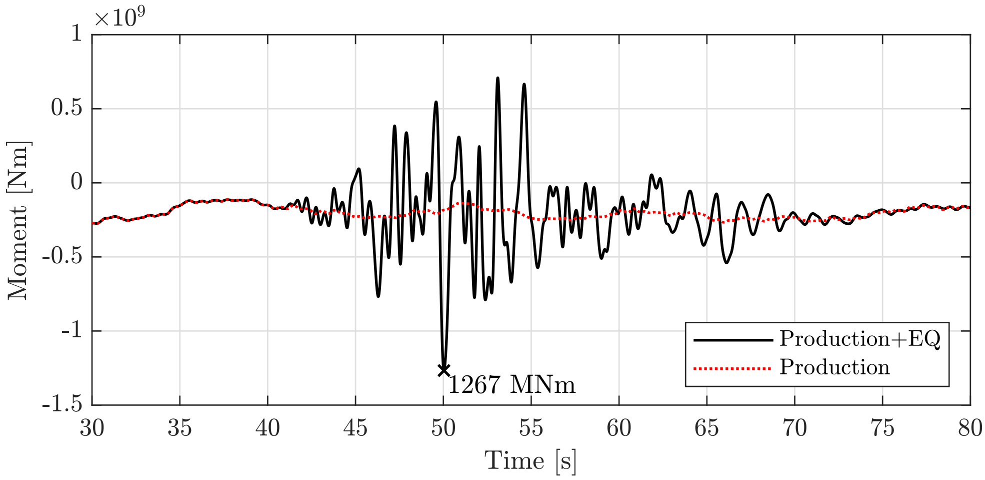


Figure 5.4: Overturning moment for a production case during an earthquake

5.1.2 Parked

The parked case simulates a situation where there are no wind loads acting on the turbine, but the full wave loads are applied as if there are a swell coming in from an old storm far away. Figure 5.5 shows the fore-aft displacement at the top of tower and transition piece, respectively. The figure shows when no wind is acting on the rotor, the heavy RNA gives the tower a steady-state displacement of around 0.22m forwards prior to the earthquake, which is the same as the initial condition at the top of tower. The same yields for the transition piece, which has its initial displacement of -0.006m. The dynamics of the response is like the production case, but the magnitude of the top of tower response is now heavily dependent on the earthquake load. The wind not pre-stressing the OWT structure gives a larger total displacement of the top of tower. Figure 5.6 shows how the transition piece also gets a larger peak displacement for the parked case.

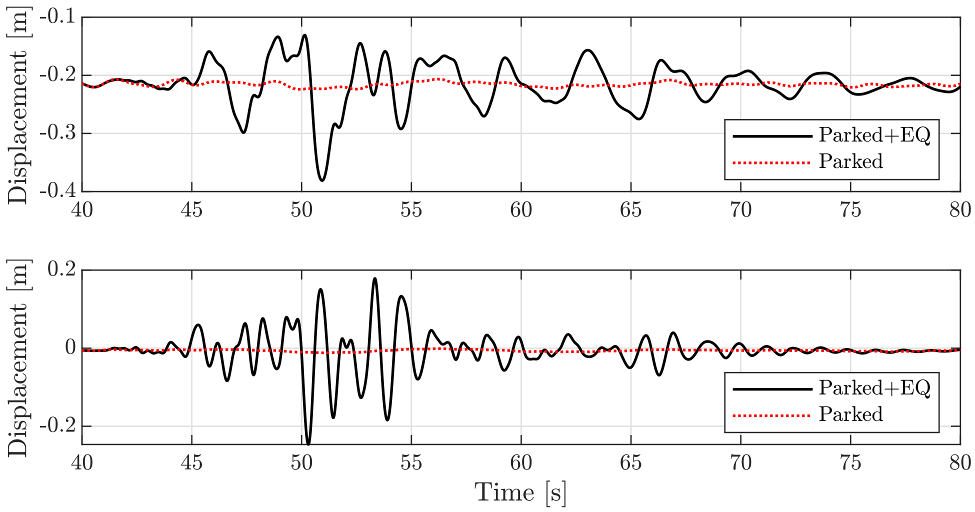


Figure 5.5: Fore-aft displacement at top of tower (upper plot) and transition piece (lower plot) for a parked case

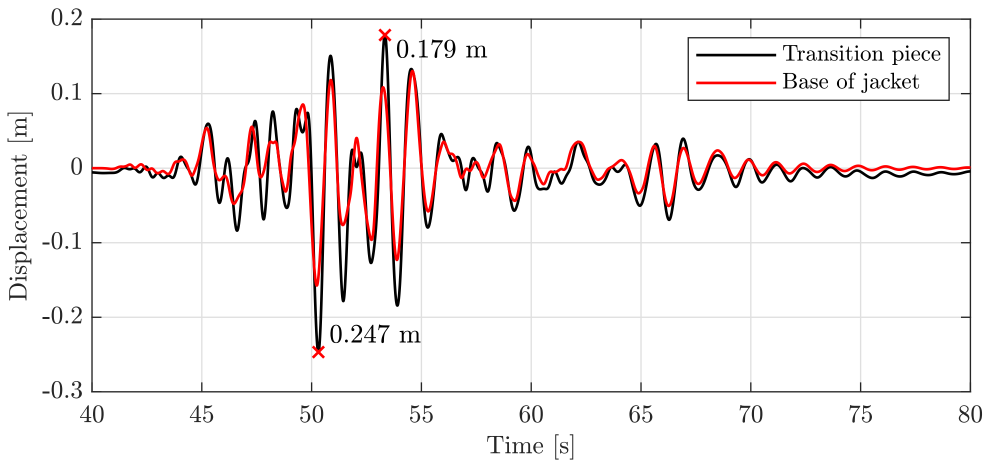


Figure 5.6: Transition piece displacement compared to the displacement of the base of the jacket

Figure 5.7 shows the fore-aft acceleration of the top of tower throughout the load case. The figure shows that the peak acceleration is higher compared to the Production+EQ case. This signifies that the aerodynamic wind force during production acts as a damper when the tower moves upwind, decreasing the acceleration when the tower moves upwind. When the tower moves along the wind, the aerodynamic wind force acts as an accelerating force during the Production+EQ case. This effect is shown as the peak acceleration in the Parked+EQ case is pointing in the opposite direction than in the Production+EQ case, and at an earlier point in the time series.

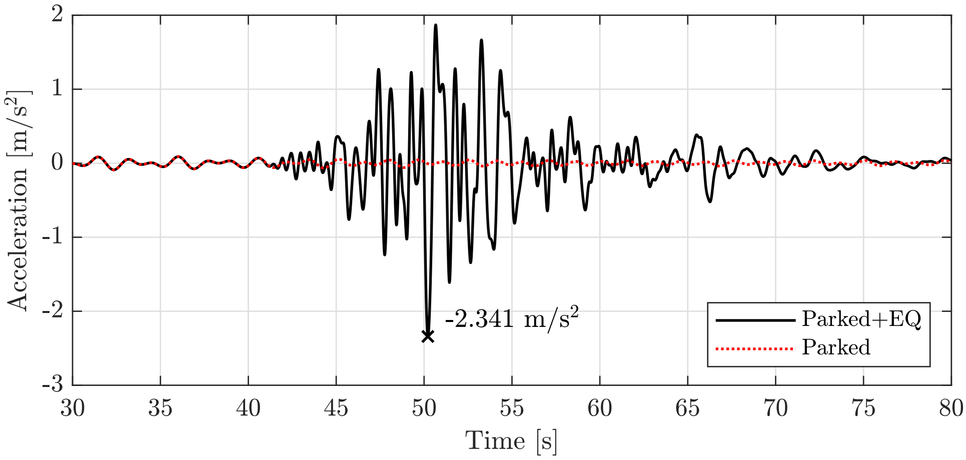


Figure 5.7: Fore-aft acceleration at the top of tower during earthquake for a parked situation

Figure 5.8 shows the overturning moment for the OWT prior to and throughout the earthquake. The figure shows a similar behaviour as for the Production+EQ case, meaning that the overturning moment is mainly due to the earthquake load. The peak overturning moment for the Parked+EQ case is 1129MNm, only 10% lower than the Production+EQ case.

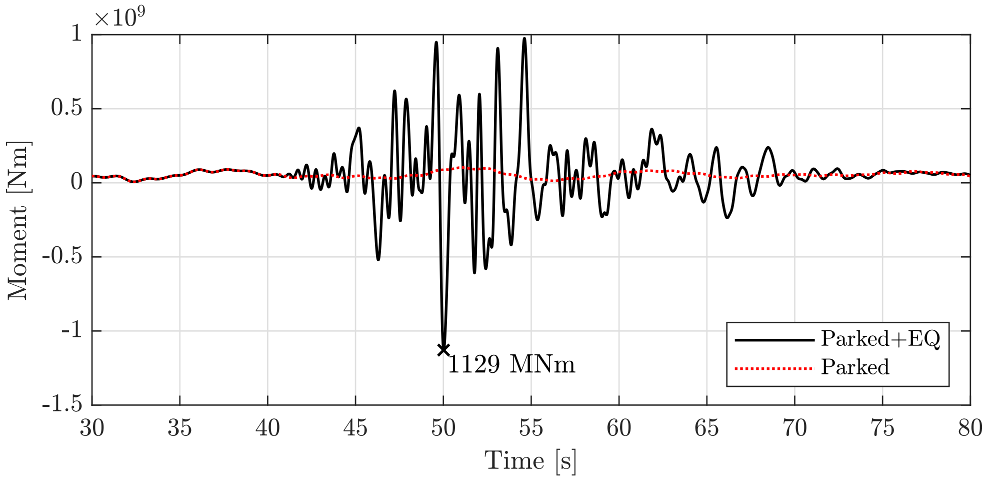


Figure 5.8: Overturning moment during a parked case

5.1.3 Maintenance

Figure 5.9 shows the fore-aft displacement at the top of tower and transition piece. It shows the same as for the two other cases, but highlights the effect of the production aerodynamic force by looking at the steady-state displacement. The Maintenance case gets a steady-state displacement of the tower top of around -0.18m, while the tower top in the Production case is fluctuating on the opposite side of the initial tower axis with displacements of +0.1m to 0.3m.

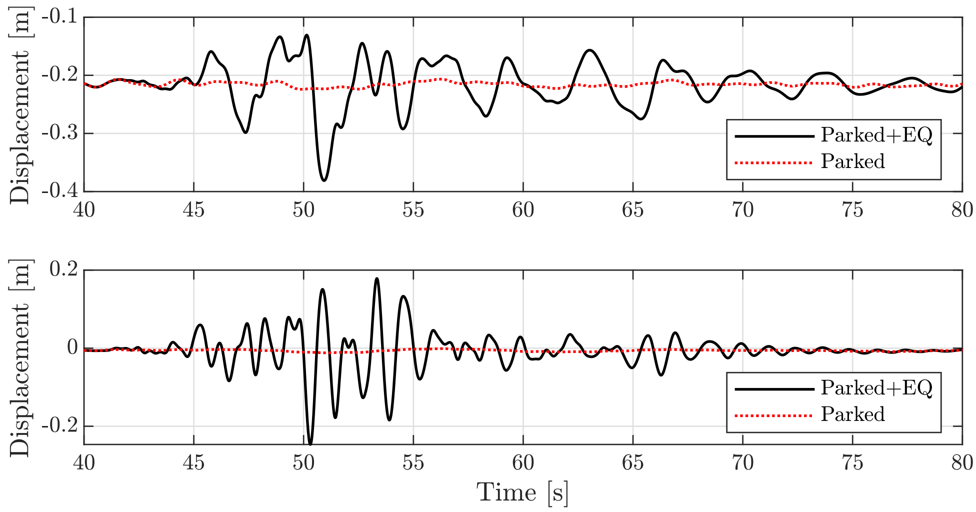


Figure 5.9: Fore-aft displacement of top of tower (upper plot) and transition piece (lower plot) for a maintenance case

Figure 5.10 shows the displacement of the transition piece and the applied displacement at the base of the jacket. The figure shows that the transition piece has almost the same displacement as for the Parked case throughout the earthquake. The two cases are almost identical, except for the wind on the parked blades. The load from these blades is almost negligible in the Maintenance case and the numbers show that the two cases are almost identical when it comes to the vibration of the transition piece.

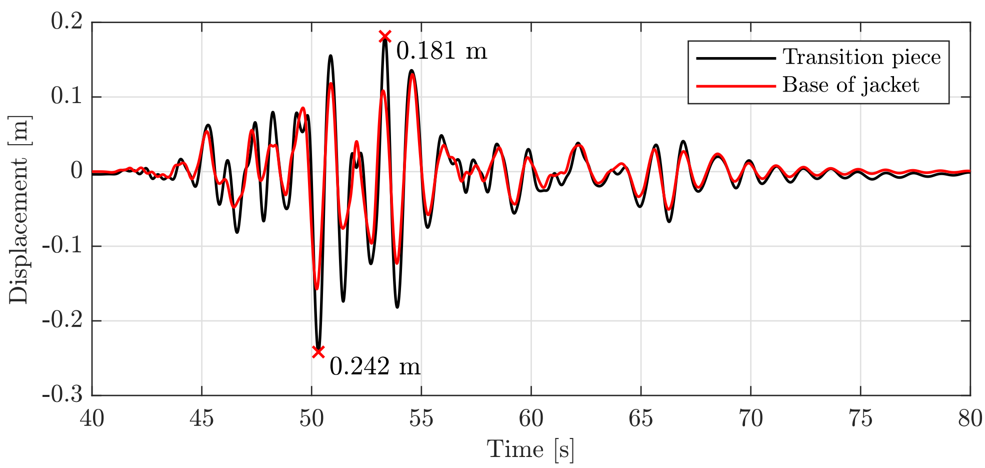


Figure 5.10: Transition piece displacement compared to the displacement of the base of the jacket

Figure 5.11 shows the acceleration of the top of tower in fore-aft direction throughout the Maintenance case. The figure shows that the peak acceleration is lower for this case compared to the Parked case. When the tower moves upwind, the wind damps the acceleration, and when the tower moves downwind, the wind amplifies the acceleration.

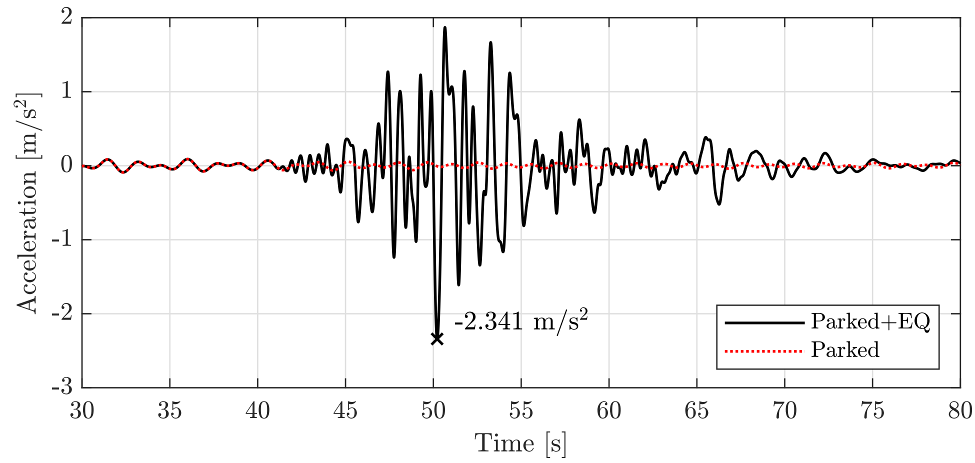


Figure 5.11: Fore-aft acceleration for the top of tower during earthquake

Figure 5.12 shows that the overturning moment decreases due to the wind on the parked blades. This is due to that the wind pushes the tower backwards and therefore moves the mass of the RNA closer to the undeformed tower axis, giving it a shorter arm to create overturning moment.

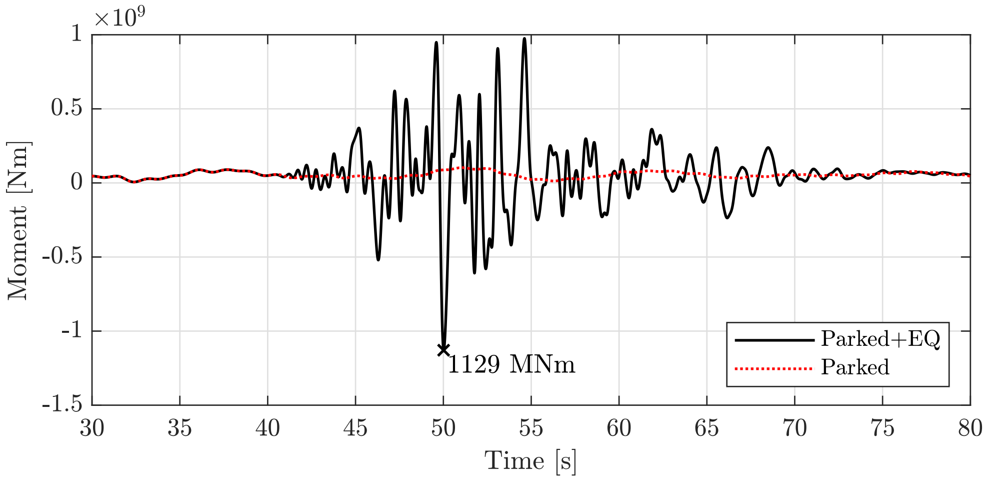


Figure 5.12: Overturning moment during a maintenance case

5.1.4 Comparing Production, Parked and Maintenance during earthquake

In this section, the displacement and overturning moment during the three cases are compared to highlight the differences. Figure 5.13 shows the top of tower fore-aft displacement. The figure shows how the earthquake response is somewhat independent of the wind load during excitation, although the total displacement differs a lot depending on the wind load. Figure 5.16 shows the overturning moment for the three cases and it is evident that the earthquake has a large impact on the forces in the jacket base. The figure also shows that the difference in overturning moment amplitude between the three cases is not severe, even though the Production case has the unfavorable production force applied. The force from the RNA during production has a large moment arm (total height of the structure) compared to the force from the RNA weight (overhang of the RNA). Figure 5.14 shows the top of tower acceleration for the three cases and clearly manifests that the acceleration response is dominated by the earthquake load. Figure 5.17 and 5.15 shows an enlargement of the overturning moment and acceleration time series for the most intensive 15 seconds of the earthquake response.

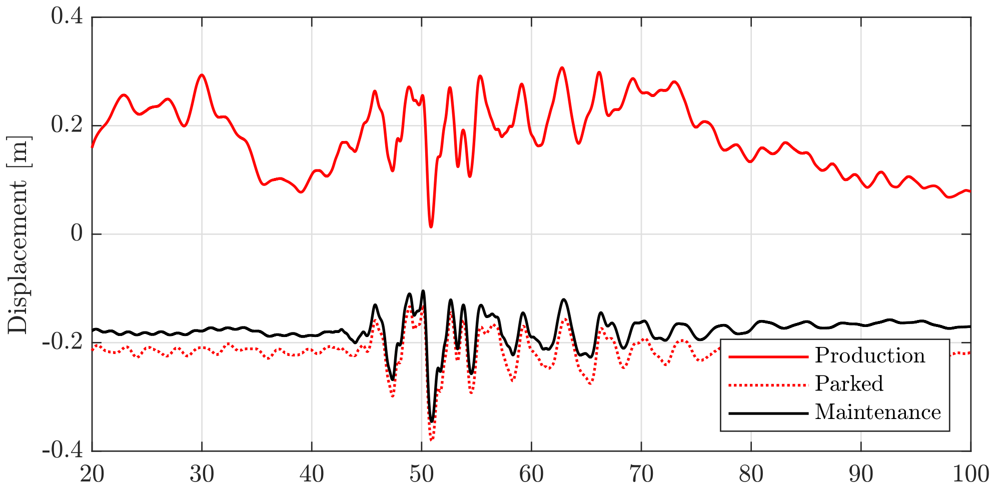


Figure 5.13: Top of tower displacement for the three cases with earthquake load applied after 40 seconds

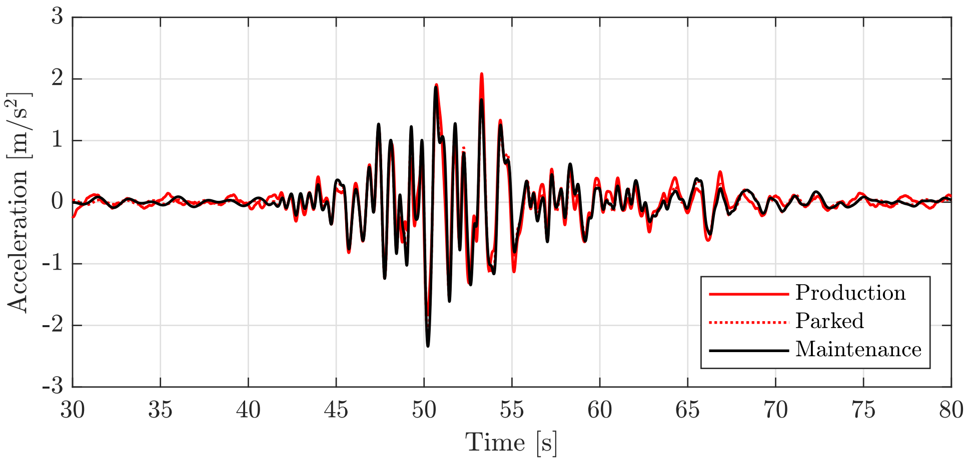


Figure 5.14: Acceleration of the top of tower for the three cases during earthquake

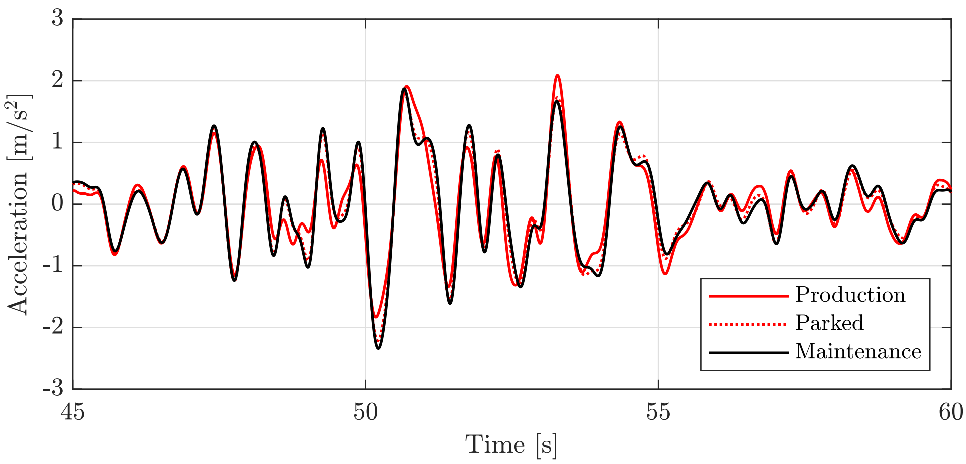


Figure 5.15: Acceleration of the top of tower for the three cases during earthquake on an enlarged time scale

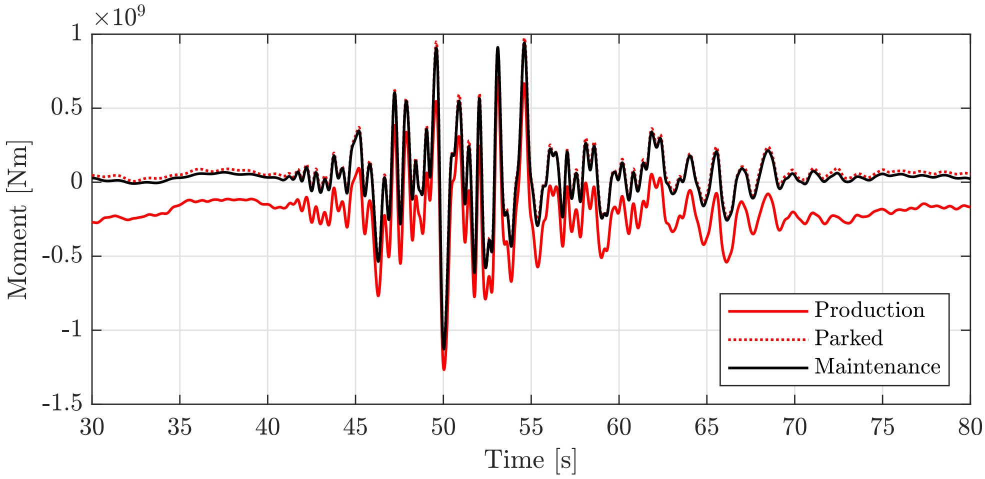


Figure 5.16: Overturning moment for the three cases during earthquake

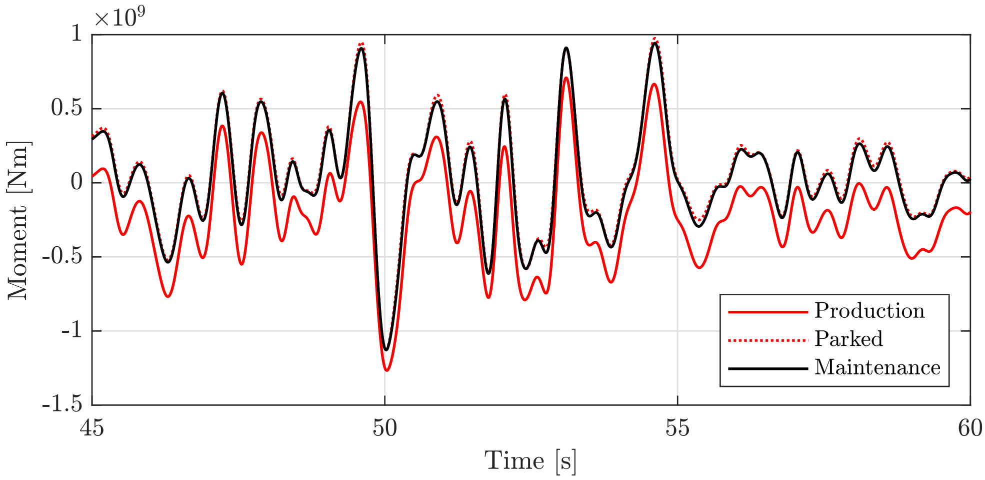


Figure 5.17: Overturning moment for the three cases during earthquake on an enlarged time scale

5.1.5 Increased earthquake amplitudes

This case investigates the effect of increased earthquake amplitudes. The earthquake applied in the aforementioned cases is now multiplied by a factor of three. The implementation in OpenFAST is to multiply the kinematic interaction motion at the top of the piles, and then calculate the new forces. Due to linear soil behaviour, a new kinematic interaction simulation is not needed. Figure 5.18 shows the top of tower and transition piece response compared to the original earthquake. It is seen that for increased earthquake amplitudes, the displacement response is still not outranging the response from the wind, such as it is seen for the accelerations. Figure 5.20 shows the acceleration response for the increased earthquake, and the peak acceleration (PA) is nearly linearly increased if compared to the original earthquake response;

$$\mathrm{PA}_{\mathrm{original}} = 2.088\mathrm{m/s}^2, \quad \mathrm{PA}_{\mathrm{increased}} = 6.069\mathrm{m/s}^2 \Rightarrow \mathrm{PA}_{\mathrm{increased}}/\mathrm{PA}_{\mathrm{original}} = 2.91$$

The peak response will not be fully linear as the influence from the aerodynamic rotor load is based on the relative displacement and velocity of the blades. Figure 5.21 shows that the increase in earthquake strength by a factor 3 increases the peak overturning moment from 1267MNm up to 3458MNm, which is an increase by a factor of 2.73.

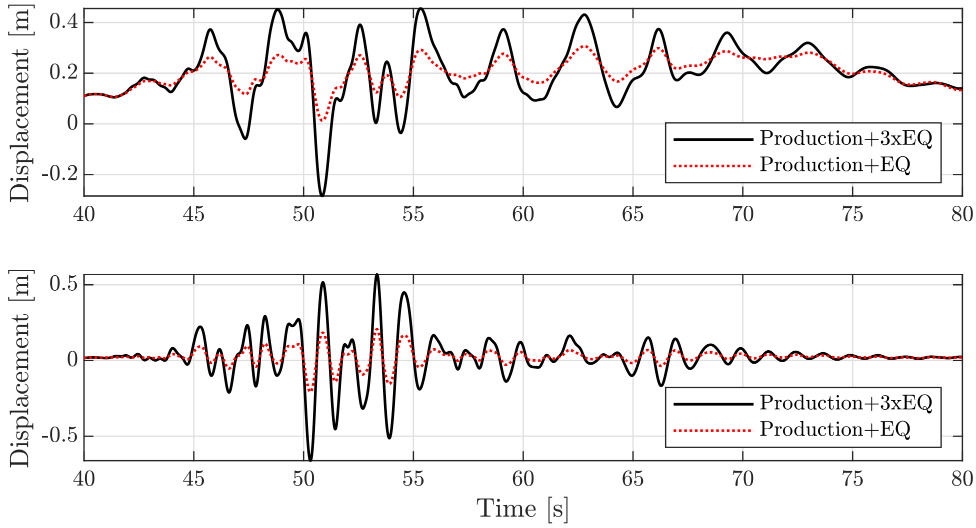


Figure 5.18: Fore-aft displacement for top of tower (upper plot) and transition piece (lower plot) for an increased earthquake

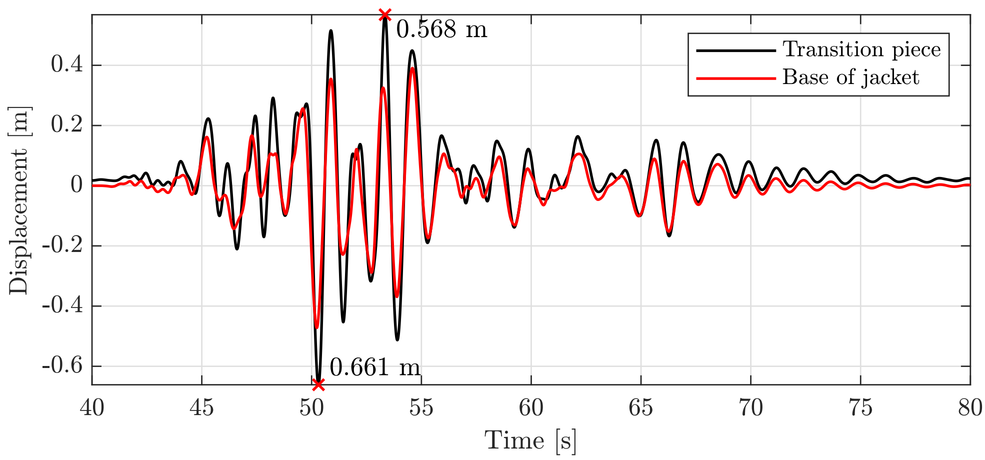


Figure 5.19: Transition piece displacement compared to the displacement of the base of the jacket

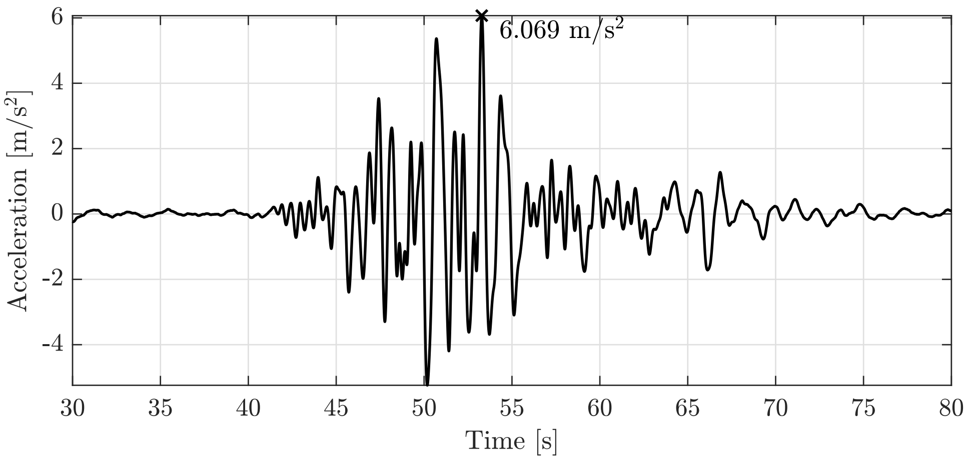


Figure 5.20: Acceleration for the top of tower during an increased earthquake

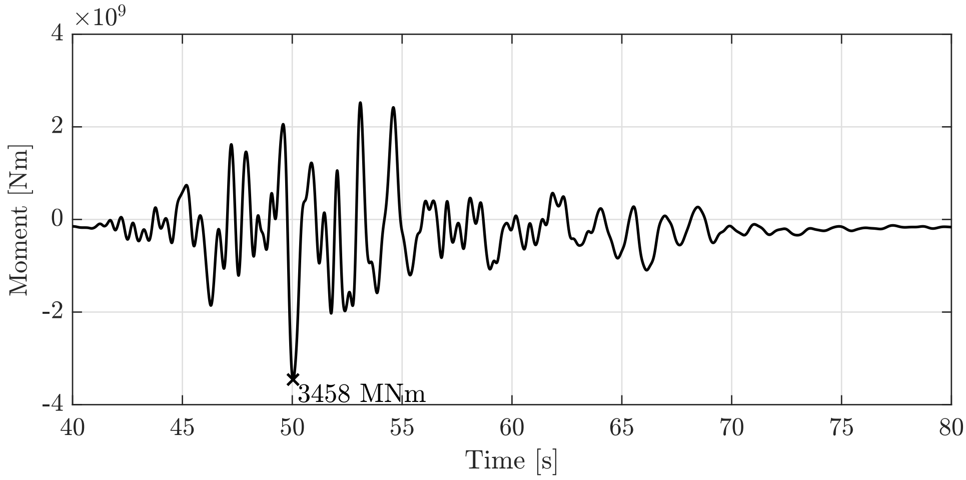


Figure 5.21: Overturning moment for an increased earthquake

5.1.6 Bi-directional earthquake loading

This case investigates the response when the Loma Prieta N-S motion is applied in the fore-aft direction, and the E-W motion is applied in the side-side direction. The calculation of the earthquake forces and moments according to (4.2.2), with $u_z(t) = \theta_z(t) = 0$, then yields:

$$
\begin{aligned}
F_x(t) &= k_{xx} \cdot u(t) + k_{x\theta_y} \cdot \theta_y(t) \\
F_y(t) &= k_{yy} \cdot u(t) + k_{y\theta_x} \cdot \theta_x(t) \\
M_x(t) &= k_{\theta_x y} \cdot u(t) + k_{\theta_x\theta_x} \cdot \theta_x(t) \\
M_y(t) &= k_{\theta_y x} \cdot u(t) + k_{\theta_y\theta_y} \cdot \theta_y(t)
\end{aligned}
$$

These loads are then applied in the same way as explained in section 4.2.1.

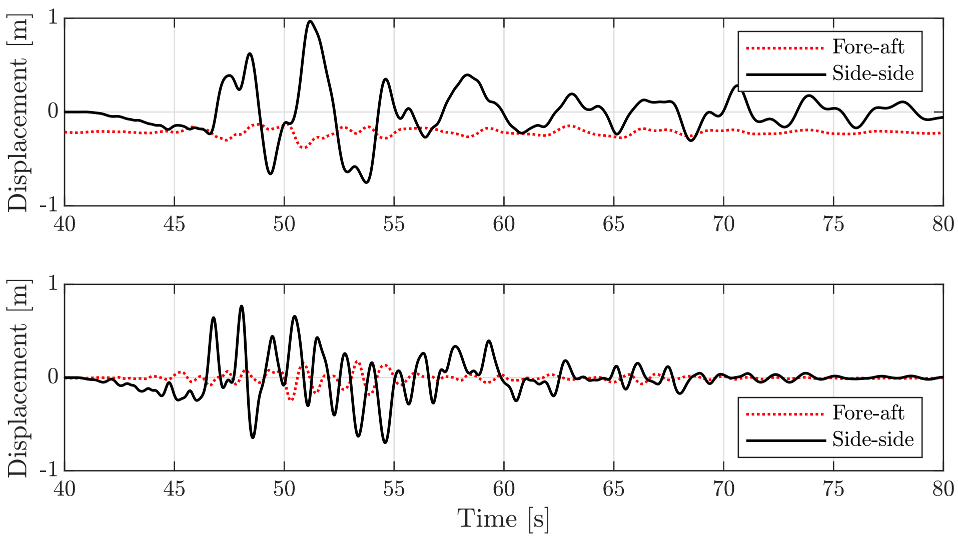


Figure 5.22: Displacement time series for tower top and transition piece when the earthquake is applied in both directions

Figure 5.22 shows the displacement in both fore-aft and side-side direction to highlight the difference between the directions for top of tower and transition piece, respectively. Figure 5.23 illustrates the tower top motion in the x-y-plane. The latter shows how top of tower achieves a 2.53 times larger displacement amplitude in the side-side (y-) direction. This is a result of the

E-W earthquake motion amplifying the soil displacement more than the N-S earthquake motion due to frequency content greater coinciding with the first natural frequency of the soil. Also, the OWT has shown to generally yield larger displacements in the side-side direction due to the RNA geometry. Figure 5.24 and 5.25 shows that the overturning moment about the y-direction does not change much compared to the Parked+EQ case, but the overturning moment about the x-axis shows that the moment is 3.5 times larger about the x-axis. The overturning moment about the x-axis is also even larger than the moment about the y-axis for the increased earthquake case. This highlights how well the E-W direction excites the first natural frequency of the soil compared to the N-S direction. This is also evident for the acceleration shown in figure 5.26 and 5.27, where the side-side acceleration is larger than for the fore-aft direction.

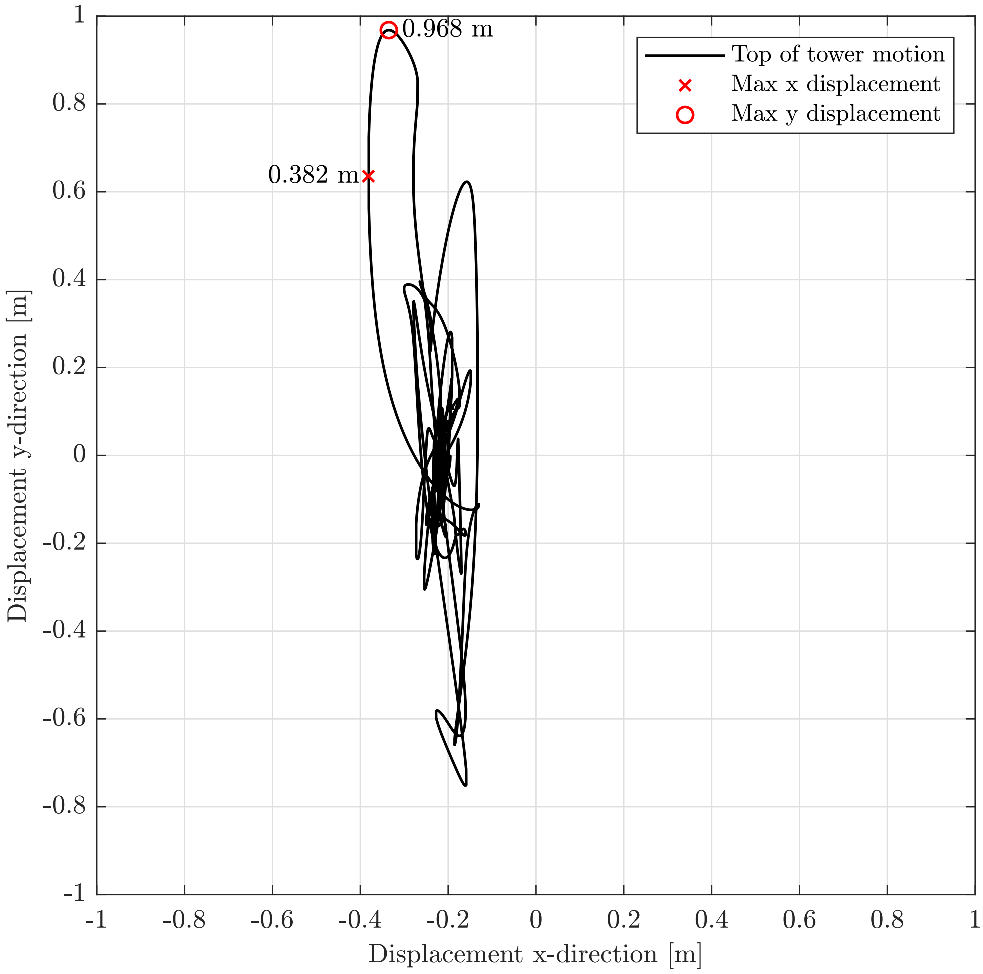


Figure 5.23: Motion of the tower top during earthquake applied in both directions

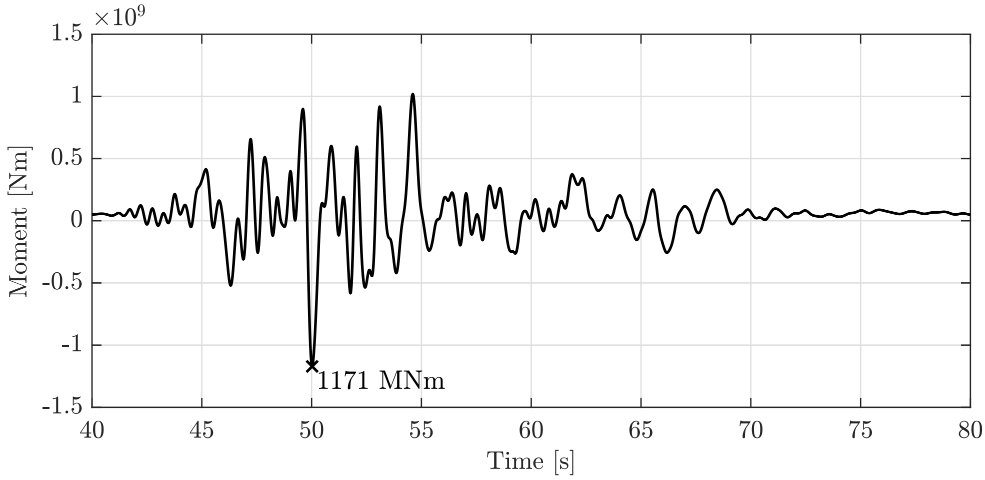


Figure 5.24: Overturning moment about the y-axis when the earthquake is applied in both directions

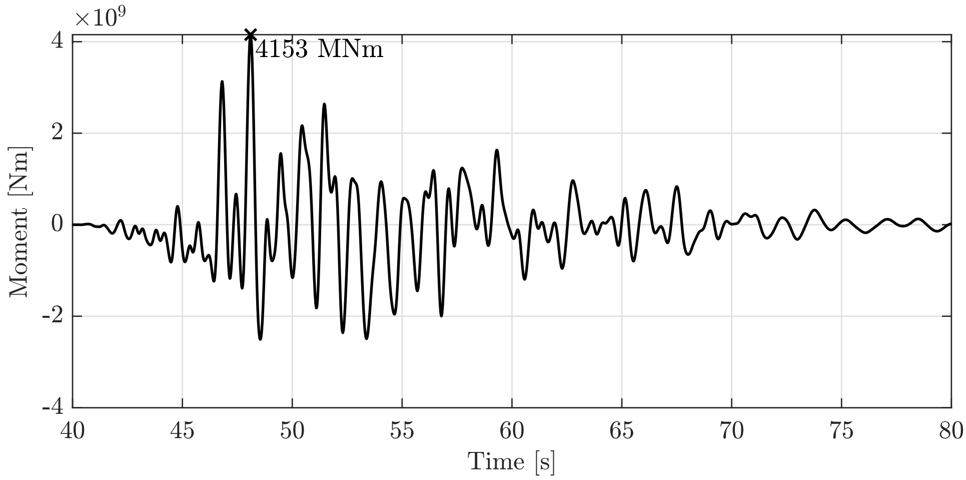


Figure 5.25: Overturning moment about the x-axis when the earthquake is applied in both directions

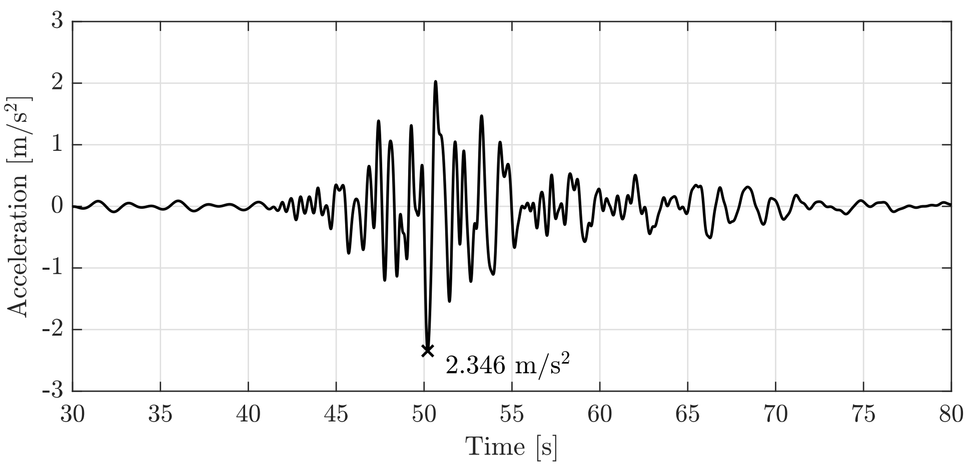


Figure 5.26: Fore-aft acceleration at the top of tower during earthquake in both directions

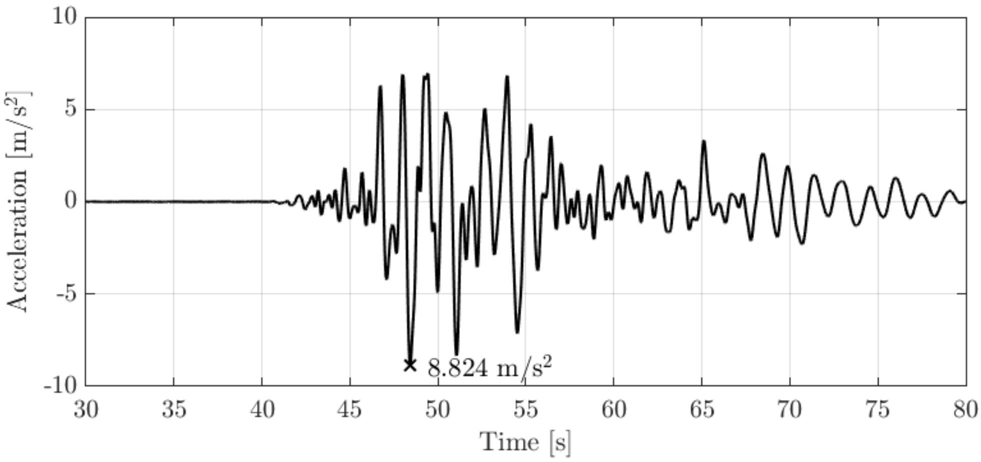


Figure 5.27: Side-side acceleration at the top of tower during earthquake in both directions

5.2 Soil non-linearity analyses

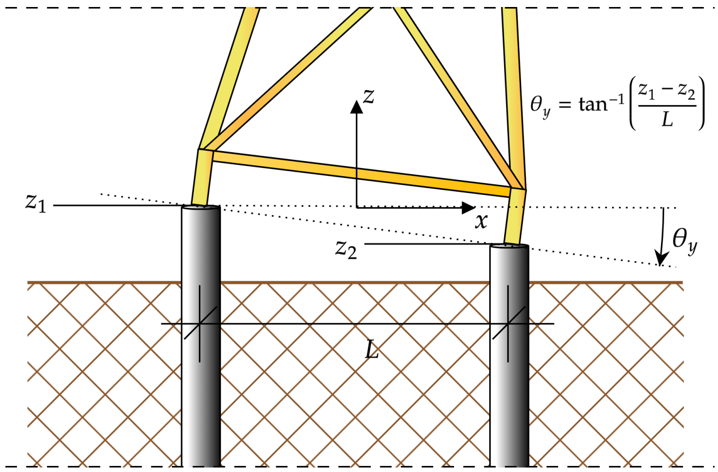


Figure 5.28: Illustration of *tilt*, θ_y

Using the integrated Abaqus model with non-linear soil behaviour according to the defined Mohr-Coulomb plasticity model, the main focus is to obtain results on the permanent tilt of the jacket, see figure (5.28). In this context, three load cases are examined:

- **Load Case I**: Earthquake excitation in x-direction at bedrock by the Loma Prieta N-S accelerations, and a 2MN concentrated force in x-direction at top of tower.
- **Load Case II**: Earthquake excitation in x-direction at bedrock by the Loma Prieta N-S accelerations times a factor of 3, and a 2MN concentrated force in x-direction at top of tower.
- **Load Case III**: Earthquake excitation in x-direction at bedrock by the Loma Prieta N-S accelerations, earthquake excitation in y-direction at bedrock by the Loma Prieta E-W accelerations, and a 2MN concentrated force in x-direction at top of tower.

The concentrated force is simulating an average production force from the RNA. The force is constant over the time, as the wind-induced production forces is assumed almost static during the earthquake excitation. No environmental loads are applied. The choice of load cases aim to show how increased earthquake intensity, and more earthquake activity affects the permanent tilt accumulation during earthquake loading. The analysis is performed in three steps:

- **Apply gravity step**: A static linear analysis applying gravity.
- **Load step**: Implicit dynamic analysis including non-linear effects, applying earthquake and concentrated force.
- **Decay step**: Implicit dynamic analysis including non-linear effects and gravity only.

The decay step is performed to make the elastic tilt diminish before measuring permanent tilt. The results are presented in figure 5.29 and 5.30, showing pile-head one and two vertical displacement, and resulting tilt, respectively.

It should be mentioned that the application of gravity to the full model will yield settlements in the soil, larger than for the piles. The soil elements tied to the piles will then experience *negative friction*. This effect is rather small, only present at the top layers, but the correct way would be to first establish the soil in-situ stresses from gravity, giving no initial displacement. The pile should then be attached to the soil, before assembling the OWT structure and gravity. The negative friction effect is investigated further down in this section.

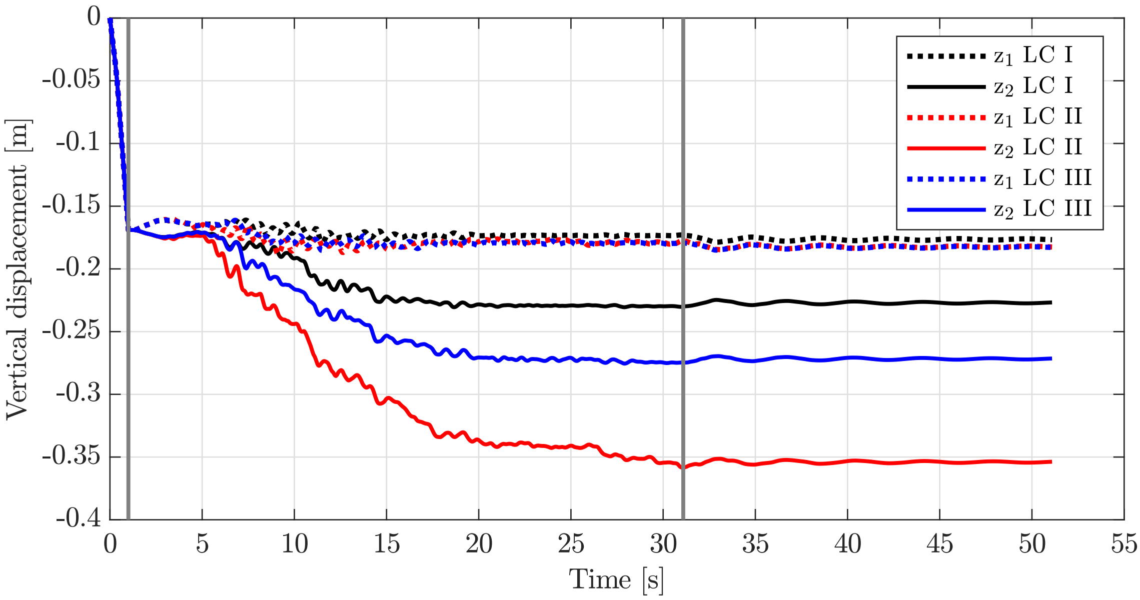


Figure 5.29: Pile-head vertical displacement for all load cases (LC). Grey vertical lines indicating step separation.

From figure 5.29 and 5.30 it is seen that the application of the concentrated force at top of tower makes an initial tilt, partly counteracted by the dynamic response of the tower. The concentrated force should have been applied in an own step before applying the earthquake. In that way the initial tilt from the concentrated force would be stabilized before the earthquake hits, which is a more realistic situation. After five seconds, the earthquake loading escalates and creates yielding shear stresses in the soil. Due to the concentrated force, tilt then starts to accumulate. The accumulation of tilt goes on until about 20 seconds for load case I and III, and until about 25 seconds for load case II. The stop in accumulation indicates that the earthquake motion has decreased to a level where the soil is not yielding any more. When the concentrated force is turned off, the oscillating tilt behaviour indicates elastic rocking of the OWT, and the permanent tilt is revealed when the tower rocking is damped out. Both the increased earthquake amplitudes and the bi-directional earthquake excitation clearly give raise to more yielding of the soil, making larger accumulation of permanent tilt. The factor 3 larger earthquake acceleration amplitudes in load case II compered to load case I, gives a factor 3.4 larger permanent tilt. A tilt of 0.288 deg, as obtained for load case II, gives a tower top horizontal displacement of about 0.9 m. An analysis without earthquake excitation is also run, resulting in no permanent tilt.

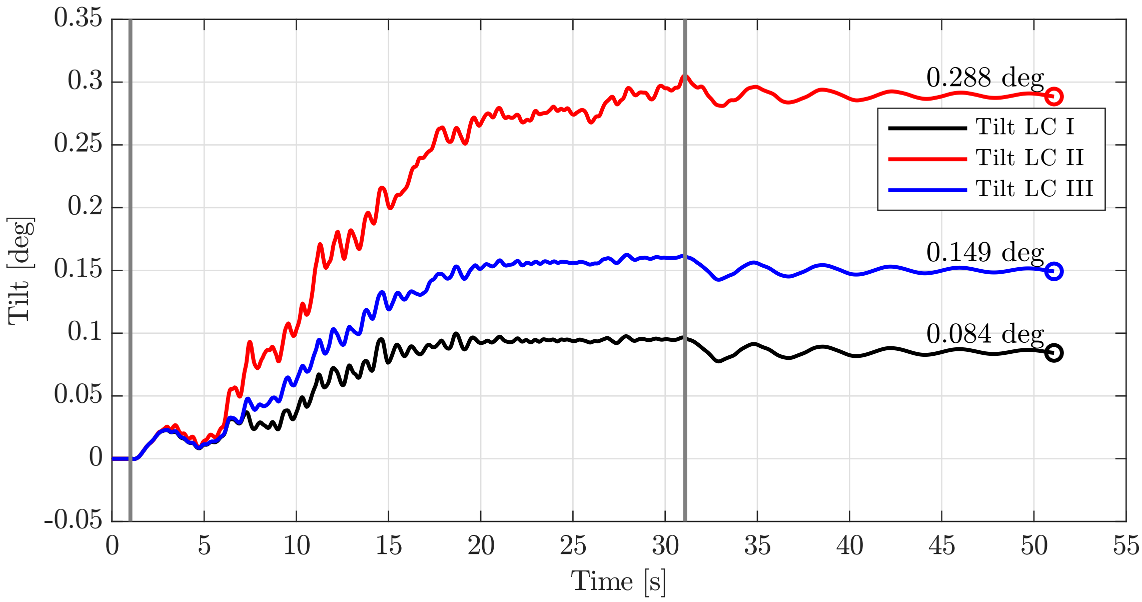


Figure 5.30: Tilt for all load cases (LC). Permanent tilt after decay highlighted with number. Grey vertical lines indicating step separation.

From the given yielding criterion, it is stated that the soil material starts to yield when the maximum shear stress, $\tau_{\max}$, reaches the material cohesion. The yielding criterion is therefore independent of the direction of the shear stress. To get permanent tilt, the yielding must take place in the vertical direction, allowing pile settlement. The horizontal earthquake excitation will give horizontal shear stress in the soil layers, which also gives vertical shear stress on each element due to equilibrium. To investigate the development of vertical shear stresses, the element vertical shear stress, τ_{xz}, during load case II is plotted for chosen elements alongside the pile in a vertical section of the soil, see figure 5.31. As the soil elements (C3D8R) uses the reduced Gaussian quadrature rule for integration of stresses, they are only obtained for one point (the centroid) in each element. This means that the shown stress is the same for the whole element. For the elements shown in the figure, positive shear stress equals forces pulling downwards on the left side of the element. The figure clearly shows how the bad gravity application gives negative friction in the two upper elements; forces pulling upwards on the left side of the elements. When the right pile (z_2) starts to settle, see figure 5.29, the negative friction is counteracted, and the shear behaviour starts following the dynamics of the soil as for rest of the lower elements. The yielding of the soil is clearly illustrated by the shear stress reaching a maximum at the level of cohesion for each layer. The lower elements inhabits less yielding as their high effective vertical stress gives a much higher cohesion than for the upper layers. It should be mentioned that when the node-to-surface tie formulation is used, stress accuracy at the tie surface is not optimized, potentially giving some less accurate results for the stresses at the pile/soil intersection.

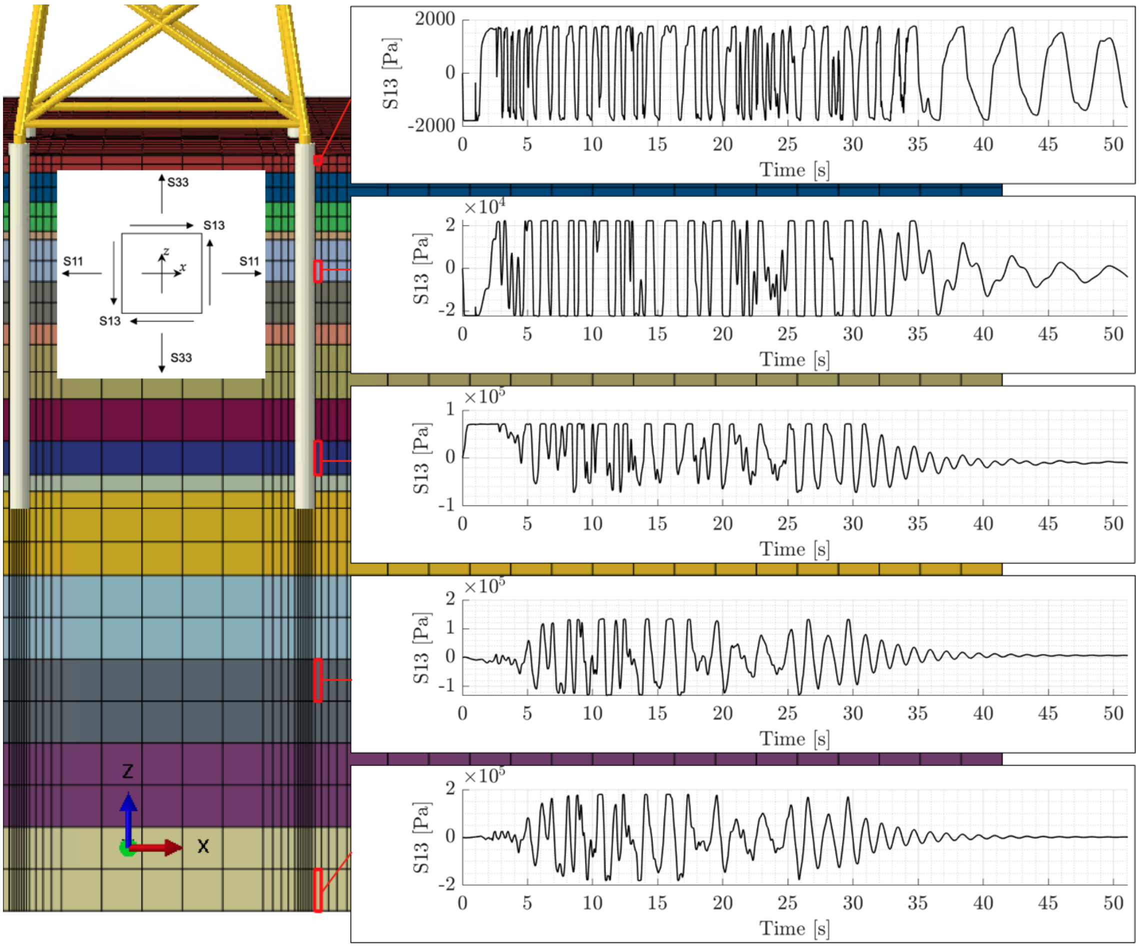


Figure 5.31: Load case II vertical shear stress for chosen elements. Abaqus positive stress naming convention for the x-z-plane shown to the left. S11 = σ_x, S33 = σ_z and S13 = τ_{xz}.

6 Conclusion and further work

6.1 Conclusion

In this thesis, the implementation of SSI-effects in the computational software OpenFAST with the use of a multi-step method and a complementary integrated Abaqus model is shown and verified. OpenFAST is a custom made, advanced and efficient software for hydro-aero-servo-elasto-dynamic analyses of wind turbines. The integrated Abaqus model is also used to investigate permanent tilt of the jacket base due to earthquake and soil non-linearities. The turbine and jacket structure studied is an offshore wind turbine based on the IEA 10-MW reference OWT situated on the piled INNWIND reference jacket. The soil modeling in Abaqus utilizes tied degree of freedom (TDOF) for the lateral boundaries, and a fixed bedrock boundary. This approach with the location of the lateral boundaries 83m from the pile center, shows a good match with the free field motion, with an error estimate of R=1.4%. The soil model, including piles, is used to extract jacket base-attachment stiffness for the OpenFAST model. To calculate earthquake excitation forces for the OpenFAST model, kinematic interaction displacements extracted from the Abaqus model is used. For the latter, a jacket with no mass, in addition to the piles, is included in the soil model, and it is excited by earthquake accelerations at the bedrock. A simpler method for computing this information is to exclude the massless jacket. Therefore, the presence of the massless jacket is investigated, and results show negligible effect; namely 0.05% more horizontal force and 0.7% less moment. Thus, if the jacket geometry is not available in the geotechnical model, the jacket could be neglected for the kinematic motion extraction. Comparing horizontal earthquake excitation response, the OpenFAST model reproduces the results from the Abaqus integrated model satisfactorily. The OpenFAST model with implementation of the presented multi-step method is therefore proven valid for simulating cases with earthquake loading.

With use of the OpenFAST model, different cases including combined effect of environmental and earthquake loading is simulated. It reveals that the operational wind-induced forces dominate the top of tower displacements, while the earthquake load dominates the top of tower accelerations and overturning moment at the jacket base. This indicates that the important design concerns with respect to earthquake load is for the acceleration limits at the RNA and for the overturning moment needed to be carried by the soil and piles. The simulation of bi-directional earthquake loading reveals that the side-side direction is more exposed for the aforementioned effects than in the fore-aft direction.

The investigation of permanent tilt with the Abaqus integrated model, including a Mohr-Coulomb plasticity model for the soil, is conducted by considering three load cases. All the cases includes gravity load and a production force at the top of tower, but with different earthquake excitation; (I) uni-directional excitation by Loma Prieta N-S accelerations (PGA = 1.2 m/s^2), (II) uni-directional excitation by Loma Prieta N-S accelerations multiplied by three (PGA = 3.6 m/s^2), and (III) bi-directional excitation by Loma Prieta N-S and E-W accelerations (PGA = 1.2 and 1.0 m/s^2). The results show how the production force at the top of tower accumulates permanent tilt during earthquake loading due to yielding shear stress in the soil. Larger earthquake activity gives more permanent tilt, and the results especially highlights that the permanent tilt increases faster than the earthquake amplitude. Load case II gives a factor 3.4 larger permanent tilt than load case I. The increased earthquake intensity creates more overall shear stress, generating more yielding of the soil, giving a larger tilt accumulation rate with time.

6.2 Recommended further work

With the given SSI-method implemented in OpenFAST and approved, several interesting analyses could be done. Also further development of the SSI-method would be an interesting study. The list below summarizes some of the main points remaining for further work in the OpenFAST model:

- Jacket fatigue analysis
- Application of different earthquakes and in different directions

- Utilize the capabilities of loads from currents and second order waves in HydroDyn
- Accommodate different soil profiles by changing the stiffness of the SSI-springs
- Implementation of other SSI-effects like radiation damping and added mass
- Implementation of non-linear SSI-springs

An earthquake could seriously shorten the lifespan of a jacket substructure from a fatigue perspective. An investigation of fatigue due to earthquake loads should be a suited study with the modelling framework made in this project. The effects of bi-directional earthquake loads is not fully investigated in this project and these effects should be further looked at. The response of a turbine only excited in the side-side direction is also not fully investigated. The hydrodynamic loads applied in the OpenFAST model are only loads from random regular waves, but the HydroDyn module also has the capabilities of calculating currents and second order waves. The effects of currents in conjunction with earthquake could lead to larger displacements or act as a dampening effect. By changing the stiffness of the SSI-springs in the SubDyn module, the natural frequencies of the model would change. This is analogous to model different soil profiles as this has an effect on the amplification in the soil. This could lead to larger displacements and accelerations for the top of tower during earthquake excitation, just as the E-W motion showed by hitting the first natural frequency of the soil more than the N-S motion. Soil radiation damping and soil added mass are SSI-effects that in conjunction with the soil-springs would yield a more exact solution of the OWT earthquake response, but the lack of these effects is conservative. With the implementation of non-linear SSI-springs in OpenFAST the program could be able to simulate the tilt of the jacket base during earthquake excitation. Since OpenFAST is a more efficient program for simulation of turbine response, the non-linear SSI-springs could help investigating the permanent tilt of the OWT faster than the integrated Abaqus model.

The non-linear Abaqus analysis of the integrated model is a time-consuming process, and several interesting implementations and load cases were not conducted in this project due to time limitations. Although, the model is now easily established by the given python-scripts, see appendix C, and further work should examine the implementation of:

- Other earthquake time series
- Other soil profiles
- Dynamic load at the tower top
- Environmental loads: Submerged water effects (added mass), current load on the jacket, wave load on the jacket, and wind load on the tower.
- A more sophisticated implementation of the gravity load
- Pile/soil interaction as a contact problem
- Other elements and mesh optimization

Other earthquake loads may be used to see how different earthquake characteristics affects the permanent tilt accumulation. Using different soil profiles while keeping the same earthquake load could be used to examine how the soil dynamic properties (natural frequencies) in conjunction with the given earthquake load characteristics give raise to different tilt accumulation. The dynamics of the top load is assumed not being of great importance, due to the wind operating in such a low frequency range that it will be nearly static during the short period of earthquake excitation. However, realistic tower top load could be obtained by extracting tower top reaction forces from an OpenFAST analysis with the given wind field conditions. Then, the assumption of the wind dynamics being negligible could be verified. The environmental loads can, as mentioned earlier, be added in Abaqus by using the Abaqus Aqua toolbox. The submerged effect will most likely give a more stable system, as the water added mass effect will lower the rocking behaviour of the OWT. A wave and current load in the same direction as the tower top load, on the other hand, will give rise to more tilt. An unfavorable wave load (in terms of frequency content) may also be more dynamic interesting, as waves operates on higher frequencies than wind, and may have an amplifying effect during the earthquake. A more sophisticated, but easy, implementation of the gravity load could be only applying gravity to the OWT and not the soil. This will not give the initial high shear stress in the soil, also not giving the negative friction at the top of the piles. The methodical implementation in Abaqus would then be to apply *line loads* on the beam elements, corresponding to the self-weight per unit length. The cumbersome part of this is the calculation of the self-weight for the different cross-sections, but such a generic operation could easily be implemented in the python-scripts. In the prolongation of the gravity implementation, further work should also investigate an implementation of the pile/soil interaction as a contact problem. To increase the computational efficiency and accuracy of the soil, the use of other mesh properties and elements could be investigated. One important aspect would be how the element mesh manage to capture the propagating waves.

References

[1] Wind Europe. *Offshore wind in Europe – key trends and statistics 2020*;. Available from: https://windeurope.org/intelligence-platform/product/offshore-wind-in-europe-key-trends-and-statistics-2020/.

[2] GWEC. *Global Offshore Wind report 2020*;. Available from: https://gwec.net/market-intelligence/resources/.

[3] IEA. *Moment of inertia #8*;. Rotor blades moment of inertia calculated. Available from: https://github.com/IEAWindTask37/IEA-10.0-198-RWT/issues/8.

[4] Georgiou I, Gelagoti F, Kourkoulis R, Gazetas G. *Seismic response of a 10MW offshore wind turbine: performance comparison between a monopile and a jacket foundation.* 16th European conference on earthquake engineering 2018. 2018.

[5] General Electric. *GE webpage for the Haliade-X 14MW*;. Available from: https://www.ge.com/renewableenergy/wind-energy/offshore-wind/haliade-x-offshore-turbine.

[6] Bortolotti P, Tarres HC, Dykes K, Merz K, Sethuraman L, Verelst D, et al. *IEA Wind Task 37 on Systems Engineering in Wind Energy – WP2.1 Reference Wind Turbines.* International Energy Agency; 2019. Available from: https://www.nrel.gov/docs/fy19osti/73492.pdf.

[7] Bak C, Zahle F, Bitsche R, Taeseong K, Yde A, Henriksen LC, et al. *Description of the DTU 10 MW Reference Wind Turbine.* Fredriksborgvej 399, 4000 Roskilde, Denmark: DTU Wind Energy; 2013.

[8] von Brostel (Rambøll) T. *Design Report - Reference Jacket.* INNWIND.EU; 2013. Deliverable 4.31. Available from: http://www.innwind.eu/publications/deliverable-reports.

[9] Rambøll. *Interface document for prelimenary jacket design.* INNWIND.EU; 2013. Deliverable 1.21. Available from: http://www.innwind.eu/publications/deliverable-reports.

[10] Kaynia A. *Seismic considerations in design of offshore wind turbines.* Soil Dynamics and Earthquake Engineering. 2019;124:399–407.

[11] NREL. *OpenFAST repository.* GitHub; 2021. Available from: https://github.com/OpenFAST/openfast.

[12] Dassault Systèmes. *Abaqus*; 2021. Available from: https://bit.ly/3qzjGwk.

[13] Chopra AK. *Dynamics of Structures: Theory and Application to Earthquake Engineering.* 4th ed. Upper Saddle River, New Jersy: Prentice Hall; 2012.

[14] Cook RD, Malkus DS, Plesha ME, Witt RJ. *Concepts and Applications of Finite Element Analysis.* 4th ed. United States: John Wiley and Sons, inc.; 2002.

[15] Weisstein EW. *Least Squares Fitting*; 2021. From MathWorld - A Wolfram Web Resource. Available from: https://mathworld.wolfram.com/LeastSquaresFitting.html.

[16] MIT. *Natural frequency extraction*; 2021. Abaqus documentation by MIT. Available from: https://abaqus-docs.mit.edu/2017/English/SIMACAEANLRefMap/simaanl-c-freqextraction.htm.

[17] Hilber HM, Hughes TJR, Taylor RL. *Improved numerical dissipation for time integration algorithms in structural dynamics.* Earthquake Engineering & Structural Dynamics. 1977;5(3):283–292. Available from: https://doi.org/10.1002/eqe.4290050306.

[18] MIT. *Implicit dynamic analysis using direct integration*; 2021. Abaqus documentation by MIT. Available from: https://abaqus-docs.mit.edu/2017/English/SIMACAEANLRefMap/simaanl-c-dynamic.htm.

[19] Mathisen KM. *Lecture 2 - Solution of Nonlinear Equilibrium Equations*; 2020. TKT4197 Nonlinear Finite Element, NTNU.

[20] Mathisen KM. *Lecture 3 - Adaptive solution algorithms*; 2020. TKT4197 Nonlinear Finite Element, NTNU.

[21] Allen MS, Rixen D, van der Seijs M, Tiso P, Abrahamsson T, Mayes RL. *Substructuring in Engineering Dynamics*. vol. 594. Switzerland: Springer; 2020.

[22] NREL. *OpenFAST - Documentation*; 2021. Documentation for installing, using and develop OpenFAST. Available from: https://openfast.readthedocs.io/en/main/.

[23] Craig R, Bampton M. *Coupling of Substructures for Dynamic analysis*. American Institute of Aeronautics and Astronautics, AIAA Journal. 1968;6(7):1313–1319.

[24] Kramer SL. *Geotechnical Earthquake Engineering*. Upper Saddle River, New Jersy: Prentice Hall; 1996.

[25] Chandrasekaran S. *Dynamic Analysis and Desig of Offshore Structures*. 2nd ed. Singapore: Springer; 2018.

[26] Macmurdo J. *XXI. Papers relating to the earthquake which occurred in India in 1819*. The Philosophical Magazine. 1824;63(310):105–119. Available from: https://doi.org/10.1080/14786442408644477.

[27] *PEER Strong Motion Database*; 2021. Available from: https://peer.berkeley.edu/peer-strong-ground-motion-databases.

[28] Li Y, Zhao M, Xu Cs, Du Xl, Li Z. *Earthquake input for finite element analysis of soil-structure interaction on rigid bedrock*. Tunnelling and underground space technology. 2018;79:250–262.

[29] Zhang W, Seylabi E, Esmaeilzadeh E, Taciroglu E. *An ABAQUS toolbox for soil-structure interaction analysis*. Computers and geotechnics. 2019;114(103143).

[30] Mathisen KM. *Lecture 5 - Incremental Relations for Time-independent Elasto-plastic Materials*; 2020. TKT4197 Nonlinear Finite Element, NTNU.

[31] MIT. *Mohr-Coulomb model*; 2021. Abaqus documentation by MIT. Available from: https://abaqus-docs.mit.edu/2017/English/SIMACAETHERefMap/simathe-c-mohrcoulomb.htm.

[32] DNV GL. *Recommended Practice, DNV-RP-C205*. DNV GL; 2010.

[33] Newman JN. *Marine Hydrodynamics*. 40th ed. United States: The MIT Press; 2017.

[34] Larsen CM. *Marine Dynamics*; 2015. Compendium in TMR 4182.

[35] Wind Europe. *Wind Basics*;. Available from: https://windeurope.org/about-wind/wind-basics/.

[36] Tamura Y, Kareem A. *Advanced Structural Wind Engineering*. Japan: Springer; 2013.

[37] Jonkman BJ. *TurbSim user's guide*; 2016. National Renewable Energy Laboratory.

[38] *Python Software Foundation*. Python; 2020. Available from: https://www.python.org/.

[39] Dassault Systèmes. *Abaqus Scripting Reference Guide*; 2014. Available from: https://tinyurl.com/hf3pctae.

[40] MIT. *Abaqus/Aqua analysis*; 2021. Abaqus documentation by MIT. Available from: https://abaqus-docs.mit.edu/2017/English/SIMACAEANLRefMap/simaanl-c-aqua.htm.

[41] MIT. *General multi-point constraints*; 2021. Abaqus documentation by MIT. Available from: https://abaqus-docs.mit.edu/2017/English/SIMACAECSTRefMap/simacst-c-mpc.htm.

[42] MIT. *Mesh tie constraint*; 2021. Abaqus documentation by MIT. Available from: https://abaqus-docs.mit.edu/2017/English/SIMACAECSTRefMap/simacst-c-tiedconstraint.htm.

[43] Mucciacciaro M, Sica S. *Nonlinear soil and pile behaviour on kinematic bending response of flexible piles*. Soil Dynamics and Earthquake Engineering. 2018;107:195–213. Available from: https://doi.org/10.1016/j.soildyn.2017.12.025.

[44] MIT. *Using substructures*; 2021. Abaqus documentation by MIT. Available from: https://abaqus-docs.mit.edu/2017/English/SIMACAEANLRefMap/simaanl-c-superelements.htm.

[45] NREL. *FAST forum*; 2021. 15 year old forum for FAST which still is in use today. Available from: https://wind.nrel.gov/forum/wind/viewforum.php?f=4&sid=f53aea05293a7ffb0d0fa40f1ad52caa.

[46] *Pull-request #739*;. Logic error in glue code for StC loads on SD with no HD. Available from: https://github.com/OpenFAST/openfast/pull/739.

[47] NREL. *OpenFAST - Documentation, development version*; 2021. Documentation for installing, using and develop OpenFAST. Available from: https://openfast.readthedocs.io/en/dev/.

[48] NREL. *ROSCO. Version 2.2.0*. GitHub; 2021. Available from: https://github.com/NREL/rosco.

[49] Bir GS. *User's guide to BModes*; 2007. National Renewable Energy Laboratory.

[50] IEA. *IEA Wind Task 37 GitHub*; 2021. GitHub repository for distributing the OpenFAST model. Available from: https://github.com/IEAWindTask37/IEA-10.0-198-RWT.

[51] NREL. *ModeShapePolyFitting.xls*;.

[52] NREL. *OpenFAST Linearization step-by-step*; 2021. Step-by-step desciption of how to perform linearization in OpenFAST. Available from: https://ebranlard-openfast.readthedocs.io/en/doc/source/working.html#performing-linearization-analyses.

Appendix

A Supplementary model description

A.1 Soil profile

Table A.1: Soil profile used in the Abaqus model

Layer	Thickness [m]	Top z-coord. [m]	Bottom z-coord. [m]	Mass density [kg/m^3]	Young's modulus [MPa]	Poisson's ratio [-]	Choesion stress [kPa]
1	2	-50	-52	1936	34.4	0.3	1.8
2	3.5	-52	-55.5	1936	66.6	0.3	6.8
3	3.5	-55.5	-59	1936	92.6	0.3	13.1
4	1	-59	-60	1936	106	0.3	17.1
5	5	-60	-65	1936	122	0.3	22.5
6	5	-65	-70	2038	192	0.3	32.0
7	2.5	-70	-72.5	2089	241	0.3	39.6
8	6.5	-72.5	-79	2140	300	0.3	49.4
9	5	-79	-84	2140	336	0.3	62.1
10	4	-84	-88	2140	362	0.3	72.0
11	2	-88	-90	2140	378	0.3	78.6
12	10	-90	-100	2140	409	0.3	91.8
13	10	-100	-110	2140	455	0.3	113.8
14	10	-110	-120	2140	497	0.3	135.8
15	10	-120	-130	2140	536	0.3	157.8
16	10	-130	-140	2140	572	0.3	179.8

A.2 RNA mass and inertia

Table A.2: Equivalent point mass properties of RNA components of the IEA 10-MW offshore wind turbine. The mass moment inertias are with respect to the global axis-directions at the tower top, see Figure A.1 for an illustration. Blade inertia calculated by [3]

Component	Mass [kg]	I_{xx} [kg m^2]	I_{yy} [kg m^2]	I_{zz} [kg m^2]
Yaw bearing	93 457	1.65E+05	1.66E+05	2.83E+05
Nacelle turret and nose	109 450	1.41E+06	2.13E+06	1.05E+06
Inner generator stator	187 673	5.94E+06	1.02E+07	7.96E+06
Outer generator rotor	169 606	5.83E+06	9.47E+06	7.46E+06
Shaft	78 894	9.96E+05	3.28E+06	2.36E+06
Hub	81 707	1.68E+06	9.89E+06	8.69E+06
Three blades	145 768	2.24E+08	1.07E+08	8.40E+07
Total	866 555	2.40E+08	1.42E+08	1.12E+08

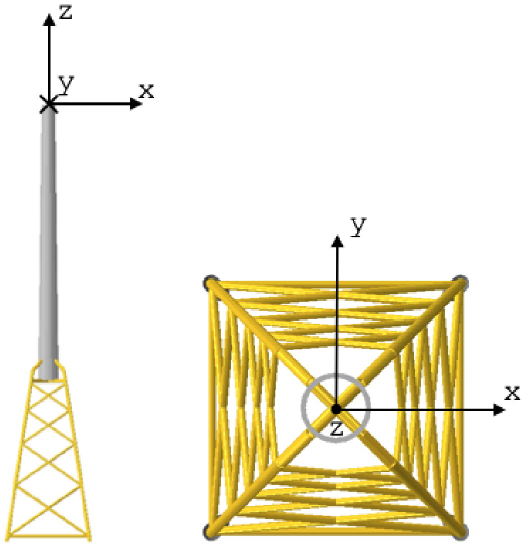


Figure A.1: Shows the axis direction and origin of the RNA mass moment inertias.

A.3 Reference jacket design drawings interpretation

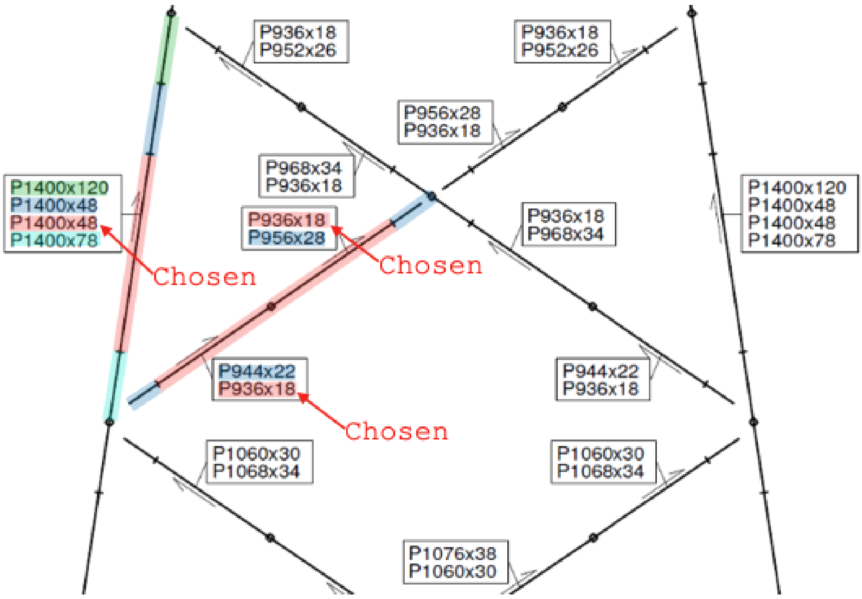


Figure A.2: Snapshot of Reference jacket design drawing. Color marking showing which cross section properties are chosen for the FE modelling. Naming convention: *Section type* x *Outer diameter* x *Wall-thickness*. *P* denotes pipe section.

B Additional theory

B.1 Example: Rigid body assemblage

Consider the wind turbine system presented in Figure B.1. Now making an idealization of the system, treating the tower as an uniform cross section with height H, distributed mass per unit length m_T and bending stiffness EI. Let the blades have an infinitely stiff uniform cross section with length L and mass per unit length m. With the given geometry including a hub overhang equal to L_0 and an infinitely stiff shaft, the system can be presented in the x-z plane as shown in Figure B.2. The RNA, i.e., the blades and the shaft, will now be treated as a rigid body, and the system might be described by two DOFs; displacement, u, and rotation, θ, of the tower top.

For this linear system, the system matrices, $\mathbf{M}$ and $\mathbf{K}$, can be determined by superpositioning of the rigid body (RB) and the tower contributions;

$$\mathbf{M} = \mathbf{M}_{\text{tower}} + \mathbf{M}_{\text{RB}}$$
$$\mathbf{K} = \mathbf{K}_{\text{tower}} + \mathbf{K}_{\text{RB}}$$

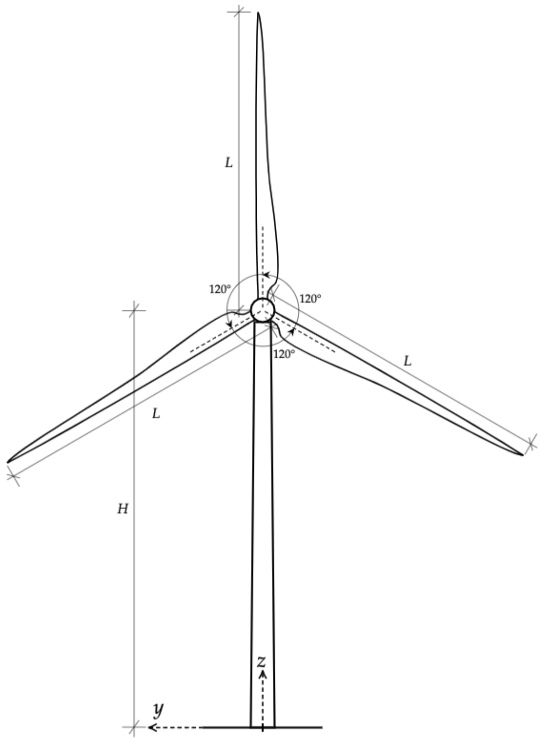


Figure B.1: Example wind turbine system.

Assuming tower mode shapes corresponding to a cubic interpolated Euler Bernoulli beam and a consistent mass approach give the following matrices for the tower contribution:

$$\mathbf{M}_{\text{tower}} = \frac{m_T H}{420} \cdot \begin{bmatrix} 156 & -22H \\ -22H & 4H^2 \end{bmatrix}$$
$$\mathbf{K}_{\text{tower}} = \frac{2EI}{H^3} \cdot \begin{bmatrix} 6 & -3H \\ -3H & 2H^2 \end{bmatrix}$$

A direct approach, using D'Alembert's principle and virtual displacements, gives the rigid body contributions (assuming small displacements and rotations). The distributed inertia resultants and mass moment of inertia around the center of mass for each part of the rigid body are visualized in Figure B.2. Note that the rigid body do not apply any stiffness to the system, and the stiffness contribution is therefore zero, $\mathbf{K}_{\text{RB}} = \mathbf{0}$. The equilibria with u as the active DOF now determine the first column of the mass matrix:

$$\sum F_x = M_{11}\ddot{u} - m_0 L_0 \ddot{u} - 3mL\ddot{u} = 0$$
$$\sum M_O = M_{21}\ddot{u} - mL\ddot{u} \cdot L/2 + 2mL\ddot{u} \cdot L/4 = 0$$
$$\Rightarrow M_{11} = m_0 L_0 + 3mL, \quad M_{21} = 0$$

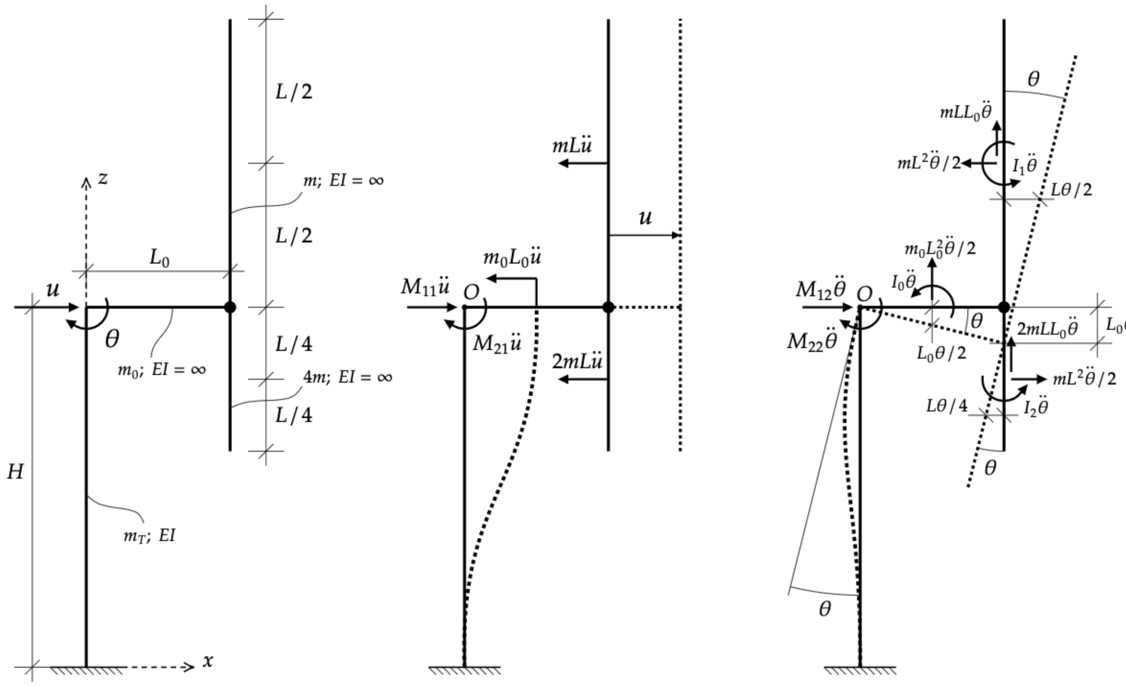


Figure B.2: Left: System idealization of the wind turbine presented in figure B.2. Note that the proportions are not correct, but are made for visualization purposes. Middle: FBD of RB contribution when u is the only active DOF. Right: FBD of RB contribution when θ is the only active DOF.

θ as the active DOF then gives the second column:

$$
\begin{aligned}
\sum F_x =& M_{12}\ddot{\theta} + mL^2\ddot{\theta}/2 - mL^2\ddot{\theta}/2 = 0 \\
\sum M_O =& M_{22}\ddot{\theta} - (I_0 + \frac{mL_0^3}{4})\ddot{\theta} \\
& - (I_1 + \frac{mL^3}{4} + mLL_0^2)\ddot{\theta} \\
& - (I_2 + \frac{mL^3}{8} + mLL_0^2)\ddot{\theta} = 0
\end{aligned}
$$

The mass moment of inertia about the center of mass I_{CM} for a bar with evenly distributed mass m and length L is

$$
I_{CM} = \frac{mL^3}{12}
$$

Applying this to the inertias of the different rigid body parts and rearranging then gives:

$$
\begin{aligned}
M_{12} =& 0 \\
M_{22} =& \frac{m_0L_0^3}{3} + \frac{mL^3}{2} + 3mLL_0^2
\end{aligned}
$$

On matrix form:

$$
\begin{aligned}
\mathbf{M} =& \frac{m_TH}{420} \cdot \begin{bmatrix} 156 & -22H \\ -22H & 4H^2 \end{bmatrix} + \begin{bmatrix} m_0L_0 + 3mL & 0 \\ 0 & \frac{m_0L_0^3}{3} + \frac{mL^3}{2} + 3mLL_0^2 \end{bmatrix} \\
\mathbf{K} =& \frac{2EI}{H^3} \cdot \begin{bmatrix} 6 & -3H \\ -3H & 2H^2 \end{bmatrix}
\end{aligned}
$$

Note that the rigid body mass matrix contribution are zero for the off-diagonal terms. This is due to the center of mass of the rigid body coincide with the displacement axis of u. Thus, horizontal motion of the tower top will not give any inertia moment at the tower top due to the rigid body inertia resultant force.

If examining the rigid body contribution further, it can be seen that the first diagonal term corresponds to the total rigid body mass as a point mass at the tower top. The last member of the second diagonal term indicates an parallel axis shift according to the Parallel axis theorem. The Parallel axis theorem states that the inertia about an axis parallel to the calculated inertia axis can be found by adding the body mass times the squared perpendicular distance between the axes;

$$I_y = I_{y'} + M \cdot d^2$$

where y denotes the new axis, y' the original axis, M the total body mass and d the perpendicular distance between the axes. It then turns out the second diagonal term of the rigid body mass matrix equals the inertia of each rigid body part around the tower top;

$$\text{Shaft: } I_{0,O} = \frac{m_0 L_0^3}{12} + m_0 L_0 \cdot \left(\frac{L_0}{2}\right)^2 = \frac{m_0 L_0^3}{3}$$

$$\text{Upper blade: } I_{1,O} = \frac{mL^3}{12} + mL \cdot [L_0^2 + \left(\frac{L}{2}\right)^2] = \frac{mL^3}{3} + mLL_0^2$$

$$\text{Lower blades: } I_{2,O} = \frac{4m(L/2)^3}{12} + 4m\frac{L}{2} \cdot [L_0^2 + \left(\frac{L}{4}\right)^2] = \frac{mL^3}{6} + 2mLL_0^2$$

Adding up all the contribution gives the total inertia of

$$I_O = I_{0,O} + I_{1,O} + I_{2,O} = \frac{m_0 L_0^3}{3} + \frac{mL^3}{2} + 3mLL_0^2$$

which is the same as the second diagonal term derived by the direct approach.

This shows that such a rigid body can be added to the rest of the structure as a point mass and its inertia, and the geometry do not need to be fully modelled in software as Abaqus. However, potential off-diagonal terms of the mass matrix will not be included, and the rigid body should therefore have a satisfying symmetry when using this approach.

B.2 Derivation of the LSF damping estimation method

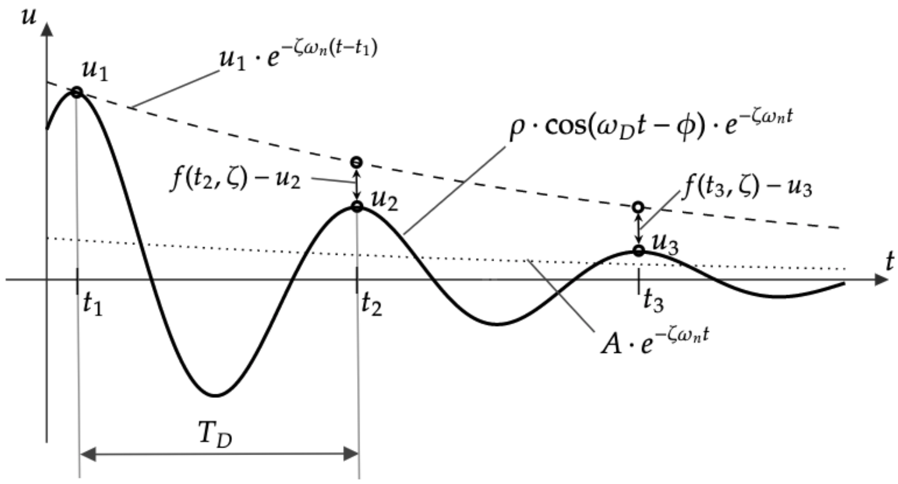


Figure B.3: Procedure illustration. The dotted line illustrates the model equation curve before fitting any parameters. Dashed line illustrates the model equation curve after fitting the first value.

This section shows the derivation of the damping estimation method named *Least Squares Fitting damping estimation method*. This particular method is not anchored in the theory, but is developed during this project based on the well known least squares fitting (LSF) data regression method [15].

Assume a time series behaving like the damped free decay response as described in equation (2.1.8), and visualized in figure B.3, and that the peak values and the corresponding time instances of the time series are known and ordered with increasing time. The peak values and time instances are denoted, respectively, u_i and t_i for $i = 1, 2, \ldots, n$. The amplitude of such response diminishes according to the non-cosine terms in equation (2.1.8) and its given damping ratio. This terms are represented by the envelope curves shown in figure 2.2, and is suited for estimating the damping ratio. The model equation, or curve to be fitted, is therefore

$$f(t, \zeta) = A \cdot e^{-\zeta\omega_n t}, \qquad \zeta \in \langle 0, 1 \rangle$$

where A represents the $t = 0$ intersection amplitude, ζ the damping ratio and ω_n the natural frequency of the response. This equation has actually only two unknowns in this case; A and ζ. ω_n is, in theory, a function of ζ due to the fact that T_D are known from t_i, and due to the relation

$$T_D = \frac{2\pi}{\omega_D} = \frac{2\pi}{\omega_n\sqrt{1-\zeta^2}} \Rightarrow \omega_n = \omega_n(\zeta) \tag{B.2.1}$$

The amplitude A is easily decided by using the first peak value and time instance:

$$\begin{aligned}
&f(t_1, \zeta) = u_1 \\
\Rightarrow &A \cdot e^{-\zeta\omega_n(\zeta)t_1} = u_1 \\
\Rightarrow &A = \frac{u_1}{e^{-\zeta\omega_n(\zeta)t_1}} \\
\Rightarrow &f(t, \zeta) = u_1 \cdot e^{-\zeta\omega_n(\zeta)(t-t_1)}
\end{aligned} \tag{B.2.2}$$

This step is visualized in figure B.3 where the model equation goes from the dotted line to the dashed line.

Now introducing the *sum of squared vertical deviations* (SSVD) function, $g(\zeta)$. The vertical deviations $f(t_i, \zeta) - u_i$ is visualized in figure B.3. The sum of squared vertical deviations should be minimized with respect to ζ to fit the model curve to the response time series. The function is defined as

$$g(\zeta) = \sum_{i=1}^{n} (f(t_i, \zeta) - u_i)^2 \tag{B.2.3}$$

Note that this procedure does not minimize the actual deviations between the model equation curve and the peaks (which would be measured perpendicular). In addition, although the unsquared sum of distances might seem a more appropriate quantity to minimize, use of the absolute values results in discontinuous derivatives which cannot be treated analytically. The square deviations from each point are therefore summed, and the resulting residual is then minimized to find the best fit. This procedure results in outlaying points being given disproportionately large weighting, which, however, should not be a problem for a good response time series. The condition for $g(\zeta)$ to be minimized is

$$\frac{\partial g(\zeta)}{\partial \zeta} = 0 \tag{B.2.4}$$

As the model function already is fitted to the first peak;

$$\begin{aligned} &f(t_1, \zeta) - u_1 = 0 \\ \Rightarrow &g(\zeta) = \sum_{i=1}^{n} (f(t_i, \zeta) - u_i)^2 = \sum_{i=2}^{n} (f(t_i, \zeta) - u_i)^2 \end{aligned}$$

Now substitute equation (B.2.2) and (B.2.1) into the SSVD function:

$$\Rightarrow g(\zeta) = \sum_{i=2}^{n} (u_1 \cdot \exp\left[-\frac{\zeta}{\sqrt{1-\zeta^2}} \frac{2\pi}{T_D}(t_i - t_1)\right] - u_i)^2 \tag{B.2.5}$$

exp denotes exponential. For convenience, let

$$\delta = \frac{\zeta}{\sqrt{1-\zeta^2}} \qquad \text{and} \qquad x_i = \frac{2\pi}{T_D}(t_i - t_1)$$

$$\Rightarrow g(\zeta) = \sum_{i=2}^{n} (u_1 \cdot e^{-\delta \cdot x_i} - u_i)^2 \tag{B.2.6}$$

Let $d\delta/d\zeta = \delta'$ and differentiate :

$$\begin{aligned} \frac{\partial g(\zeta)}{\partial \zeta} &= \sum_{i=2}^{n} 2 \cdot (u_1 \cdot e^{-\delta \cdot x_i} - u_i) \cdot \frac{\partial}{\partial \zeta}(u_1 \cdot e^{-\delta \cdot x_i} - u_i) \\ &= 2 \cdot \sum_{i=2}^{n} u_1 u_i x_i \delta' e^{-\delta x_i} - u_1^2 x_i \delta' e^{-2\delta x_i} \end{aligned}$$

Set equal zero and start solving for ζ:

$$\begin{aligned} \frac{\partial g(\zeta)}{\partial \zeta} &= 0 \\ \Rightarrow u_1 \delta' \sum_{i=2}^{n} u_i x_i e^{-\delta x_i} &= u_1^2 \delta' \sum_{i=2}^{n} x_i e^{-2\delta x_i} \\ \Rightarrow \sum_{i=2}^{n} u_i x_i e^{-\delta x_i} &= u_1 \sum_{i=2}^{n} x_i e^{-2\delta x_i} \end{aligned}$$

Apply the natural logarithm to each side:

$$\Rightarrow \sum_{i=2}^{n} \ln(u_i x_u) - \delta x_i = \sum_{i=2}^{n} \ln(u_1 x_i) - 2\delta x_i$$

$$\Rightarrow \ln(\prod_{i=2}^{n} u_i x_i) - \delta \sum_{i=2}^{n} x_i = \ln(\prod_{i=2}^{n} u_1 x_i) - 2\delta \sum_{i=2}^{n} x_i$$

$$\Rightarrow \delta = \ln\left(\frac{\prod_{i=2}^{n} u_1 x_i}{\prod_{i=2}^{n} u_i x_i}\right) \cdot \frac{1}{\sum_{i=2}^{n} x_i} \tag{B.2.7}$$

Now consider the logarithmic term:

$$\begin{aligned}\frac{\prod_{i=2}^{n} u_1 x_i}{\prod_{i=2}^{n} u_i x_i} =& \frac{u_1 x_2 \cdot u_1 x_3 \cdot ... \cdot u_1 x_n}{u_2 x_2 \cdot u_3 x_3 \cdot ... \cdot u_n x_n} \\ =& \frac{u_1 \cdot u_1 \cdot ... \cdot u_1}{u2 \cdot u_3 \cdot ... \cdot u_n} = \frac{u_1^{n-1}}{\prod_{i=2}^{n} u_i}\end{aligned}$$

Now consider the x_i summation and utilize the $\sum_{k=1}^{n} k = n(n+1)/2$ relation:

$$\sum_{i=2}^{n} x_i = \sum_{i=2}^{n} \frac{2\pi}{T_D}(t_i - t_1)$$

$$t_i = t_1 + (i-1) \cdot T_D, \quad i = 2,\ 3,\ ...\ n$$

$$\begin{aligned}\Rightarrow \sum_{i=2}^{n} x_i = \frac{2\pi}{T_D}&[(t_1 + T_D - t_1) + (t_1 + 2T_D - t_1) + ... + (t_1 + (n-1)T_D - t_1)] \\ &= 2\pi(1 + 2 + ... + (n-1)) \\ &= 2\pi \sum_{k=1}^{n-1} k \\ &= 2\pi \frac{n(n-1)}{2} \\ &= \pi n(n-1)\end{aligned}$$

Insert the expressions back into equation (B.2.7):

$$\Rightarrow \delta = \ln\left(\frac{u_1^{n-1}}{\prod_{i=2}^{n} u_i}\right) \cdot \frac{1}{\pi n(n-1)} \tag{B.2.8}$$

$$\delta = \frac{\zeta}{\sqrt{1-\zeta^2}}$$

$$\Rightarrow \zeta = \frac{\delta}{\sqrt{1+\delta^2}} \tag{B.2.9}$$

Equation (B.2.9) then gives the estimated (optimized) damping ratio.

C Abaqus python-scripts

This section presents the python-scripts along with ".csv"-files holding the input parameters used to build the Abaqus model. The following scripts are included:

1. `tower.py`
   - Building the tower part (including the RNA)
   - Utilizes `towerData.csv`
2. `transitionPiece.py`
   - Building the transition piece part
   - Utilizes `transitionPieceData.csv`
3. `jacket.py`
   - Building the jacket part
   - Utilizes `jacketData.csv`
4. `piles.py`
   - Building the piles as one part
   - Utilizes `pileData.csv`
5. `soil.py`
   - Building the soil part
   - Utilizes `soilData.csv`
6. `assemble_full_model.py`
   - Assembles the integrated model with the given part connections and boundary conditions
   - Utilizes all the above-mentioned scripts
7. `soilSlice.py`
   - Builds a soil slice part
   - Utilizes `soil.csv`
8. `assemble_OWT_only.py`
   - Assembles the OWT (not including piles) clamped at base
   - Utilizes the relevant part building script
9. `assemble_soil_and_piles_stiffness_analysis.py`
   - Assembles soil and piles, with boundary conditions for extracting base-attachment stiffness used by the OpenFAST model
   - Utilizes the relevant part building script
10. `ODB_get_stiffness.py`
    - After running a static analysis of the setup from script 9, this script makes the pile-head stiffness matrix (used in OpenFAST) from the ODB-file
    - Saves the result to `ODB_stiffness_output.txt`
11. `assemble_soil_and_piles.py`
    - Assembles soil and piles, with boundary conditions for extracting kinematic interaction motion used by the OpenFAST model. Massless jacket need to be added separately.

- Utilizes the relevant part building script

12. assemble_soil_slice.py
    - Assembles the soil slice with the given boundary conditions
    - Utilizes the soilSlice.py

13. substructuring.py
    - Assembles the soil and part ready to generate a substructure to use along with the OWT.
    - Utilizes the relevant part building script

The scripts are added in the following listings. The csv-files follows after the scripts.

Listing 1: tower.py

```
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
import numpy as np


# Input
data = open('towerData.csv','r')
lines = data.readlines()
towerBotLevel = float(lines[0])
towerTopLevel = float(lines[1])
botRadius = float(lines[2])
topRadius = float(lines[3])
botWallThickness = float(lines[4])
topWallThickness = float(lines[5])
elements = int(lines[6])
matN = lines[7].strip()
E = float(lines[8])
nu = float(lines[9])
rho = float(lines[10])
alpha = float(lines[11])
beta = float(lines[12])
RNAmass = float(lines[13])
Ixx = float(lines[14])
Iyy = float(lines[15])
Izz = float(lines[16])
data.close()

mN = 'Model-1'
pN = 'Tower'
MODEL = mdb.models[mN] # Pass by reference


# Make node coordinates and element wallThickness
a1 = (topRadius - botRadius)/(towerTopLevel-towerBotLevel)
b1 = botRadius - a1 * towerBotLevel
a2 = (topWallThickness - botWallThickness)/(towerTopLevel-towerBotLevel)
b2 = botWallThickness - a2 * towerBotLevel
nodes = np.linspace(towerBotLevel,towerTopLevel,elements+1)
middleOfElements = np.linspace(towerBotLevel,towerTopLevel,elements*2+1)
middleOfElements = middleOfElements[1:-1:2]
profileRadius = a1*middleOfElements + b1
wallThickness = a2*middleOfElements + b2


# Make material
MODEL.Material(
        name = matN)
MODEL.materials[matN].Density(
        table = ((rho, ), ))
MODEL.materials[matN].Elastic(
        table = ((E, nu), ))
MODEL.materials[matN].Damping(
        alpha = alpha,
```

```
        beta = beta)


# Make the tower
MODEL.Part( # Making part
        dimensionality = THREE_D,
        name = pN,
        type = DEFORMABLE_BODY)
PART = MODEL.parts[pN] # Pass by reference
for i in range(0,elements):
        PART.WirePolyLine( # Wire feature
                mergeType = MERGE,
                meshable = ON,
                points = ((0,0,nodes[i]), (0,0,nodes[i+1])))
        PART.Set( # Make set of the created wire
                name = 'Beam_' + str(i+1),
                edges = PART.edges.findAt(((0,0,middleOfElements[i]), )))
        MODEL.PipeProfile( # Pipe profile
                name = 'Pipe_profile_' + str(i+1),
                r = profileRadius[i],
                t = wallThickness[i])
        MODEL.BeamSection( # Beam section
                consistentMassMatrix = False,
                integration = DURING_ANALYSIS,
                material = matN,
                name = 'Tower_section_' + str(i+1),
                poissonRatio = nu,
                profile = 'Pipe_profile_' + str(i+1),
                temperatureVar = LINEAR)
        PART.SectionAssignment( # Assign section
                offset = 0.0,
                offsetField = '',
                offsetType = MIDDLE_SURFACE,
                region = PART.sets['Beam_' + str(i+1)],
                sectionName = 'Tower_section_' + str(i+1),
                thicknessAssignment = FROM_SECTION)
        PART.assignBeamSectionOrientation( # Assign beam section orientation
                method = N1_COSINES,
                n1 = (0.0, 1.0, 0.0),
                region = PART.sets['Beam_' + str(i+1)])
        PART.seedEdgeByNumber( # Seed beam
                constraint = FINER,
                edges = PART.sets['Beam_' + str(i+1)].edges,
                number = 1)
        PART.setElementType( # Set element type
                elemTypes = (ElemType(
                                                elemCode=B31,
                                                elemLibrary=STANDARD), ),
                regions = PART.sets['Beam_' + str(i+1)])
PART.generateMesh()


# Make some node sets:
Bottom_node = PART.nodes.getByBoundingSphere(center = (0.0, 0.0, nodes[0]), radius = 0)
Top_node = PART.nodes.getByBoundingSphere(center = (0.0, 0.0, nodes[-1]), radius = 0)
        # Syntax for getting a specific node number:
        # PART.Set(name = 'Set_name', nodes = MeshNodeArray([PART.nodes[5], ]))
PART.Set(
        name = 'Top_node',
        nodes = Top_node)
PART.Set(
        name = 'Bottom_node',
        nodes = Bottom_node)


# Add RNA
PART.engineeringFeatures.PointMassInertia(
        alpha = alpha,
        composite = 0.0,
        i11 = Ixx,
        i22 = Iyy,
        i33 = Izz,
        mass = RNAmass,
        name  = 'RNA',
        region = PART.sets['Top_node'])
```

Listing 2: `transitionPiece.py`

```
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
```

```
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

# Load data
data = open('transitionPieceData.csv','r')
lines = data.readlines()
nodeData = []
beamData = []
sectionData = []
for i in range(1,15):
        temp = [str(j) for j in lines[i].strip().split(',')]
        nodeData.append(temp)
for i in range(16,33):
        temp = [str(j) for j in lines[i].strip().split(',')]
        beamData.append(temp)
for i in range(34,39):
        temp = [str(j) for j in lines[i].strip().split(',')]
        sectionData.append(temp)
matN = lines[40].strip()
E = float(lines[41])
nu = float(lines[42])
rho = float(lines[43])
alpha = float(lines[44])
beta = float(lines[45])
data.close()

mN = 'Model-1'
pN = 'Transition_piece'
MODEL = mdb.models[mN]

# Make material
MODEL.Material(
        name = matN)
MODEL.materials[matN].Density(
        table = ((rho, ), ))
MODEL.materials[matN].Elastic(
        table = ((E, nu), ))
MODEL.materials[matN].Damping(
        alpha = alpha,
        beta = beta)


# Make the transition piece
mdb.models[mN].Part(
        dimensionality = THREE_D,
        name = pN,
        type = DEFORMABLE_BODY)
PART = MODEL.parts[pN]
beamSeeding = {}
for i in range(len(sectionData)): # Go through all the section types
        MODEL.PipeProfile( # Make profiles
                name = str(sectionData[i][0]) + '_profile',
                r = float(sectionData[i][1]),
                t = float(sectionData[i][2]))
        MODEL.BeamSection( # Make beam secitons
                consistentMassMatrix = False,
                integration = DURING_ANALYSIS,
                material = matN,
                name = str(sectionData[i][0]),
                poissonRatio = nu,
                profile = str(sectionData[i][0]) + '_profile',
                temperatureVar = LINEAR)
        beamSeeding[str(sectionData[i][0])] = int(sectionData[i][3]) # Making this dictionary for
             simplicity
for i in range(len(beamData)): # Go through and create all the beams
        startIx = int(beamData[i][1])-1
        endIx = int(beamData[i][2])-1
        xs = float(nodeData[startIx][1])
        ys = float(nodeData[startIx][2])
        zs = float(nodeData[startIx][3])
        xe = float(nodeData[endIx][1])
        ye = float(nodeData[endIx][2])
        ze = float(nodeData[endIx][3])
        PART.WirePolyLine( # Make wire features
                mergeType = MERGE,
                meshable = ON,
                points = ((xs,ys,zs), (xe,ye,ze)))
        # Now need to identify the index of the wire just created to make a set of its edge
        # The index is the index in the PART.edges array.
        # This is needed because, for some reason, ABAQUS do not put the newliest created
        # wire edge at the end of the PART.edges array.
        ix = 0 # Starting index to check
        findStr = 'Wire-' + str(i+1)
```

```
        check = (findStr == PART.edges[ix].featureName) # TRUE if the index corresponds to the
             current ix value
        while(check == False):
                ix += 1
                check = (findStr == PART.edges[ix].featureName)
        pointOnEdge = PART.edges[ix].pointOn
        PART.Set( # Make set of the wire just created
                name = str(beamData[i][3]) + '_set',
                edges = PART.edges.findAt(pointOnEdge))
        PART.SectionAssignment( # Assign section
                offset = 0.0,
                offsetField = '',
                offsetType = MIDDLE_SURFACE,
                region = PART.sets[str(beamData[i][3]) + '_set'],
                sectionName = str(beamData[i][4]),
                thicknessAssignment = FROM_SECTION)
        if(str(beamData[i][4]) == 'TP_section_tower_piece'): # This beam is vertical, and need
             another n1 orientation
                n1 = (0.0, 1.0, 0.0)
        else:
                n1 = (0.0, 0.0, -1.0)
        PART.assignBeamSectionOrientation( # Assign beam section orientation
                method = N1_COSINES,
                n1 = n1,
                region = PART.sets[str(beamData[i][3]) + '_set'])
        PART.seedEdgeByNumber( # Seed beam
                constraint = FINER,
                edges = PART.sets[str(beamData[i][3]) + '_set'].edges,
                number = beamSeeding[str(beamData[i][4])])
        PART.setElementType(
                elemTypes = (ElemType(
                                                elemCode=B31,
                                                elemLibrary=STANDARD), ),
                regions = PART.sets[str(beamData[i][3]) + '_set'])
PART.generateMesh()


# Add some extra sets
PART.Set(
        name = 'Lower_connection_set',
        vertices = PART.vertices.findAt(
                ((float(nodeData[3-1][1]),float(nodeData[3-1][2]),float(nodeData[3-1][3])), ),
                ((float(nodeData[6-1][1]),float(nodeData[6-1][2]),float(nodeData[6-1][3])), ),
                ((float(nodeData[9-1][1]),float(nodeData[9-1][2]),float(nodeData[9-1][3])), ),
                ((float(nodeData[12-1][1]),float(nodeData[12-1][2]),float(nodeData[12-1][3])), )))
PART.Set(
        name = 'Upper_connection_set',
        vertices = PART.vertices.findAt(
                ((float(nodeData[2-1][1]),float(nodeData[2-1][2]),float(nodeData[2-1][3])), )))

# Could have used PART.getFeatureEdges(...) to identify the Wire edges
```

Listing 3: `jacket.py`

```
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *


# Load data
data = open('jacketData.csv','r')
lines = data.readlines()
nodeData = []
beamData = []
sectionData = []
for i in range(1,49):
        temp = [str(j) for j in lines[i].strip().split(',')]
        nodeData.append(temp)
for i in range(50,146):
        temp = [str(j) for j in lines[i].strip().split(',')]
        beamData.append(temp)
for i in range(147,159):
        temp = [str(j) for j in lines[i].strip().split(',')]
        sectionData.append(temp)
matN = lines[160].strip()
E = float(lines[161])
```

```
nu = float(lines[162])
rho = float(lines[163])
alpha = float(lines[164])
beta = float(lines[165])
data.close()

mN = 'Model-1'
pN = 'Jacket'
MODEL = mdb.models[mN] # Pass by reference

# Make material
MODEL.Material(
        name = matN)
MODEL.materials[matN].Density(
        table = ((rho, ), ))
MODEL.materials[matN].Elastic(
        table = ((E, nu), ))
MODEL.materials[matN].Damping(
        alpha = alpha,
        beta = beta)


# Make the jacekt
mdb.models[mN].Part(
        dimensionality = THREE_D,
        name = pN,
        type = DEFORMABLE_BODY)
PART = MODEL.parts[pN]
beamSeeding = {}
for i in range(len(sectionData)): # Go through all the section types
        MODEL.PipeProfile( # Make profiles
                name = str(sectionData[i][0]) + '_profile',
                r = float(sectionData[i][1]),
                t = float(sectionData[i][2]))
        MODEL.BeamSection( # Make beam secitons
                consistentMassMatrix = False,
                integration = DURING_ANALYSIS,
                material = matN,
                name = str(sectionData[i][0]),
                poissonRatio = nu,
                profile = str(sectionData[i][0]) + '_profile',
                temperatureVar = LINEAR)
        beamSeeding[str(sectionData[i][0])] = int(sectionData[i][3]) # Making this dictionary for
             simplicity
for i in range(len(beamData)): # Go through and create all the beams
        startIx = int(beamData[i][1])-1
        endIx = int(beamData[i][2])-1
        xs = float(nodeData[startIx][1])
        ys = float(nodeData[startIx][2])
        zs = float(nodeData[startIx][3])
        xe = float(nodeData[endIx][1])
        ye = float(nodeData[endIx][2])
        ze = float(nodeData[endIx][3])
        PART.WirePolyLine( # Make wire features
                mergeType = MERGE,
                meshable = ON,
                points = ((xs,ys,zs), (xe,ye,ze)))
        # Now need to identify the index of the wire just created to make a set of its edge
        # The index is the index in the PART.edges array.
        # This is needed because, for some reason, ABAQUS do not put the newliest created
        # wire edge at the end of the PART.edges array.
        ix = 0 # Starting index to check
        findStr = 'Wire-' + str(i+1)
        check = (findStr == PART.edges[ix].featureName) # TRUE if the index corresponds to the
             current ix value
        while(check == False):
                ix += 1
                check = (findStr == PART.edges[ix].featureName)
        pointOnEdge = PART.edges[ix].pointOn
        PART.Set( # Make set of the wire just created
                name = str(beamData[i][3]) + '_set',
                edges = PART.edges.findAt(pointOnEdge))
        PART.SectionAssignment( # Assign section
                offset = 0.0,
                offsetField = '',
                offsetType = MIDDLE_SURFACE,
                region = PART.sets[str(beamData[i][3]) + '_set'],
                sectionName = str(beamData[i][4]),
                thicknessAssignment = FROM_SECTION)
        if(str(beamData[i][4]) == 'Jacket_section_legs_level_B'): # This legs is vertical, and need
             another n1 orientation
                n1 = (0.0, 1.0, 0.0)
        else:
                n1 = (0.0, 0.0, -1.0)
        PART.assignBeamSectionOrientation( # Assign beam section orientation
                method = N1_COSINES,
```

```
                n1 = n1,
                region = PART.sets[str(beamData[i][3]) + '_set'])
        PART.seedEdgeByNumber( # Seed beam
                constraint = FINER,
                edges = PART.sets[str(beamData[i][3]) + '_set'].edges,
                number = beamSeeding[str(beamData[i][4])])
        PART.setElementType(
                elemTypes = (ElemType(
                                                elemCode=B31,
                                                elemLibrary=STANDARD), ),
                regions = PART.sets[str(beamData[i][3]) + '_set'])
PART.generateMesh()


# Add some extra sets
PART.Set(
        name = 'Lower_connection_set',
        vertices = PART.vertices.findAt(
                ((float(nodeData[1-1][1]),float(nodeData[1-1][2]),float(nodeData[1-1][3])), ),
                ((float(nodeData[9-1][1]),float(nodeData[9-1][2]),float(nodeData[9-1][3])), ),
                ((float(nodeData[17-1][1]),float(nodeData[17-1][2]),float(nodeData[17-1][3])), ),
                ((float(nodeData[25-1][1]),float(nodeData[25-1][2]),float(nodeData[25-1][3])), )))
PART.Set(
        name = 'Upper_connection_set',
        vertices = PART.vertices.findAt(
                ((float(nodeData[8-1][1]),float(nodeData[8-1][2]),float(nodeData[8-1][3])), ),
                ((float(nodeData[16-1][1]),float(nodeData[16-1][2]),float(nodeData[16-1][3])), ),
                ((float(nodeData[24-1][1]),float(nodeData[24-1][2]),float(nodeData[24-1][3])), ),
                ((float(nodeData[32-1][1]),float(nodeData[32-1][2]),float(nodeData[32-1][3])), )))

# Make element sets of the different section types (for easier applying Abaqus Aqua settings)
sectionDictionary = {};
for i in sectionData: # Create the dictionary
        sectionDictionary[i[0]] = []
for i in sectionDictionary: # Go through the beams and put their names in their respective section
     key of the section dictionary
        for j in range(len(beamData)):
                if(beamData[j][4] == i):
                        sectionDictionary[i].append(beamData[j][3])

for i in sectionDictionary:
        els = [];
        for j in range(len(sectionDictionary[i])):
                els.append(PART.sets[sectionDictionary[i][j] + '_set'].elements)
        PART.Set(
                name = i + '_element_set',
                elements = els)
```

Listing 4: `piles.py`

```
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

import numpy as np


# Input
data = open('pileData.csv','r')
lines = data.readlines()
levels = []
sectionData = []
levels = [float(j) for j in lines[0].strip().split(',')]
matN = lines[1].strip()
E = float(lines[2])
nu = float(lines[3])
rho = float(lines[4])
alpha = float(lines[5])
beta = float(lines[6])
bw =  float(lines[7])
for i in range(9,14):
        temp = [str(j) for j in lines[i].strip().split(',')]
        sectionData.append(temp)
data.close()

mN = 'Model-1'
```

```
pN = 'Piles'
MODEL = mdb.models[mN] # Pass by reference

cX = (-0.5*bw, 0.5*bw, 0.5*bw, -0.5*bw) # Center x-coordinates
cY = (-0.5*bw, -0.5*bw, 0.5*bw, 0.5*bw) # Center y-coordinates

# Make material
MODEL.Material(
        name = matN)
MODEL.materials[matN].Density(
        table = ((rho, ), ))
MODEL.materials[matN].Elastic(
        table = ((E, nu), ))
MODEL.materials[matN].Damping(
        alpha = alpha,
        beta = beta)


# Make piles
MODEL.Part( # Making part
        dimensionality = THREE_D,
        name = pN,
        type = DEFORMABLE_BODY)
PART = MODEL.parts[pN] # Pass by reference
for i in range(len(sectionData)):
        MODEL.PipeProfile( # Pipe profile
                name = str(sectionData[i][0]) + '_profile',
                r = float(sectionData[i][1]),
                t = float(sectionData[i][2]))
        MODEL.BeamSection( # Make beam secitons
                name = str(sectionData[i][0]),
                integration = DURING_ANALYSIS,
                profile = str(sectionData[i][0]) + '_profile',
                material = matN,
                poissonRatio = nu,
                consistentMassMatrix = False)
wNr = 1
for i in range(4): # Four piles
        for j in range(len(levels)-1): # Different levels
                PART.WirePolyLine( # Wire feature
                        points = ((cX[i], cY[i], levels[j]), (cX[i], cY[i], levels[j+1])))
                featureName = 'Wire-' + str(wNr) # This is the name of the wire just created
                PART.Set( # Make set of wire (needed for section assignment)
                        name = 'Pile_' + str(i+1) + '_level_' + str(j+1),
                        edges = PART.getFeatureEdges(featureName))
                PART.SectionAssignment( # Assign section
                        region = PART.sets['Pile_' + str(i+1) + '_level_' + str(j+1)],
                        sectionName = str(sectionData[j][0]),
                        offsetType = MIDDLE_SURFACE)
                PART.assignBeamSectionOrientation( # Assign beam direction
                        method = N1_COSINES,
                        n1 = (0.0, 1.0, 0.0),
                        region = PART.sets['Pile_' + str(i+1) + '_level_' + str(j+1)])
                PART.seedEdgeByNumber( # Seed beam
                        constraint = FINER,
                        edges = PART.sets['Pile_' + str(i+1) + '_level_' + str(j+1)].edges,
                        number = int(sectionData[j][3]))
                PART.setElementType( # Set element type
                        elemTypes = (ElemType(
                                        elemCode=B31,
                                        elemLibrary=STANDARD), ),
                        regions = PART.sets['Pile_' + str(i+1) + '_level_' + str(j+1)])
                wNr += 1
PART.generateMesh()

# Make some extra sets:
PART.Set(
        name = 'Top_connection',
        vertices = PART.vertices.findAt(
                ((cX[0], cY[0], levels[-1]), ),
                ((cX[1], cY[1], levels[-1]), ),
                ((cX[2], cY[2], levels[-1]), ),
                ((cX[3], cY[3], levels[-1]), )))
for i in range(4): # Four piles
        PART.Set( # Whole pile set
                name = 'Pile_' + str(i+1),
                edges = PART.edges.getByBoundingCylinder(
                        center1 = (cX[i], cY[i], levels[0]),
                        center2 = (cX[i], cY[i], levels[-1]),
                        radius = 0.01))
        PART.Set( # Top node
                name = 'Pile_' + str(i+1) + '_top_node',
                vertices = PART.vertices.findAt(((cX[i], cY[i], levels[-1]), )))

# Find z-coordinate of nodes closest to mudline
nZc = np.linspace(levels[4],levels[5],int(sectionData[4][3])+1) # Calculating the z coordinates of
```

```
    the nodes at the top level section of the pile
a = abs(nZc - mudline)
b = min(a)
ix = np.where(a == b)
zC = float(nZc[ix[0][0]]) # Transforms from numpy.float type
for i in range(4): # Four piles
        PART.Set( # Set of mudline node
                name = 'Pile_' + str(i+1) + '_mudline_node',
                nodes = PART.nodes.getByBoundingSphere(
                        center = (cX[i], cY[i], zC),
                        radius = 0.01))
PART.SetByBoolean( # Set of all mudline nodes
        name = 'Mudlie_nodes',
        sets = (PART.sets['Pile_1_mudline_node'],
                        PART.sets['Pile_2_mudline_node'],
                        PART.sets['Pile_3_mudline_node'],
                        PART.sets['Pile_4_mudline_node']))
for i in range(4): # Four piles
        PART.Set( # Node set of all nodes ut to the node closest to the mudline. For soil connection
                name = 'Pile_' + str(i+1) + '_soil_connection',
                nodes = PART.nodes.getByBoundingCylinder(
                        center1 = (cX[i], cY[i], levels[0]),
                        center2 = (cX[i], cY[i], zC),
                        radius = 0.01))
PART.SetByBoolean( # Set of all connection nodes
        name = 'Soil_connection_nodes',
        sets = (PART.sets['Pile_1_soil_connection'],
                        PART.sets['Pile_2_soil_connection'],
                        PART.sets['Pile_3_soil_connection'],
                        PART.sets['Pile_4_soil_connection']))
```

Listing 5: `soil.py`

```
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *


# Load data
data = open('soilData.csv','r')
lines = data.readlines()
layerData = []
levels = []
E = []
nu = []
rho = []
C = []
bw = float(lines[0])
holeDepth = float(lines[1])
holeRadius = float(lines[2])
xWidth = float(lines[3])
yWidth = float(lines[4])
meshOffset = float(lines[5])
horizontalCoarseSeeding = float(lines[6])
horizontalFineSeeding = int(lines[7])
holeEdgeSeeding = int(lines[8])
maxFreqOfInterest = float(lines[9])
verticalMeshFactor = float(lines[10])
alpha = float(lines[11])
beta = float(lines[12])
mudline = float(lines[13])
for i in range(15,31): # Change the range if number of layers are changed
        temp = [float(j) for j in lines[i].strip().split(',')]
        layerData.append(temp)
for i in range(len(layerData)):
        levels.append(layerData[i][0])
        rho.append(layerData[i][1])
        E.append(layerData[i][2])
        nu.append(layerData[i][3])
        C.append(layerData[i][4])
levels.insert(0,mudline)


# Helping variables
offsetVector = (-0.5*bw - meshOffset, # For vertical datum planes
                                -0.5*bw,
```

```
                                -0.5*bw + meshOffset,
                                +0.5*bw - meshOffset,
                                +0.5*bw,
                                +0.5*bw + meshOffset)
holeCenter = ((-0.5*bw, -0.5*bw), # x and y coordinates of hole centers
                          (+0.5*bw, -0.5*bw),
                          (+0.5*bw, +0.5*bw),
                          (-0.5*bw, +0.5*bw))
sideOuter = ((0, -0.5*yWidth), # x and y coordinates for the outer edges of the sides
                         (0.5*xWidth, 0),
                         (0, 0.5*yWidth),
                         (-0.5*xWidth, 0))
sideNormal = ((0, -0.49*yWidth), # x and y coordinates for the normal edges of the sides
                         (0.49*xWidth, 0),
                         (0, 0.49*yWidth),
                         (-0.49*xWidth, 0))
outerFine = (-0.5*bw - 0.5*meshOffset, # Points along the outer edges to find fine edges
                         -0.5*bw + 0.5*meshOffset,
                         +0.5*bw - 0.5*meshOffset,
                         +0.5*bw + 0.5*meshOffset)
outerCoarse = (-0.5*bw - 1.1*meshOffset, # Points along the outer edges to find coarse edges
                          0,
                          +0.5*bw + 1.1*meshOffset)
normalCoarse = (-0.5*bw - meshOffset, # Points along the normal edges to find coarse edges
                                -0.5*bw,
                                -0.5*bw + meshOffset,
                                +0.5*bw - meshOffset,
                                +0.5*bw,
                                +0.5*bw + meshOffset)
holeFine = ((-0.5*meshOffset, -1.0*meshOffset), # x-y offset coordinates for point on fine edges
     around each hole
                        (+0.5*meshOffset, -1.0*meshOffset),
                        (+1.0*meshOffset, -0.5*meshOffset),
                        (+1.0*meshOffset, +0.5*meshOffset),
                        (+0.5*meshOffset, +1.0*meshOffset),
                        (-0.5*meshOffset, +1.0*meshOffset),
                        (-1.0*meshOffset, +0.5*meshOffset),
                        (-1.0*meshOffset, -0.5*meshOffset))


# Model setup
mN = 'Model-1'
pN = 'Soil'
MODEL = mdb.models[mN]


# Make the different soil layer materials:
for i in range(len(levels)-1):
        MODEL.Material(
                name = 'Soil_layer_' + str(i+1))
        MODEL.materials['Soil_layer_' + str(i+1)].Density(
                table = ((rho[i], ), ))
        MODEL.materials['Soil_layer_' + str(i+1)].Elastic(
                table = ((E[i], nu[i]), ))
        MODEL.materials['Soil_layer_' + str(i+1)].Damping(
                alpha = alpha,
                beta = beta)
        MODEL.materials['Soil_layer_' + str(i+1)].MohrCoulombPlasticity(
                table = ((0.0, 0.0), ))
        MODEL.materials['Soil_layer_' + str(i+1)].mohrCoulombPlasticity.MohrCoulombHardening(
                table = ((C[i], 0.0), ))
        MODEL.materials['Soil_layer_' + str(i+1)].mohrCoulombPlasticity.TensionCutOff(
                dependencies = 0,
                table = ((0.0, 0.0), ),
                temperatureDependency = OFF)


# Make the soil box
MODEL.Part( # Making part
        dimensionality = THREE_D,
        name = pN,
        type = DEFORMABLE_BODY)
PART = MODEL.parts[pN] # Pass by reference
PART.ReferencePoint(
        point = (0.0, 0.0, 0.0))
PART.DatumPlaneByPrincipalPlane(
        offset = mudline,
        principalPlane = XYPLANE)
datumPlaneID = PART.features['Datum_plane-1'].id
PART.DatumAxisByPrincipalAxis(
        principalAxis = XAXIS)
datumAxisID = PART.features['Datum_axis-1'].id
MODEL.ConstrainedSketch(
        gridSpacing = 16.46,
        name = '__profile__',
    sheetSize = 658.78,
```

```
    transform = PART.MakeSketchTransform(
        sketchPlane = PART.datums[datumPlaneID],
        sketchPlaneSide = SIDE1,
        sketchUpEdge = PART.datums[datumAxisID],
        sketchOrientation = BOTTOM,
        origin=(0.0, 0.0, mudline)))
PART.projectReferencesOntoSketch(
        filter = COPLANAR_EDGES,
        sketch = MODEL.sketches['__profile__'])
MODEL.sketches['__profile__'].rectangle(
        point1 = (-0.5*xWidth, -0.5*yWidth),
    point2 = (0.5*xWidth, 0.5*yWidth))
PART.SolidExtrude(
        depth = mudline-levels[-1],
    flipExtrudeDirection = ON,
    sketch = MODEL.sketches['__profile__'],
    sketchOrientation = BOTTOM,
    sketchPlane = PART.datums[datumPlaneID],
    sketchPlaneSide = SIDE1,
    sketchUpEdge = PART.datums[datumAxisID])
del MODEL.sketches['__profile__']


# Dig the holes
MODEL.ConstrainedSketch(
        gridSpacing = 14.14,
        name = '__profile__',
        sheetSize = 565.68,
        transform = PART.MakeSketchTransform(
                sketchPlane = PART.datums[datumPlaneID],
                sketchPlaneSide = SIDE1,
                sketchUpEdge = PART.datums[datumAxisID],
                sketchOrientation = BOTTOM,
                origin = (0.0, 0.0, mudline)))
PART.projectReferencesOntoSketch(
        filter = COPLANAR_EDGES,
        sketch = MODEL.sketches['__profile__'])
MODEL.sketches['__profile__'].CircleByCenterPerimeter(
        center = holeCenter[0],
        point1 = (holeCenter[0][0]-holeRadius, holeCenter[0][1]))
MODEL.sketches['__profile__'].linearPattern(
        angle1=0.0,
        angle2 = 90.0,
        geomList = (MODEL.sketches['__profile__'].geometry[7], ),
        number1 = 2,
        number2 = 2,
        spacing1 = bw,
        spacing2 = bw,
        vertexList = ())
PART.CutExtrude(
        depth = holeDepth,
    flipExtrudeDirection = OFF,
    sketch = MODEL.sketches['__profile__'],
    sketchOrientation = BOTTOM,
    sketchPlane = PART.datums[datumPlaneID],
    sketchPlaneSide = SIDE1,
    sketchUpEdge = PART.datums[datumAxisID])
del MODEL.sketches['__profile__']
PART.DatumPlaneByPrincipalPlane(
        offset = mudline - holeDepth,
        principalPlane = XYPLANE)
datumPlaneID = PART.features['Datum plane-2'].id
PART.PartitionCellByDatumPlane( # Partition horizontal plane
        cells = PART.cells,
        datumPlane = PART.datums[datumPlaneID])
for i in range(4): # Partition soil under holes
        eLine = PART.edges.findAt(((-0.5*xWidth, -0.5*yWidth, levels[-1] + 0.001), )) # Lower vertical edge to extrude along
        PART.PartitionCellByExtrudeEdge(
                cells = PART.cells,
                edges = PART.edges.findAt(((holeCenter[i][0], holeCenter[i][1] + holeRadius, mudline - holeDepth), )),
                line = eLine[0], # Need a specific reference variable to the given edge
                sense = FORWARD)


# Make layers
coincide = False
planeNr = 3 # Next datum plane created will be "Datum plane-3"
for i in range(1,len(levels)-1): # levels[-1] is the soil box bottom
        if(levels[i] != mudline - holeDepth): # If one of the layer lays at the level of pile bottom, the datum is already created
                PART.DatumPlaneByPrincipalPlane(
                        offset = levels[i],
                        principalPlane = XYPLANE)
                datumPlaneID = PART.features['Datum plane-'+ str(planeNr)].id
```

```
                planeNr += 1
                PART.PartitionCellByDatumPlane(
                        cells = PART.cells,
                        datumPlane = PART.datums[datumPlaneID])
        else:
                coincide = True
bottomPileLayer = -1
if(coincide == False): # If none of the layers coincide with the bottom of the pile, the bottom pile
     level need to be added to the "levels" list. But keep the original levels in "layerLevels"
        for i in range(len(levels)-1): # Identify layer where the bottom pile is
                if((mudline - holeDepth > levels[i+1]) and (mudline - holeDepth < levels[i])):
                        layerLevels = levels[:]
                        levels.insert(i+1,mudline - holeDepth)
                        bottomPileLayer = i+1 # save the index of the layer
                        break


# Make layer cell sets
for i in range(len(layerLevels)-1): # Make rest
                PART.Set(
                        name = 'Layer_' + str(i+1),
                        cells = PART.cells.getByBoundingBox(
                                xMin =-0.5*xWidth, xMax = 0.5*xWidth,
                                yMin =-0.5*yWidth, yMax = 0.5*yWidth,
                                zMin = layerLevels[i+1], zMax = layerLevels[i]))


# Make hole edge set
holeEdges = []
for i in range(len(levels)):
        for h in range(4): # Four holes
                e = PART.edges.findAt(((holeCenter[h][0], holeCenter[h][1] + holeRadius,levels[i]) ,)
                     )
                holeEdges.append(e)
PART.Set(
        name = 'Hole_edges',
        edges = holeEdges)


# Vertical partitioning
for i in range(len(offsetVector)):
        PART.DatumPlaneByPrincipalPlane(
                offset = offsetVector[i],
                principalPlane = YZPLANE)
        datumPlaneID = PART.features['Datum plane-' + str(planeNr)].id
        PART.PartitionCellByDatumPlane(
                cells = PART.cells,
                datumPlane = PART.datums[datumPlaneID])
        planeNr += 1
        PART.DatumPlaneByPrincipalPlane(
                offset = offsetVector[i],
                principalPlane = XZPLANE)
        datumPlaneID = PART.features['Datum plane-' + str(planeNr)].id
        PART.PartitionCellByDatumPlane(
                cells = PART.cells,
                datumPlane = PART.datums[datumPlaneID])
        planeNr += 1


# Make fine edge set
fineEdges = []
for z in range(len(levels)): # Levels
        for s in range(4): # Four sides
                for p in range(4): # Four edge pieces per side
                        if(s == 0 or s == 2): # Side 1 and 3
                                e = PART.edges.findAt(((sideOuter[s][0] + outerFine[p], sideOuter[s
                                     ][1], levels[z]), ))
                                fineEdges.append(e)
                        else: # Side 2 and 4
                                e = PART.edges.findAt(((sideOuter[s][0], sideOuter[s][1] + outerFine[
                                     p], levels[z]), ))
                                fineEdges.append(e)
        for h in range(4): # Four holes
                for p in range(8): # Eight parts per hole
                        e = PART.edges.findAt(((holeCenter[h][0] + holeFine[p][0], holeCenter[h][1] +
                             holeFine[p][1], levels[z]), ))
                        fineEdges.append(e)
PART.Set(
        name = 'Fine_edges',
        edges = fineEdges)

# Make coarse edge set
coarseEdges = []
for z in range(len(levels)): # Levels
        for s in range(4): # Four sides
                for p in range(3): # 3 edge parts per side
```

```
                        if(s == 0 or s == 2): # Side 1 and 3
                                e = PART.edges.findAt(((sideOuter[s][0] + outerCoarse[p], sideOuter[s
                                    ][1], levels[z]), ))
                                coarseEdges.append(e)
                        else:
                                e = PART.edges.findAt(((sideOuter[s][0], sideOuter[s][1] +
                                    outerCoarse[p], levels[z]), ))
                                coarseEdges.append(e)
                for n in range(6): # Six normal edge part on each side
                        if(s == 0 or s == 2): # Side 1 and 3
                                e = PART.edges.findAt(((sideNormal[s][0] + normalCoarse[n],
                                    sideNormal[s][1], levels[z]), ))
                                coarseEdges.append(e)
                        else:
                                e = PART.edges.findAt(((sideNormal[s][0], sideNormal[s][1] +
                                    normalCoarse[n], levels[z]), ))
                                coarseEdges.append(e)
        for a in range(2): # Two axis
                for o in range(6): # Inner edges on the mesh offset planes
                        if(a == 0): # Along x-axis
                                e = PART.edges.findAt(((0 + offsetVector[o], 0, levels[z]), ))
                                coarseEdges.append(e)
                        else: # Along y-axis
                                e = PART.edges.findAt(((0, 0 + offsetVector[o], levels[z]), ))
                                coarseEdges.append(e)
PART.Set(
        name = 'Coarse_edges',
        edges = coarseEdges)


# Make vertical edges set
for l in range(len(levels)-1): # Layers
        verticalEdges = []
        lEdges = PART.edges.getByBoundingBox( # All edges of that layer
                                xMin =-0.5*xWidth, xMax = 0.5*xWidth,
                                yMin =-0.5*yWidth, yMax = 0.5*yWidth,
                                zMin = levels[l+1], zMax = levels[l])
        for e in range(len(lEdges)):
                zTest = lEdges[e].pointOn[0][2]
                # If the edge is not in the xy-planes of the layer, it is a vertical edge
                if(((zTest in levels) == False) and (zTest != mudline - holeDepth)):
                        verticalEdges.append(PART.edges.findAt(lEdges[e].pointOn))
        PART.Set(
                name = 'Vertical_edges_' + str(l+1),
                edges = verticalEdges)



# Make and assign section
for i in range(len(layerLevels)-1):
        MODEL.HomogeneousSolidSection(
                material = 'Soil_layer_' + str(i+1),
                name = 'Soil_layer_' + str(i+1) + '_section',
                thickness = None)
        PART.SectionAssignment(
                offset = 0.0,
                offsetField = '',
                offsetType = MIDDLE_SURFACE,
                region = PART.sets['Layer_' + str(i+1)],
                sectionName = 'Soil_layer_' + str(i+1) + '_section',
                thicknessAssignment = FROM_SECTION)


# Seed edges
PART.seedEdgeBySize(
        constraint = FINER,
        deviationFactor = 0.1,
        edges = PART.sets['Coarse_edges'].edges,
        size = horizontalCoarseSeeding)
PART.seedEdgeByNumber( # Fine edges
        constraint = FINER,
        edges = PART.sets['Fine_edges'].edges,
        number = horizontalFineSeeding)
PART.seedEdgeByNumber( # Hole edges
        constraint = FINER,
        edges = PART.sets['Hole_edges'].edges,
        number = holeEdgeSeeding)
layerIx = 0
for i in range(len(levels)-1):
        if(i == bottomPileLayer): # If the bottom pile level coincide with one of the soil layers,
            bottomPile is -1, and the if statement will never run
                layerIx -= 1
        d = abs(levels[i+1] - levels[i])
        s = ((E[layerIx] / (2*(1+nu[layerIx])*rho[layerIx]))**0.5)/(maxFreqOfInterest*
            verticalMeshFactor) # v_s/(f_max*8)
        if(s>d): # If the max length is more than the layer depth
```

```
                nr = 1
        else:
                nr = d // s + 1 # Makes the layer fullfill the length requirement
        PART.seedEdgeByNumber( # Vertical edges
                constraint = FINER,
                edges = PART.sets['Vertical_edges_' + str(i+1)].edges,
                number = int(nr))
        layerIx += 1


# Set element type
PART.setElementType(
        elemTypes = (ElemType( # Hed
                                        elemCode = C3D8R,
                                        elemLibrary = STANDARD,
                                        secondOrderAccuracy = OFF,
                                        kinematicSplit = AVERAGE_STRAIN,
                                        hourglassControl = DEFAULT,
                                        distortionControl = DEFAULT),
                                ElemType( # Wedge
                                        elemCode = C3D6,
                                        elemLibrary = STANDARD),
                                ElemType( # Tet
                                        elemCode = C3D4,
                                        elemLibrary = STANDARD)),
        regions = Region(
                cells = PART.cells))


# Set mesh control
PART.setMeshControls(
        technique = SWEEP,
        algorithm = MEDIAL_AXIS,
        regions = PART.cells)


# Mesh part
PART.generateMesh()


# Add some more sets
holeFaces = []
for i in range(4): # Four holes
        f = PART.faces.getByBoundingCylinder(
                center1 = (holeCenter[i][0], holeCenter[i][1], mudline - holeDepth),
                center2 = (holeCenter[i][0], holeCenter[i][1], mudline),
                radius = holeRadius+0.01)
        holeFaces.append(f)
PART.Set(
        name = 'Hole_surfaces',
        faces = holeFaces)
PART.Set(
        name = 'Base',
        faces = PART.faces.getByBoundingBox(
                xMin = -xWidth,
                yMin = -yWidth,
                zMin = levels[-1] - 0.001,
                xMax = xWidth,
                yMax = yWidth,
                zMax = levels[-1] + 0.001))
PART.Set(
        name = 'Side_1',
        faces = PART.faces.getByBoundingBox(
                xMin = -xWidth,
                yMin = -0.5*yWidth - 0.001,
                zMin = levels[-1] - 0.001,
                xMax = xWidth,
                yMax = -0.5*yWidth + 0.001,
                zMax = mudline + 0.001))
PART.Set(
        name = 'Side_2',
        faces = PART.faces.getByBoundingBox(
                xMin = 0.5*xWidth - 0.001,
                yMin = -yWidth,
                zMin = levels[-1] - 0.001,
                xMax = 0.5*xWidth + 0.001,
                yMax = yWidth,
                zMax = mudline + 0.001))
PART.Set(
        name = 'Side_3',
        faces = PART.faces.getByBoundingBox(
                xMin = -xWidth,
                yMin = 0.5*yWidth - 0.001,
                zMin = levels[-1] - 0.001,
                xMax = xWidth,
                yMax = 0.5*yWidth + 0.001,
```

```
                zMax = mudline + 0.001))
PART.Set(
        name = 'Side_4',
        faces = PART.faces.getByBoundingBox(
                xMin = -0.5*xWidth - 0.001,
                yMin = -yWidth,
                zMin = levels[-1] - 0.001,
                xMax = -0.5*xWidth + 0.001,
                yMax = yWidth,
                zMax = mudline + 0.001))
PART.SetByBoolean( # x-sides
        name = 'x_sides',
        sets = (PART.sets['Side_2'],
                        PART.sets['Side_4']))
PART.SetByBoolean( # y-sides
        name = 'y_sides',
        sets = (PART.sets['Side_1'],
                        PART.sets['Side_3']))
PART.SetByBoolean( # All sides
        name = 'All_sides',
        sets = (PART.sets['Side_1'],
                        PART.sets['Side_2'],
                        PART.sets['Side_3'],
                        PART.sets['Side_4']))

# Make node set for base boundary condition
PART.Set(
        name = 'Base_nodes',
        nodes = PART.nodes.getByBoundingBox(
                xMin = -xWidth, xMax = xWidth,
                yMin = -yWidth, yMax = yWidth,
                zMin = levels[-1] - 0.001, zMax = levels[-1] + 0.001))

# Make node sets for MPC tie constraints
cornerNodes = PART.nodes.getByBoundingBox( # To get the height of the node levels
        xMin = -0.5*xWidth - 0.01, xMax = -0.5*xWidth + 0.01,
        yMin = -0.5*yWidth  -0.01, yMax = -0.5*yWidth + 0.01,
        zMin = levels[-1] + 0.01, zMax = mudline)
for i in range(len(cornerNodes)): # Go through all heights
        z = cornerNodes[i].coordinates[2]
        PART.Set(
                name = 'level_' + str(i+1) + '_slave_nodes', # Note that this will not be sorted
                     according to z-coordinates
                nodes = (PART.nodes.getByBoundingBox( # Side 1
                                        xMin = -0.5*xWidth + 0.01, xMax = 0.5*xWidth ,
                                        yMin = -0.5*yWidth, yMax = -0.5*yWidth + 0.01,
                                        zMin = z - 0.01, zMax = z + 0.01),
                                PART.nodes.getByBoundingBox( # Side 2
                                        xMin = 0.5*xWidth - 0.01, xMax = 0.5*xWidth,
                                        yMin = -0.5*yWidth + 0.01, yMax = 0.5*yWidth,
                                        zMin = z - 0.01, zMax = z + 0.01),
                                PART.nodes.getByBoundingBox( # Side 3
                                        xMin = -0.5*xWidth, xMax = 0.5*xWidth - 0.01,
                                        yMin = 0.5*yWidth - 0.01, yMax = 0.5*yWidth,
                                        zMin = z - 0.01, zMax = z + 0.01),
                                PART.nodes.getByBoundingBox( # Side 4
                                        xMin = -0.5*xWidth, xMax = -0.5*xWidth + 0.01,
                                        yMin = -0.5*yWidth + 0.01, yMax = 0.5*yWidth - 0.01,
                                        zMin = z - 0.01, zMax = z + 0.01)))
        PART.Set(
                name = 'level_' + str(i+1) + '_master_node',
                nodes = PART.nodes.getByBoundingSphere(
                        center = (-0.5*xWidth, -0.5*yWidth,z),
                        radius = 0))
```

Listing 6: `assemble_full_model.py`

```
execfile('soil.py')
execfile('piles.py')
execfile('jacket.py')
execfile('transitionPiece.py')
execfile('tower.py')


MODEL.rootAssembly.DatumCsysByDefault(CARTESIAN)
ASSEMBLY = MODEL.rootAssembly

# Make instances
ASSEMBLY.Instance(
        name = 'Soil',
        part = MODEL.parts['Soil'],
        dependent = ON)
ASSEMBLY.Instance(
        name = 'Piles',
        part = MODEL.parts['Piles'],
        dependent = ON)
```

```
ASSEMBLY.Instance(
        name = 'Jacket',
        part = MODEL.parts['Jacket'],
        dependent = ON)
ASSEMBLY.Instance(
        name = 'TP',
        part = MODEL.parts['Transition_piece'],
        dependent = ON)
ASSEMBLY.Instance(
        name = 'Tower',
        part = MODEL.parts['Tower'],
        dependent = ON)
SOIL = ASSEMBLY.instances['Soil']
PILES = ASSEMBLY.instances['Piles']
JACKET = ASSEMBLY.instances['Jacket']
TP = ASSEMBLY.instances['TP']
TOWER = ASSEMBLY.instances['Tower']


# Tie the parts together
MODEL.Tie( # Tie piles to soil. Pile as master
        name = 'Piles_to_Soil',
        master = PILES.sets['Soil_connection_nodes'],
        slave = SOIL.sets['Hole_surfaces'],
        adjust = OFF,
        positionToleranceMethod = SPECIFIED,
        positionTolerance = holeRadius*2) # holeRadius from soil.py. *2 to be sure every node finds a
              connection
MODEL.Tie( # Tie piles to jacket. Pile as master
        name = 'Piles_to_Jacket',
        master = PILES.sets['Top_connection'],
        slave = JACKET.sets['Lower_connection_set'])
MODEL.Tie( # Tie jacket to TP. Jacket master
        name = 'Jacet_to_TP',
        master = JACKET.sets['Upper_connection_set'],
        slave = TP.sets['Lower_connection_set'])
MODEL.Tie( # Tie TP to tower. TP master
        name = 'TP_to_Tower',
        master = TP.sets['Upper_connection_set'],
        slave = TOWER.sets['Bottom_node'])


# Tie soil layers (each mesh layer outer nodes are tied together)
for i in range(len(cornerNodes)):
        MODEL.MultipointConstraint(
                name = 'MPCtie_' + str(i+1),
                controlPoint = SOIL.sets['level_' + str(i+1) + '_master_node'], # Master
                surface = SOIL.sets['level_' + str(i+1) + '_slave_nodes'], # Slave
                mpcType = TIE_MPC)


# Set boundary conditions
MODEL.PinnedBC(
        name = 'Pinned_base',
        createStepName = 'Initial',
        region = SOIL.sets['Base_nodes'])


# Set view
VIEW = session.viewports['Viewport: 1']
VIEW.setValues(
        displayedObject = ASSEMBLY)
VIEW.assemblyDisplay.setValues(
        renderBeamProfiles = ON,
        mesh = OFF,
        renderStyle = SHADED)
VIEW.assemblyDisplay.geometryOptions.setValues(
        datumPoints = OFF,
        datumAxes = OFF,
        datumPlanes = OFF,
        referencePointLabels = OFF,
        referencePointSymbols = OFF,
        datumCoordSystems = ON)
session.View( # Set User-1 view
        name = 'User-1',
        nearPlane = 314.55,
        farPlane = 766.22,
        width = 527.27,
        height = 252.55,
        projection = PERSPECTIVE,
        cameraPosition = (-293.47, -427.39, 149.41),
        cameraUpVector = (0.34203, 0.50761, 0.79079),
        cameraTarget = (71.143, 83.475, -57.634),
        viewOffsetX = 77.216,
        viewOffsetY = -3.5872,
        autoFit = OFF)
```

```
VIEW.view.setValues(session.views['User-1']) # Set current camera view
VIEW.view.setRotationCenter(
        rotationCenter = (0.0,0.0,0.0))


# Set colors:
CMAP = VIEW.colorMappings['Section']
CMAP.updateOverrides(
        overrides =
        {'Jacket_section_Hbars':(True, '#FFD700','Default', '#FFD700'),
    'Jacket_section_bracings_level_1':(True,'#FFD700', 'Default', '#FFD700'),
    'Jacket_section_bracings_level_2':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_bracings_level_3':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_bracings_level_4':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_0':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_1':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_2':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_3':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_4':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_B':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_T':(True, '#FFD700', 'Default', '#FFD700'),
    'Pile_section_level_1':(True, '#999999', 'Default', '#999999'),
    'Pile_section_level_2':(True, '#999999', 'Default', '#999999'),
    'Pile_section_level_3':(True, '#999999', 'Default', '#999999'),
    'Pile_section_level_4':(True, '#999999', 'Default', '#999999'),
    'Pile_section_level_5':(True, '#999999', 'Default', '#999999'),
    'TP_section_BM_leg':(True, '#FFD700', 'Default', '#FFD700'),
    'TP_section_B_leg':(True, '#FFD700', 'Default', '#FFD700'),
    'TP_section_MT_leg': (True, '#FFD700', 'Default', '#FFD700'),
    'TP_section_T_leg': (True, '#FFD700', 'Default', '#FFD700'),
    'TP_section_tower_piece': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_1': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_2': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_3': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_4': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_5': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_6': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_7': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_8': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_9': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_10': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_11': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_12': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_13': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_14': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_15': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_16': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_17': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_18': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_19': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_20': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_21': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_22': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_23': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_24': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_25': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_26': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_27': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_28': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_29': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_30': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_31': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_32': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_33': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_34': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_35': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_36': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_37': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_38': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_39': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_40': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_41': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_42': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_43': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_44': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_45': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_46': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_47': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_48': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_49': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_50': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_51': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_52': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_53': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_54': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_55': (True, '#CCCCCC', 'Default', '#CCCCCC'),
```

```
    'Tower_section_56': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_57': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_58': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_59': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_60': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_61': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_62': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_63': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_64': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_65': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_66': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_67': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_68': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_69': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_70': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_71': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_72': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_73': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_74': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_75': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_76': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_77': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_78': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_79': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_80': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_81': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_82': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_83': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_84': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_85': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_86': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_87': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_88': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_89': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_90': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_91': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_92': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_93': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_94': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_95': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_96': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_97': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_98': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_99': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_100': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_101': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_102': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_103': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_104': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_105': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_106': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_107': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_108': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_109': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_110': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_111': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_112': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_113': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_114': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_115': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_116': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_117': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_118': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_119': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_120': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_121': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_122': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_123': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_124': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_125': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_126': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_127': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_128': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_129': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_130': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_131': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_132': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_133': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_134': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_135': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_136': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_137': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_138': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_139': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_140': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_141': (True, '#CCCCCC', 'Default', '#CCCCCC'),
```

```
    'Tower_section_142': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_143': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_144': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_145': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_146': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_147': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_148': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_149': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_150': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_151': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_152': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_153': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_154': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_155': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_156': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_157': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_158': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_159': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_160': (True, '#CCCCCC', 'Default', '#CCCCCC')})
VIEW.setColor(colorMapping = CMAP)
```

Listing 7: `soilSlice.py`

```
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *


# Load data
data = open('soilData.csv','r')
lines = data.readlines()
layerData = []
levels = []
E = []
nu = []
rho = []
C = []
bw = float(lines[0])
holeDepth = float(lines[1])
holeRadius = float(lines[2])
xWidth = float(lines[3])
yWidth = float(lines[4])
meshOffset = float(lines[5])
horizontalCoarseSeeding = float(lines[6])
horizontalFineSeeding = int(lines[7])
holeEdgeSeeding = int(lines[8])
maxFreqOfInterest = float(lines[9])
verticalMeshFactor = float(lines[10])
alpha = float(lines[11])
beta = float(lines[12])
mudline = float(lines[13])
for i in range(15,31): # Change the range if number of layers are changed
        temp = [float(j) for j in lines[i].strip().split(',')]
        layerData.append(temp)
for i in range(len(layerData)):
        levels.append(layerData[i][0])
        rho.append(layerData[i][1])
        E.append(layerData[i][2])
        nu.append(layerData[i][3])
        C.append(layerData[i][4])
levels.insert(0,mudline)

# Set slice specific variables:
yWidth = 1


# Helping variables
offsetVector = (-0.5*bw - meshOffset, # For vertical datum planes
                                -0.5*bw,
                                -0.5*bw + meshOffset,
                                +0.5*bw - meshOffset,
                                +0.5*bw,
                                +0.5*bw + meshOffset)
coarseEdgeXY = ((-0.5*bw - 1.1*meshOffset,-0.5*yWidth),
                                (0.5*bw + 1.1*meshOffset,-0.5*yWidth),
                                (0,-0.5*yWidth),
```

```
                                (0.5*bw + 1.1*meshOffset,0.5*yWidth),
                                (-0.5*bw - 1.1*meshOffset,0.5*yWidth),
                                (0,0.5*yWidth))
fineEdgeXY = ((-0.5*bw - 0.5*meshOffset,-0.5*yWidth),
                          (-0.5*bw + 0.5*meshOffset,-0.5*yWidth),
                          (0.5*bw - 0.5*meshOffset,-0.5*yWidth),
                          (0.5*bw + 0.5*meshOffset,-0.5*yWidth),
                          (0.5*bw + 0.5*meshOffset,0.5*yWidth),
                          (0.5*bw - 0.5*meshOffset,0.5*yWidth),
                          (-0.5*bw + 0.5*meshOffset,0.5*yWidth),
                          (-0.5*bw - 0.5*meshOffset,0.5*yWidth))


# Model setup
mN = 'Model-1'
pN = 'Soil_slice'
MODEL = mdb.models[mN]


# Make the different soil layer materials:
for i in range(len(levels)-1):
        MODEL.Material(
                name = 'Soil_layer_' + str(i+1))
        MODEL.materials['Soil_layer_' + str(i+1)].Density(
                table = ((rho[i], ), ))
        MODEL.materials['Soil_layer_' + str(i+1)].Elastic(
                table = ((E[i], nu[i]), ))
        MODEL.materials['Soil_layer_' + str(i+1)].Damping(
                alpha = alpha,
                beta = beta)
        MODEL.materials['Soil_layer_' + str(i+1)].MohrCoulombPlasticity(
                table = ((0.0, 0.0), ))
        MODEL.materials['Soil_layer_' + str(i+1)].mohrCoulombPlasticity.MohrCoulombHardening(
                table = ((C[i], 0.0), ))
        MODEL.materials['Soil_layer_' + str(i+1)].mohrCoulombPlasticity.TensionCutOff(
                dependencies = 0,
                table = ((0.0, 0.0), ),
                temperatureDependency = OFF)

# Make the soil box
MODEL.Part( # Making part
        dimensionality = THREE_D,
        name = pN,
        type = DEFORMABLE_BODY)
PART = MODEL.parts[pN] # Pass by reference
PART.ReferencePoint(
        point = (0.0, 0.0, 0.0))
PART.DatumPlaneByPrincipalPlane(
        offset = mudline,
        principalPlane = XYPLANE)
datumPlaneID = PART.features['Datum_plane-1'].id
PART.DatumAxisByPrincipalAxis(
        principalAxis = XAXIS)
datumAxisID = PART.features['Datum_axis-1'].id
MODEL.ConstrainedSketch(
        gridSpacing = 16.46,
        name = '__profile__',
    sheetSize = 658.78,
    transform = PART.MakeSketchTransform(
        sketchPlane = PART.datums[datumPlaneID],
        sketchPlaneSide = SIDE1,
        sketchUpEdge = PART.datums[datumAxisID],
        sketchOrientation = BOTTOM,
        origin=(0.0, 0.0, mudline)))
PART.projectReferencesOntoSketch(
        filter = COPLANAR_EDGES,
        sketch = MODEL.sketches['__profile__'])
MODEL.sketches['__profile__'].rectangle(
        point1 = (-0.5*xWidth, -0.5*yWidth),
    point2 = (0.5*xWidth, 0.5*yWidth))
PART.SolidExtrude(
        depth = mudline-levels[-1],
    flipExtrudeDirection = ON,
    sketch = MODEL.sketches['__profile__'],
    sketchOrientation = BOTTOM,
    sketchPlane = PART.datums[datumPlaneID],
    sketchPlaneSide = SIDE1,
    sketchUpEdge = PART.datums[datumAxisID])
del MODEL.sketches['__profile__']

PART.DatumPlaneByPrincipalPlane(
        offset = mudline - holeDepth,
        principalPlane = XYPLANE)
datumPlaneID = PART.features['Datum_plane-2'].id
PART.PartitionCellByDatumPlane( # Partition horizontal plane
        cells = PART.cells,
```

```
        datumPlane = PART.datums[datumPlaneID])

# Make layers
coincide = False
planeNr = 3 # Next datum plane created will be "Datum plane-3"
for i in range(1,len(levels)-1): # levels[-1] is the soil box bottom
        if(levels[i] != mudline - holeDepth): # If one of the layer lays at the level of pile bottom,
              the datum is already created
                PART.DatumPlaneByPrincipalPlane(
                        offset = levels[i],
                        principalPlane = XYPLANE)
                datumPlaneID = PART.features['Datum_plane-'+ str(planeNr)].id
                planeNr += 1
                PART.PartitionCellByDatumPlane(
                        cells = PART.cells,
                        datumPlane = PART.datums[datumPlaneID])
        else:
                coincide = True
bottomPileLayer = -1
if(coincide == False): # If none of the layers coincide with the bottom of the pile, the bottom pile
     level need to be added to the "levels" list. But keep the original levels in "layerLevels"
        for i in range(len(levels)-1): # Identify layer where the bottom pile is
                if((mudline - holeDepth > levels[i+1]) and (mudline - holeDepth < levels[i])):
                        layerLevels = levels[:]
                        levels.insert(i+1,mudline - holeDepth)
                        bottomPileLayer = i+1 # save the index of the layer
                        break


# Make layer cell sets
for i in range(len(layerLevels)-1): # Make rest
                PART.Set(
                        name = 'Layer_' + str(i+1),
                        cells = PART.cells.getByBoundingBox(
                                xMin =-0.5*xWidth, xMax = 0.5*xWidth,
                                yMin =-0.5*yWidth, yMax = 0.5*yWidth,
                                zMin = layerLevels[i+1], zMax = layerLevels[i]))

# Vertical partitioning
for i in range(len(offsetVector)):
        PART.DatumPlaneByPrincipalPlane(
                offset = offsetVector[i],
                principalPlane = YZPLANE)
        datumPlaneID = PART.features['Datum_plane-' + str(planeNr)].id
        PART.PartitionCellByDatumPlane(
                cells = PART.cells,
                datumPlane = PART.datums[datumPlaneID])
        planeNr += 1



# Make fine edge set
fineEdges = []
for i in range(len(levels)):
        for j in range(len(fineEdgeXY)):
                e = PART.edges.findAt(((fineEdgeXY[j][0], fineEdgeXY[j][1], levels[i]), ))
                fineEdges.append(e)
PART.Set(
        name = 'Fine_edges',
        edges = fineEdges)

# Make coarse edges set
coarseEdges = []
for i in range(len(levels)):
        for j in range(len(coarseEdgeXY)):
                e = PART.edges.findAt(((coarseEdgeXY[j][0], coarseEdgeXY[j][1], levels[i]), ))
                coarseEdges.append(e)
PART.Set(
        name = 'Coarse_edges',
        edges = coarseEdges)

# Make vertical edges set
for l in range(len(levels)-1): # Layers
        verticalEdges = []
        lEdges = PART.edges.getByBoundingBox( # All edges of that layer
                                xMin =-0.5*xWidth, xMax = 0.5*xWidth,
                                yMin =-0.5*yWidth, yMax = 0.5*yWidth,
                                zMin = levels[l+1], zMax = levels[l])
        for e in range(len(lEdges)):
                zTest = lEdges[e].pointOn[0][2]
                # If the edge is not in the xy-planes of the layer, it is a vertical edge
                if(((zTest in levels) == False) and (zTest != mudline - holeDepth)):
                        verticalEdges.append(PART.edges.findAt(lEdges[e].pointOn))
        PART.Set(
                name = 'Vertical_edges_' + str(l+1),
                edges = verticalEdges)
```

```
# Make and assign section
for i in range(len(layerLevels)-1):
        MODEL.HomogeneousSolidSection(
                material = 'Soil_layer_' + str(i+1),
                name = 'Soil_layer_' + str(i+1) + '_section',
                thickness = None)
        PART.SectionAssignment(
                offset = 0.0,
                offsetField = '',
                offsetType = MIDDLE_SURFACE,
                region = PART.sets['Layer_' + str(i+1)],
                sectionName = 'Soil_layer_' + str(i+1) + '_section',
                thicknessAssignment = FROM_SECTION)


# Seed edges
PART.seedEdgeByNumber(
        constraint = FINER,
        edges = PART.edges,
        number = 1)
PART.seedEdgeBySize(
        constraint = FINER,
        deviationFactor = 0.1,
        edges = PART.sets['Coarse_edges'].edges,
        size = horizontalCoarseSeeding)
PART.seedEdgeByNumber( # Fine edges
        constraint = FINER,
        edges = PART.sets['Fine_edges'].edges,
        number = horizontalFineSeeding)
layerIx = 0
for i in range(len(levels)-1):
        if(i == bottomPileLayer): # If the bottom pile level coincide with one of the soil layers,
             bottomPile is -1, and the if statement will never run
                layerIx -= 1
        d = abs(levels[i+1] - levels[i])
        s = ((E[layerIx] / (2*(1+nu[layerIx])*rho[layerIx]))**0.5)/(maxFreqOfInterest*
             verticalMeshFactor) # v_s/(f_max*8)
        if(s>d): # If the max length is more than the layer depth
                nr = 1
        else:
                nr = d // s + 1 # Makes the layer fullfill the length requirement
        PART.seedEdgeByNumber( # Vertical edges
                constraint = FINER,
                edges = PART.sets['Vertical_edges_' + str(i+1)].edges,
                number = int(nr))
        layerIx += 1


# Set element type
PART.setElementType(
        elemTypes = (ElemType( # Hed
                                        elemCode = C3D8R,
                                        elemLibrary = STANDARD,
                                        secondOrderAccuracy = OFF,
                                        kinematicSplit = AVERAGE_STRAIN,
                                        hourglassControl = DEFAULT,
                                        distortionControl = DEFAULT),
                                ElemType( # Wedge
                                        elemCode = C3D6,
                                        elemLibrary = STANDARD),
                                ElemType( # Tet
                                        elemCode = C3D4,
                                        elemLibrary = STANDARD)),
        regions = Region(
                cells = PART.cells))


# Set mesh control
PART.setMeshControls(
        technique = SWEEP,
        algorithm = MEDIAL_AXIS,
        regions = PART.cells)


# Mesh part
PART.generateMesh()

# Make node set for base boundary condition
PART.Set(
        name = 'Base_nodes',
        nodes = PART.nodes.getByBoundingBox(
                xMin = -xWidth, xMax = xWidth,
                yMin = -yWidth, yMax = yWidth,
                zMin = levels[-1] - 0.001, zMax = levels[-1] + 0.001))
```

```
# Make node sets for MPC tie constraints
cornerNodes = PART.nodes.getByBoundingBox( # To get the height of the node levels
        xMin = -0.5*xWidth - 0.01, xMax = -0.5*xWidth + 0.01,
        yMin = -0.5*yWidth  -0.01, yMax = -0.5*yWidth + 0.01,
        zMin = levels[-1] + 0.01, zMax = mudline)
for i in range(len(cornerNodes)): # Go through all heights
        z = cornerNodes[i].coordinates[2]
        PART.Set(
                name = 'level_' + str(i+1) + '_slave_nodes', # Note that this will not be sorted
                    according to z-coordinates
                nodes = (PART.nodes.getByBoundingBox(
                                        xMin = -0.5*xWidth+0.01, xMax = 0.5*xWidth,
                                        yMin = -0.5*yWidth, yMax = 0.5*yWidth,
                                        zMin = z-0.01, zMax = z+0.01),
                                PART.nodes.getByBoundingSphere( # Corner 4
                                        center = (-0.5*xWidth, 0.5*yWidth, z),
                                        radius = 0)))
        PART.Set(
                name = 'level_' + str(i+1) + '_master_node',
                nodes = PART.nodes.getByBoundingSphere(
                        center = (-0.5*xWidth, -0.5*yWidth,z),
                        radius = 0))

PART.Set(
        name = 'Side_1',
        faces = PART.faces.getByBoundingBox(
                xMin = -xWidth,
                yMin = -0.5*yWidth - 0.001,
                zMin = levels[-1] - 0.001,
                xMax = xWidth,
                yMax = -0.5*yWidth + 0.001,
                zMax = mudline + 0.001))
PART.Set(
        name = 'Side_2',
        faces = PART.faces.getByBoundingBox(
                xMin = -xWidth,
                yMin = 0.5*yWidth - 0.001,
                zMin = levels[-1] - 0.001,
                xMax = xWidth,
                yMax = 0.5*yWidth + 0.001,
                zMax = mudline + 0.001))
PART.SetByBoolean( # All sides
        name = 'Sides',
        sets = (PART.sets['Side_1'],
                        PART.sets['Side_2']))
```

Listing 8: `assemble_OWT_only.py`

```
execfile('jacket.py')
execfile('transitionPiece.py')
execfile('tower.py')


MODEL.rootAssembly.DatumCsysByDefault(CARTESIAN)
ASSEMBLY = MODEL.rootAssembly

# Make instances
ASSEMBLY.Instance(
        name = 'Jacket',
        part = MODEL.parts['Jacket'],
        dependent = ON)
ASSEMBLY.Instance(
        name = 'TP',
        part = MODEL.parts['Transition_piece'],
        dependent = ON)
ASSEMBLY.Instance(
        name = 'Tower',
        part = MODEL.parts['Tower'],
        dependent = ON)
JACKET = ASSEMBLY.instances['Jacket']
TP = ASSEMBLY.instances['TP']
TOWER = ASSEMBLY.instances['Tower']


# Tie the parts together
MODEL.Tie( # Tie jacket to TP. Jacket master
        name = 'Jacet_to_TP',
        master = JACKET.sets['Upper_connection_set'],
        slave = TP.sets['Lower_connection_set'])
MODEL.Tie( # Tie TP to tower. TP master
        name = 'TP_to_Tower',
        master = TP.sets['Upper_connection_set'],
        slave = TOWER.sets['Bottom_node'])


# Set view
```

```
VIEW = session.viewports['Viewport: 1']
VIEW.setValues(
        displayedObject = ASSEMBLY)
VIEW.assemblyDisplay.setValues(
        renderBeamProfiles = ON,
        mesh = OFF,
        renderStyle = SHADED)
VIEW.assemblyDisplay.geometryOptions.setValues(
        datumPoints = OFF,
        datumAxes = OFF,
        datumPlanes = OFF,
        referencePointLabels = OFF,
        referencePointSymbols = OFF,
        datumCoordSystems = ON)
session.View( # Set User-1 view
        name = 'User-1',
        nearPlane = 314.55,
        farPlane = 766.22,
        width = 527.27,
        height = 252.55,
        projection = PERSPECTIVE,
        cameraPosition = (-293.47, -427.39, 149.41),
        cameraUpVector = (0.34203, 0.50761, 0.79079),
        cameraTarget = (71.143, 83.475, -57.634),
        viewOffsetX = 77.216,
        viewOffsetY = -3.5872,
        autoFit = OFF)
VIEW.view.setValues(session.views['User-1']) # Set current camera view
VIEW.view.setRotationCenter(
        rotationCenter = (0.0,0.0,0.0))


# Set colors:
CMAP = VIEW.colorMappings['Section']
CMAP.updateOverrides(
    overrides =
    {'Jacket_section_Hbars':(True, '#FFD700','Default', '#FFD700'),
    'Jacket_section_bracings_level_1':(True,'#FFD700', 'Default', '#FFD700'),
    'Jacket_section_bracings_level_2':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_bracings_level_3':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_bracings_level_4':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_0':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_1':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_2':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_3':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_4':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_B':(True, '#FFD700', 'Default', '#FFD700'),
    'Jacket_section_legs_level_T':(True, '#FFD700', 'Default', '#FFD700'),
    'Pile_section_level_1':(True, '#999999', 'Default', '#999999'),
    'Pile_section_level_2':(True, '#999999', 'Default', '#999999'),
    'Pile_section_level_3':(True, '#999999', 'Default', '#999999'),
    'Pile_section_level_4':(True, '#999999', 'Default', '#999999'),
    'Pile_section_level_5':(True, '#999999', 'Default', '#999999'),
    'TP_section_BM_leg':(True, '#FFD700', 'Default', '#FFD700'),
    'TP_section_B_leg':(True, '#FFD700', 'Default', '#FFD700'),
    'TP_section_MT_leg': (True, '#FFD700', 'Default', '#FFD700'),
    'TP_section_T_leg': (True, '#FFD700', 'Default', '#FFD700'),
    'TP_section_tower_piece': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_1': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_2': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_3': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_4': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_5': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_6': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_7': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_8': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_9': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_10': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_11': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_12': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_13': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_14': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_15': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_16': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_17': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_18': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_19': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_20': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_21': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_22': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_23': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_24': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_25': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_26': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_27': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_28': (True, '#CCCCCC', 'Default', '#CCCCCC'),
```

```
    'Tower_section_29': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_30': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_31': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_32': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_33': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_34': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_35': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_36': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_37': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_38': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_39': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_40': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_41': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_42': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_43': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_44': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_45': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_46': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_47': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_48': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_49': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_50': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_51': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_52': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_53': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_54': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_55': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_56': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_57': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_58': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_59': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_60': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_61': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_62': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_63': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_64': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_65': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_66': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_67': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_68': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_69': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_70': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_71': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_72': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_73': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_74': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_75': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_76': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_77': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_78': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_79': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_80': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_81': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_82': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_83': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_84': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_85': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_86': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_87': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_88': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_89': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_90': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_91': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_92': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_93': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_94': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_95': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_96': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_97': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_98': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_99': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_100': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_101': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_102': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_103': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_104': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_105': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_106': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_107': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_108': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_109': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_110': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_111': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_112': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_113': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_114': (True, '#CCCCCC', 'Default', '#CCCCCC'),
```

```
    'Tower_section_115': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_116': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_117': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_118': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_119': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_120': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_121': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_122': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_123': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_124': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_125': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_126': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_127': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_128': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_129': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_130': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_131': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_132': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_133': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_134': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_135': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_136': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_137': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_138': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_139': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_140': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_141': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_142': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_143': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_144': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_145': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_146': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_147': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_148': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_149': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_150': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_151': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_152': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_153': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_154': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_155': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_156': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_157': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_158': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_159': (True, '#CCCCCC', 'Default', '#CCCCCC'),
    'Tower_section_160': (True, '#CCCCCC', 'Default', '#CCCCCC')})
VIEW.setColor(colorMapping = CMAP)
```

Listing 9: `assemble_soil_and_piles_stiffness_analysis.py`

```
execfile('soil.py')
execfile('piles.py')

# Creates an analysis for getting stiffnesses at pile top. Does a unit displacement/rotation in for
    each DOF

MODEL.rootAssembly.DatumCsysByDefault(CARTESIAN)
ASSEMBLY = MODEL.rootAssembly

# Make instances
ASSEMBLY.Instance(
        name = 'Piles',
        part = MODEL.parts['Piles'],
        dependent = ON)
ASSEMBLY.Instance(
        name = 'Soil',
        part = MODEL.parts['Soil'],
        dependent = ON)

SOIL = ASSEMBLY.instances['Soil']
PILES = ASSEMBLY.instances['Piles']

# Make steps
MODEL.StaticStep(
                name = 'U1',
                previous = 'Initial')
for i in range(1,6): # Six DOFS
        MODEL.StaticStep(
                name = 'U'+str(i+1),
                previous = 'U'+str(i))

# Tie piles to soil
MODEL.Tie( # Tie piles to soil. Pile as master
        name = 'Piles_to_soil',
        master = PILES.sets['Soil_connection_nodes'],
        slave = SOIL.sets['Hole_surfaces'],
```

```
        adjust = OFF,
        positionToleranceMethod = SPECIFIED,
        positionTolerance = holeRadius*2) # holeRadius from soil.py. *2 to be sure every node finds a
              connection


# Set boundary conditions
MODEL.EncastreBC( # Fixed base
        name = 'Fixed_base',
        createStepName = 'Initial',
        region = SOIL.sets['Base'])
MODEL.EncastreBC( # Fixed outer planes
        name = 'Fixed_outer_planes',
        createStepName = 'Initial',
        region = SOIL.sets['All_sides'])


# Set unit displacements/rotations
displ = ((1.0, 0.0, 0.0, 0.0, 0.0, 0.0),
                 (0.0, 1.0, 0.0, 0.0, 0.0, 0.0),
                 (0.0, 0.0, 1.0, 0.0, 0.0, 0.0),
                 (0.0, 0.0, 0.0, 1.0, 0.0, 0.0),
                 (0.0, 0.0, 0.0, 0.0, 1.0, 0.0),
                 (0.0, 0.0, 0.0, 0.0, 0.0, 1.0))
for i in range(6): # Six different unit displacements
        MODEL.DisplacementBC(
                name = 'U' + str(i+1),
                createStepName = 'U' + str(i+1),
                region = PILES.sets['Top_connection'],
                u1 = displ[i][0], u2 = displ[i][1], u3 = displ[i][2],
                ur1 = displ[i][3], ur2 = displ[i][4], ur3 = displ[i][5])
        if(i<5): # Not on last
                MODEL.boundaryConditions['U'+str(i+1)].deactivate('U'+str(i+2))


# Create and run job
mdb.Job(
        name = 'Get_stiffness',
        model = 'Model-1',
        description = 'Unit displacement/rotation of each DOF at top of piles',
        numCpus = 32,
        numDomains = 32)
```

Listing 10: `ODB_get_stiffness.py`

```
####### Create stiffness matrix from odb #######

# Is to be run after getting the soil and pile unit displacement results

# Specify node labels
n1 = 8 # Node label of top node pile 1
n2 = 14 # Node label of top node pile 2
n3 = 2 # Node label of top node pile 3
n4 = 20 # Node label of top node pile 4

nodeLabels = (n1, n2, n3, n4)
ODB = session.odbs['Get_stiffness.odb']

k = [] # Holding all stiffness matrices
for i in range(4): # 4 Piles
        kP = [] # Stiffness matrix for each pile
        DOFix = 0
        PILE = nodeLabels[i]-1
        for j in range(6): # 6 DOFs
                row = [] # Holding one row at a time of the stiffness matrix of each pile
                if(j < 3):
                        DOFtype = 'RF'
                else:
                        DOFtype = 'RM'
                if(DOFix == 3): # Reset DOF index
                        DOFix = 0
                for l in range(6): # 6 Steps (unit displacements)
                        STEP = 'U' + str(l+1)
                        r = ODB.steps[STEP].frames[-1].fieldOutputs[DOFtype].values[PILE].data[DOFix]
                        row.append(r)
                kP.append(row)
                DOFix += 1
        k.append(kP)

# Make an avarage
cutoff = 1E4 # Specify values with an absolute value less than this to be set as zero in the average
     matrix

kAvg = []
for i in range(6): # Row
        row = []
```

```
        for j in range(6): # Column
                n = (k[0][i][j] + k[1][i][j] + k[2][i][j] + k[3][i][j])/4
                if(abs(n)<cutoff):
                        n = 0
                row.append(n)
        kAvg.append(row)

outputFile = open('ODB_stiffness_output.txt','w')

for i in range(4): # Piles
        outputFile.write('
             ____________________________________________________________________________\n')
        outputFile.write('Pile {}:\n'.format(i+1))
        for j in range(6): # Row
                for l in range(6): # Column
                        outputFile.write('{:15.5E}'.format(k[i][j][l]))
                outputFile.write('\n')
        outputFile.write('\n')

outputFile.write('____________________________________________________________________________\n')
outputFile.write('Average:\n')
for i in range(6): # Row
        for j in range(6): # Column
                outputFile.write('{:15.5E}'.format(kAvg[i][j]))
        outputFile.write('\n')

outputFile.close()
print('Done: Stiffness matrix extracted and saved to ODB_stiffness_output.txt')
```

Listing 11: `assemble_soil_and_piles.py`

```
execfile('soil.py')
execfile('piles.py')


MODEL.rootAssembly.DatumCsysByDefault(CARTESIAN)
ASSEMBLY = MODEL.rootAssembly

# Make instances
ASSEMBLY.Instance(
        name = 'Piles',
        part = MODEL.parts['Piles'],
        dependent = ON)
ASSEMBLY.Instance(
        name = 'Soil',
        part = MODEL.parts['Soil'],
        dependent = ON)

SOIL = ASSEMBLY.instances['Soil']
PILES = ASSEMBLY.instances['Piles']

# Tie piles to soil
MODEL.Tie( # Tie piles to soil. Pile as master
        name = 'Piles_to_soil',
        master = PILES.sets['Soil_connection_nodes'],
        slave = SOIL.sets['Hole_surfaces'],
        adjust = OFF,
        positionToleranceMethod = SPECIFIED,
        positionTolerance = holeRadius*2) # holeRadius from soil.py. *2 to be sure every node finds a
              connection

# Tie soil layers (each mesh layer outer nodes are tied together)
for i in range(len(cornerNodes)):
        MODEL.MultipointConstraint(
                name = 'MPCtie_' + str(i+1),
                controlPoint = SOIL.sets['level_' + str(i+1) + '_master_node'], # Master
                surface = SOIL.sets['level_' + str(i+1) + '_slave_nodes'], # Slave
                mpcType = TIE_MPC)
```

Listing 12: `assemble_soil_slice.py`

```
execfile('soilSlice.py')

MODEL.rootAssembly.DatumCsysByDefault(CARTESIAN)
ASSEMBLY = MODEL.rootAssembly

# Make instances
ASSEMBLY.Instance(
        name = 'Soil_slice',
        part = MODEL.parts['Soil_slice'],
        dependent = ON)
SOIL = ASSEMBLY.instances['Soil_slice']
```

```
# Set boundary conditions
MODEL.PinnedBC(
        name = 'Pinned_base',
        createStepName = 'Initial',
        region = SOIL.sets['Base_nodes'])

MODEL.DisplacementBC(
        amplitude = UNSET,
        createStepName = 'Initial',
    distributionType = UNIFORM,
    fieldName = '',
    localCsys = None,
    name = 'Hold_sides',
    region = SOIL.sets['Sides'],
    u1 = UNSET,
    u2 = SET,
    u3 = UNSET,
    ur1 = UNSET,
    ur2 = UNSET,
    ur3 = UNSET)

# Tie
for i in range(len(cornerNodes)):
        MODEL.MultipointConstraint(
                name = 'MPCtie_' + str(i+1),
                controlPoint = SOIL.sets['level_' + str(i+1) + '_master_node'], # Master
                surface = SOIL.sets['level_' + str(i+1) + '_slave_nodes'], # Slave
                mpcType = TIE_MPC)
```

Listing 13: `substructuring.py`

```
execfile('soil.py')
execfile('piles.py')

# Makes a model ready for creating a substructure of the soil and piles


MODEL.rootAssembly.DatumCsysByDefault(CARTESIAN)
ASSEMBLY = MODEL.rootAssembly

# Make instances
ASSEMBLY.Instance(
        name = 'Piles',
        part = MODEL.parts['Piles'],
        dependent = ON)
ASSEMBLY.Instance(
        name = 'Soil',
        part = MODEL.parts['Soil'],
        dependent = ON)

SOIL = ASSEMBLY.instances['Soil']
PILES = ASSEMBLY.instances['Piles']

# Tie piles to soil
MODEL.Tie( # Tie piles to soil. Pile as master
        name = 'Piles_to_soil',
        master = PILES.sets['Soil_connection_nodes'],
        slave = SOIL.sets['Hole_surfaces'],
        adjust = OFF,
        positionToleranceMethod = SPECIFIED,
        positionTolerance = holeRadius*2) # holeRadius from soil.py. *2 to be sure every node finds a
              connection

# Tie soil layers (each mesh layer outer nodes are tied together)
for i in range(len(cornerNodes)):
        MODEL.MultipointConstraint(
                name = 'MPCtie_' + str(i+1),
                controlPoint = SOIL.sets['level_' + str(i+1) + '_master_node'], # Master
                surface = SOIL.sets['level_' + str(i+1) + '_slave_nodes'], # Slave
                mpcType = TIE_MPC)

# Set boundary conditions
MODEL.PinnedBC( # Pinned base
        name = 'Pinned_base',
        createStepName = 'Initial',
        region = SOIL.sets['Base_nodes'])

mdb.models.changeKey(
        fromName = 'Model-1',
        toName = 'Substructure_generation')
MODEL = mdb.models['Substructure_generation']
ASSEMBLY = MODEL.rootAssembly
SOIL = ASSEMBLY.instances['Soil']
PILES = ASSEMBLY.instances['Piles']

MODEL.SubstructureGenerateStep(
```

```
        name = 'Generate_substructure',
        previous = 'Initial',
        recoveryMatrix = NONE,
        substructureIdentifier = 1)

MODEL.RetainedNodalDofsBC(
        createStepName = 'Generate_substructure',
        name = 'Retained_DOFs',
        region = PILES.sets['Top_connection'],
        u1 = ON,
        u2 = ON,
        u3 = ON,
        ur1 = ON,
        ur2 = ON,
        ur3 = ON)

mdb.Job(
        atTime = None,
        contactPrint = OFF,
        description = '',
        echoPrint = OFF,
        explicitPrecision = SINGLE,
        getMemoryFromAnalysis = True,
        historyPrint = OFF,
        memory = 90,
        memoryUnits = PERCENTAGE,
        model = 'Substructure_generation',
        modelPrint = OFF,
        multiprocessingMode = DEFAULT,
        name = 'Substr_generation',
    nodalOutputPrecision = SINGLE,
    numCpus = 32,
    numDomains = 32,
    numGPUs = 0,
    queue = None,
    resultsFormat = ODB,
    scratch = '',
    type = ANALYSIS,
    userSubroutine = '',
    waitHours = 0,
    waitMinutes = 0)
```

Listing 14: `towerData.csv`

```
26
131.63
4.15
2.75
0.07
0.03
160
Tower_steel
2.1E+11
0.3
8500
0.158963354
0.005616645
866555.056
2.4001665900E+08
1.4210211500E+08
1.1184641280E+08
```

Listing 15: `transitionPieceData.csv`

```
nodeData - format: [0]nodeNR,[1]x,[2]y,[3]z
1,0,0,18
2,0,0,26
3,-7,-7,18
4,-6.38931297709924,-6.38931297709924,22
5,-2.93449314192417,-2.93449314192417,26
6,7,-7,18
7,6.38931297709924,-6.38931297709924,22
8,2.93449314192417,-2.93449314192417,26
9,7,7,18
10,6.38931297709924,6.38931297709924,22
11,2.93449314192417,2.93449314192417,26
12,-7,7,18
13,-6.38931297709924,6.38931297709924,22
14,-2.93449314192417,2.93449314192417,26
beamData - format: [0]beamNr,[1]node1Nr,[2]node2Nr,[3]beamName,[4]beamSection
1,1,2,Tower_beam,TP_section_tower_piece
2,1,3,Beam_B_1,TP_section_B_leg
3,3,4,Beam_BM_1,TP_section_BM_leg
4,4,5,Beam_MT_1,TP_section_MT_leg
```

```
5,5,2,Beam_T_1,TP_section_T_leg
6,1,6,Beam_B_2,TP_section_B_leg
7,6,7,Beam_BM_2,TP_section_BM_leg
8,7,8,Beam_MT_2,TP_section_MT_leg
9,8,2,Beam_T_2,TP_section_T_leg
10,1,9,Beam_B_3,TP_section_B_leg
11,9,10,Beam_BM_3,TP_section_BM_leg
12,10,11,Beam_MT_3,TP_section_MT_leg
13,11,2,Beam_T_3,TP_section_T_leg
14,1,12,Beam_B_4,TP_section_B_leg
15,12,13,Beam_BM_4,TP_section_BM_leg
16,13,14,Beam_MT_4,TP_section_MT_leg
17,14,2,Beam_T_4,TP_section_T_leg
sectionData - format: [0]sectionName,[1]profileRadius,[2]profileWallThickness,[3]sectionSeeding
TP_section_MT_leg,0.7,0.08,1
TP_section_BM_leg,0.7,0.08,1
TP_section_B_leg,0.7,0.08,1
TP_section_T_leg,0.7,0.08,1
TP_section_tower_piece,4.15,0.07,1
Material data:
TP_steel
1.05E+12
0.3
7850
0.158963354
0.005616645
```

Listing 16: `jacketData.csv`

```
nodeData - format: [0]nodeNR,[1]x,[2]y,[3]z
1,-17,-17,-48.5
2,-17,-17,-47.5
3,-16.8036641221374,-16.8036641221374,-46.214
4,-13.5838167938931,-13.5838167938931,-25.124
5,-10.9822900763359,-10.9822900763359,-8.084
6,-8.88152671755725,-8.88152671755725,5.676
7,-7.18320610687023,-7.18320610687023,16.8
8,-7,-7,18
9,17,-17,-48.5
10,17,-17,-47.5
11,16.8036641221374,-16.8036641221374,-46.214
12,13.5838167938931,-13.5838167938931,-25.124
13,10.9822900763359,-10.9822900763359,-8.084
14,8.88152671755725,-8.88152671755725,5.676
15,7.18320610687023,-7.18320610687023,16.8
16,7,-7,18
17,17,17,-48.5
18,17,17,-47.5
19,16.8036641221374,16.8036641221374,-46.214
20,13.5838167938931,13.5838167938931,-25.124
21,10.9822900763359,10.9822900763359,-8.084
22,8.88152671755725,8.88152671755725,5.676
23,7.18320610687023,7.18320610687023,16.8
24,7,7,18
25,-17,17,-48.5
26,-17,17,-47.5
27,-16.8036641221374,16.8036641221374,-46.214
28,-13.5838167938931,13.5838167938931,-25.124
29,-10.9822900763359,10.9822900763359,-8.084
30,-8.88152671755725,8.88152671755725,5.676
31,-7.18320610687023,7.18320610687023,16.8
32,-7,7,18
33,0,-15.02315348429,-34.5516553220993
34,0,-12.1453038662081,-15.7017403236632
35,0,-9.82082182341998,-0.476382943400867
36,0,-7.94259545466944,11.8259997719152
37,15.02315348429,0,-34.5516553220993
38,12.1453038662081,0,-15.7017403236632
39,9.82082182341998,0,-0.476382943400867
40,7.94259545466944,0,11.8259997719152
41,0,15.02315348429,-34.5516553220993
42,0,12.1453038662081,-15.7017403236632
43,0,9.82082182341998,-0.476382943400867
44,0,7.94259545466944,11.8259997719152
45,-15.02315348429,0,-34.5516553220993
46,-12.1453038662081,0,-15.7017403236632
47,-9.82082182341998,0,-0.476382943400867
48,-7.94259545466944,0,11.8259997719152
beamData - format: [0]beamNr,[1]node1Nr,[2]node2Nr,[3]beamName,[4]beamSection
1,1,2,Leg_1_B,Jacket_section_legs_level_B
2,2,3,Leg_1_0,Jacket_section_legs_level_0
3,3,4,Leg_1_1,Jacket_section_legs_level_1
4,4,5,Leg_1_2,Jacket_section_legs_level_2
5,5,6,Leg_1_3,Jacket_section_legs_level_3
6,6,7,Leg_1_4,Jacket_section_legs_level_4
7,7,8,Leg_1_T,Jacket_section_legs_level_T
```

```
8,9,10,Leg_2_B,Jacket_section_legs_level_B
9,10,11,Leg_2_0,Jacket_section_legs_level_0
10,11,12,Leg_2_1,Jacket_section_legs_level_1
11,12,13,Leg_2_2,Jacket_section_legs_level_2
12,13,14,Leg_2_3,Jacket_section_legs_level_3
13,14,15,Leg_2_4,Jacket_section_legs_level_4
14,15,16,Leg_2_T,Jacket_section_legs_level_T
15,17,18,Leg_3_B,Jacket_section_legs_level_B
16,18,19,Leg_3_0,Jacket_section_legs_level_0
17,19,20,Leg_3_1,Jacket_section_legs_level_1
18,20,21,Leg_3_2,Jacket_section_legs_level_2
19,21,22,Leg_3_3,Jacket_section_legs_level_3
20,22,23,Leg_3_4,Jacket_section_legs_level_4
21,23,24,Leg_3_T,Jacket_section_legs_level_T
22,25,26,Leg_4_B,Jacket_section_legs_level_B
23,26,27,Leg_4_0,Jacket_section_legs_level_0
24,27,28,Leg_4_1,Jacket_section_legs_level_1
25,28,29,Leg_4_2,Jacket_section_legs_level_2
26,29,30,Leg_4_3,Jacket_section_legs_level_3
27,30,31,Leg_4_4,Jacket_section_legs_level_4
28,31,32,Leg_4_T,Jacket_section_legs_level_T
29,3,33,Brace_side_1_1,Jacket_section_bracings_level_1
30,4,33,Brace_side_1_2,Jacket_section_bracings_level_1
31,4,34,Brace_side_1_3,Jacket_section_bracings_level_2
32,5,34,Brace_side_1_4,Jacket_section_bracings_level_2
33,5,35,Brace_side_1_5,Jacket_section_bracings_level_3
34,6,35,Brace_side_1_6,Jacket_section_bracings_level_3
35,6,36,Brace_side_1_7,Jacket_section_bracings_level_4
36,7,36,Brace_side_1_8,Jacket_section_bracings_level_4
37,11,33,Brace_side_1_9,Jacket_section_bracings_level_1
38,12,33,Brace_side_1_10,Jacket_section_bracings_level_1
39,12,34,Brace_side_1_11,Jacket_section_bracings_level_2
40,13,34,Brace_side_1_12,Jacket_section_bracings_level_2
41,13,35,Brace_side_1_13,Jacket_section_bracings_level_3
42,14,35,Brace_side_1_14,Jacket_section_bracings_level_3
43,14,36,Brace_side_1_15,Jacket_section_bracings_level_4
44,15,36,Brace_side_1_16,Jacket_section_bracings_level_4
45,11,37,Brace_side_2_1,Jacket_section_bracings_level_1
46,12,37,Brace_side_2_2,Jacket_section_bracings_level_1
47,12,38,Brace_side_2_3,Jacket_section_bracings_level_2
48,13,38,Brace_side_2_4,Jacket_section_bracings_level_2
49,13,39,Brace_side_2_5,Jacket_section_bracings_level_3
50,14,39,Brace_side_2_6,Jacket_section_bracings_level_3
51,14,40,Brace_side_2_7,Jacket_section_bracings_level_4
52,15,40,Brace_side_2_8,Jacket_section_bracings_level_4
53,19,37,Brace_side_2_9,Jacket_section_bracings_level_1
54,20,37,Brace_side_2_10,Jacket_section_bracings_level_1
55,20,38,Brace_side_2_11,Jacket_section_bracings_level_2
56,21,38,Brace_side_2_12,Jacket_section_bracings_level_2
57,21,39,Brace_side_2_13,Jacket_section_bracings_level_3
58,22,39,Brace_side_2_14,Jacket_section_bracings_level_3
59,22,40,Brace_side_2_15,Jacket_section_bracings_level_4
60,23,40,Brace_side_2_16,Jacket_section_bracings_level_4
61,19,41,Brace_side_3_1,Jacket_section_bracings_level_1
62,20,41,Brace_side_3_2,Jacket_section_bracings_level_1
63,20,42,Brace_side_3_3,Jacket_section_bracings_level_2
64,21,42,Brace_side_3_4,Jacket_section_bracings_level_2
65,21,43,Brace_side_3_5,Jacket_section_bracings_level_3
66,22,43,Brace_side_3_6,Jacket_section_bracings_level_3
67,22,44,Brace_side_3_7,Jacket_section_bracings_level_4
68,23,44,Brace_side_3_8,Jacket_section_bracings_level_4
69,27,41,Brace_side_3_9,Jacket_section_bracings_level_1
70,28,41,Brace_side_3_10,Jacket_section_bracings_level_1
71,28,42,Brace_side_3_11,Jacket_section_bracings_level_2
72,29,42,Brace_side_3_12,Jacket_section_bracings_level_2
73,29,43,Brace_side_3_13,Jacket_section_bracings_level_3
74,30,43,Brace_side_3_14,Jacket_section_bracings_level_3
75,30,44,Brace_side_3_15,Jacket_section_bracings_level_4
76,31,44,Brace_side_3_16,Jacket_section_bracings_level_4
77,27,45,Brace_side_4_1,Jacket_section_bracings_level_1
78,28,45,Brace_side_4_2,Jacket_section_bracings_level_1
79,28,46,Brace_side_4_3,Jacket_section_bracings_level_2
80,29,46,Brace_side_4_4,Jacket_section_bracings_level_2
81,29,47,Brace_side_4_5,Jacket_section_bracings_level_3
82,30,47,Brace_side_4_6,Jacket_section_bracings_level_3
83,30,48,Brace_side_4_7,Jacket_section_bracings_level_4
84,31,48,Brace_side_4_8,Jacket_section_bracings_level_4
85,3,45,Brace_side_4_9,Jacket_section_bracings_level_1
86,4,45,Brace_side_4_10,Jacket_section_bracings_level_1
87,4,46,Brace_side_4_11,Jacket_section_bracings_level_2
88,5,46,Brace_side_4_12,Jacket_section_bracings_level_2
89,5,47,Brace_side_4_13,Jacket_section_bracings_level_3
90,6,47,Brace_side_4_14,Jacket_section_bracings_level_3
91,6,48,Brace_side_4_15,Jacket_section_bracings_level_4
92,7,48,Brace_side_4_16,Jacket_section_bracings_level_4
93,3,11,Hbrace_side_1,Jacket_section_Hbars
```

```
94,11,19,Hbrace_side_2,Jacket_section_Hbars
95,19,27,Hbrace_side_3,Jacket_section_Hbars
96,27,3,Hbrace_side_4,Jacket_section_Hbars
sectionData - format: [0]sectionName,[1]profileRadius,[2]profileWallThickness,[3]sectionSeeding
Jacket_section_legs_level_T,0.7,0.066,2
Jacket_section_legs_level_4,0.7,0.042,10
Jacket_section_legs_level_3,0.7,0.042,10
Jacket_section_legs_level_2,0.7,0.042,10
Jacket_section_legs_level_1,0.7,0.07,10
Jacket_section_legs_level_0,0.7,0.12,2
Jacket_section_legs_level_B,0.7,0.12,2
Jacket_section_Hbars,0.52,0.02,10
Jacket_section_bracings_level_4,0.416,0.016,5
Jacket_section_bracings_level_3,0.42,0.02,5
Jacket_section_bracings_level_2,0.468,0.018,5
Jacket_section_bracings_level_1,0.53,0.03,5
Material data:
Jacket_steel
2.1E+11
0.3
7850
0.158963354
0.005616645
```

Listing 17: `pileData.csv`

```
-92,-86,-70,-65,-59,-48.5
Pile_steel
2.1E+11
0.3
7850
0
0
34
sectionData - format: [0]sectionName,[1]profileRadius,[2]profileWallThickness,[3]sectionSeeding
Pile_section_level_1,1.219,0.03,10
Pile_section_level_2,1.219,0.028,10
Pile_section_level_3,1.219,0.036,10
Pile_section_level_4,1.219,0.05,10
Pile_section_level_5,1.219,0.052,14
```

Listing 18: `soilData.csv`

```
34
42
1.219
200
200
5
5
3
4
6
8
0.429488496
0.003026692
-50
layerData - format: [0]z-coordinate,[1]rho,[2]E,[3]nu,[4]C
-52,1936.79918450561,34383444.8741728,0.3,1800
-55.5,1936.79918450561,66583254.691456,0.3,6750
-59,1936.79918450561,92580258.6422224,0.3,13050
-60,1936.79918450561,105976894.526331,0.3,17100
-65,1936.79918450561,121563835.155407,0.3,22500
-70,2038.73598369011,191742229.88206,0.3,32000
-72.5,2089.70438328236,241074400.512351,0.3,39625
-79,2140.67278287462,299770806.591885,0.3,49400
-84,2140.67278287462,335993118.855382,0.3,62050
-88,2140.67278287462,361820322.687293,0.3,71950
-90,2140.67278287462,378059509.297743,0.3,78550
-100,2140.67278287462,408606272.458314,0.3,91750
-110,2140.67278287462,454982682.559079,0.3,113750
-120,2140.67278287462,497050698.586205,0.3,135750
-130,2140.67278287462,535826046.867277,0.3,157750
-140,2140.67278287462,571978765.373149,0.3,179750
```

D OpenFAST input files

D.1 Main input file

```
------- OpenFAST MAIN INPUT FILE -------------------------------------------
10MW IEA RWT on INNWIND jacket made by Daniel Martens Pedersen
---------------------- SIMULATION CONTROL ----------------------------------
FALSE         Echo           - Echo input data to <RootName>.ech (flag)
"FATAL"       AbortLevel     - Error level when simulation should abort (string) {"
    WARNING", "SEVERE", "FATAL"}
        130   TMax           - Total run time (s)
      0.005   DT             - Recommended module time step (s)
          2   InterpOrder    - Interpolation order for input/output time history (-)
              {1=linear, 2=quadratic}
          0   NumCrctn       - Number of correction iterations (-) {0= explicit
              calculation, i.e., no corrections}
      99999   DT_UJac        - Time between calls to get Jacobians (s)
      1E+06   UJacSclFact    - Scaling factor used in Jacobians (-)
---------------------- FEATURE SWITCHES AND FLAGS --------------------------
          1   CompElast      - Compute structural dynamics (switch) {1=ElastoDyn; 2=
              ElastoDyn + BeamDyn for blades}
          1   CompInflow     - Compute inflow wind velocities (switch) {0=still air; 1=
              InflowWind; 2=external from OpenFOAM}
          2   CompAero       - Compute aerodynamic loads (switch) {0=None; 1=AeroDyn v
              14; 2=AeroDyn v15}
          1   CompServo      - Compute control and electrical-drive dynamics (switch)
              {0=None; 1=ServoDyn}
          1   CompHydro      - Compute hydrodynamic loads (switch) {0=None; 1=HydroDyn}
          1   CompSub        - Compute sub-structural dynamics (switch) {0=None; 1=
              SubDyn; 2=External Platform MCKF}
          0   CompMooring    - Compute mooring system (switch) {0=None; 1=MAP++; 2=
              FEAMooring; 3=MoorDyn; 4=OrcaFlex}
          0   CompIce        - Compute ice loads (switch) {0=None; 1=IceFlow; 2=IceDyn}
---------------------- INPUT FILES -----------------------------------------
"10MW_ElastoDyn.dat"    EDFile          - Name of file containing ElastoDyn input
    parameters (quoted string)
"unused"    BDBldFile(1)    - Name of file containing BeamDyn input parameters for blade
     1 (quoted string)
"unused"    BDBldFile(2)    - Name of file containing BeamDyn input parameters for blade
     2 (quoted string)
"unused"    BDBldFile(3)    - Name of file containing BeamDyn input parameters for blade
     3 (quoted string)
"10MW_InflowFile.dat"    InflowFile      - Name of file containing inflow wind input
    parameters (quoted string)
"10MW_AeroDyn15.dat"    AeroFile        - Name of file containing aerodynamic input
    parameters (quoted string)
"10MW_ServoDyn.dat"    ServoFile       - Name of file containing control and electrical-
    drive input parameters (quoted string)
"Jacket_HydroDyn-Kopi.dat"    HydroFile       - Name of file containing hydrodynamic
    input parameters (quoted string)
"Jacket_SubDyn-Kopi.dat"    SubFile         - Name of file containing sub-structural
    input parameters (quoted string)
"unused"      MooringFile    - Name of file containing mooring system input parameters
    (quoted string)
"unused"      IceFile        - Name of file containing ice input parameters (quoted
    string)
---------------------- OUTPUT ----------------------------------------------
True          SumPrint       - Print summary data to "<RootName>.sum" (flag)
          1   SttsTime       - Amount of time between screen status messages (s)
      99999   ChkptTime      - Amount of time between creating checkpoint files for
          potential restart (s)
      0.005   DT_Out         - Time step for tabular output (s) (or "default")
          0   TStart         - Time to begin tabular output (s)
          1   OutFileFmt     - Format for tabular (time-marching) output file (switch)
              {0: uncompressed binary [<RootName>.outb], 1: text file [<RootName>.out],
              2: binary file [<RootName>.outb], 3: both 1 and 2}
True          TabDelim       - Use tab delimiters in text tabular output file? (flag) {
    uses spaces if false}
"ES20.12E3"   OutFmt         - Format used for text tabular output, excluding the time
    channel.  Resulting field should be 10 characters. (quoted string)
---------------------- LINEARIZATION ---------------------------------------
```

```
False         Linearize       - Linearization analysis (flag)
False         CalcSteady      - Calculate a steady-state periodic operating point before
     linearization? [unused if Linearize=False] (flag)
          3   TrimCase        - Controller parameter to be trimmed {1:yaw; 2:torque; 3:
              pitch} [used only if CalcSteady=True] (-)
     0.0001   TrimTol         - Tolerance for the rotational speed convergence [used
         only if CalcSteady=True] (-)
      0.001   TrimGain        - Proportional gain for the rotational speed error (>0) [
          used only if CalcSteady=True] (rad/(rad/s) for yaw or pitch; Nm/(rad/s) for
          torque)
          0   Twr_Kdmp        - Damping factor for the tower [used only if CalcSteady=
              True] (N/(m/s))
          0   Bld_Kdmp        - Damping factor for the blades [used only if CalcSteady=
              True] (N/(m/s))
          1   NLinTimes       - Number of times to linearize (-) [>=1] [unused if
              Linearize=False]
         60   LinTimes        - List of times at which to linearize (s) [1 to NLinTimes]
              [used only when Linearize=True and CalcSteady=False]
          1   LinInputs       - Inputs included in linearization (switch) {0=none; 1=
              standard; 2=all module inputs (debug)} [unused if Linearize=False]
          1   LinOutputs      - Outputs included in linearization (switch) {0=none; 1=
              from OutList(s); 2=all module outputs (debug)} [unused if Linearize=False]
False         LinOutJac       - Include full Jacobians in linearization output (for
    debug) (flag) [unused if Linearize=False; used only if LinInputs=LinOutputs=2]
False         LinOutMod       - Write module-level linearization output files in
    addition to output for full system? (flag) [unused if Linearize=False]
---------------------- VISUALIZATION ------------------------------------------
          0   WrVTK           - VTK visualization data output: (switch) {0=none; 1=
              initialization data only; 2=animation; 3=mode shapes}
          2   VTK_type        - Type of VTK visualization data: (switch) {1=surfaces; 2=
              basic meshes (lines/points); 3=all meshes (debug)} [unused if WrVTK=0]
true          VTK_fields      - Write mesh fields to VTK data files? (flag) {true/false}
     [unused if WrVTK=0]
         25   VTK_fps         - Frame rate for VTK output (frames per second){will use
             closest integer multiple of DT} [used only if WrVTK=2 or WrVTK=3]
```

D.2 AeroDyn input file

```
------- AERODYN v15.03.* INPUT FILE ----------------------------------------
IEA-10.0-198 Reference Wind Turbine aerodynamic input properties
======  General Options  ===================================================
False                 Echo        - Echo the input to "<rootname>.AD.ech"?  (flag)
0.005                 DTAero      - Time interval for aerodynamic calculations {or "
    default"} (s)
1                     WakeMod     - Type of wake/induction model (switch) {0=none, 1=
    BEMT, 2=DBEMT, 3=OLAF} [WakeMod cannot be 2 or 3 when linearizing]
2                     AFAeroMod   - Type of blade airfoil aerodynamics model (switch)
    {1=steady model, 2=Beddoes-Leishman unsteady model} [AFAeroMod must be 1 when
    linearizing]
1                     TwrPotent   - Type tower influence on wind based on potential
    flow around the tower (switch) {0=none, 1=baseline potential flow, 2=potential flow
    with Bak correction}
0                     TwrShadow   - Calculate tower influence on wind based on
    downstream tower shadow (switch) {0=none, 1=Powles model, 2=Eames model}
True                  TwrAero     - Calculate tower aerodynamic loads? (flag)
False                 FrozenWake  - Assume frozen wake during linearization? (flag) [
    used only when WakeMod=1 and when linearizing]
False                 CavitCheck  - Perform cavitation check? (flag) [AFAeroMod must be
     1 when CavitCheck=true]
False                 CompAA      - Flag to compute AeroAcoustics calculation [only
    used when WakeMod=1 or 2]
AeroAcousticsInput.dat AA_InputFile - AeroAcoustics input file [used only when CompAA=
    true]
======  Environmental Conditions  ==========================================
1.225                             AirDens     - Air density (kg/m^3)
1.464e-05                         KinVisc     - Kinematic air viscosity (m^2/s)
340.3                             SpdSound    - Speed of sound (m/s)
101325                            Patm        - Atmospheric pressure (Pa) [used only
    when CavitCheck=True]
1700                              Pvap        - Vapour pressure of fluid (Pa) [used
    only when CavitCheck=True]
```

```
0.5                                        FluidDepth  - Water depth above mid-hub
    height (m) [used only when CavitCheck=True]
======  Blade-Element/Momentum Theory Options  == [used only when WakeMod=1]
2                       SkewMod     - Type of skewed-wake correction model (switch) {1=
    uncoupled, 2=Pitt/Peters, 3=coupled} [unused when WakeMod=0 or 3]
"default"                       SkewModFactor - Constant used in Pitt/Peters skewed wake
    model {or "default" is 15/32*pi} (-) [used only when SkewMod=2; unused when WakeMod
    =0 or 3]
True                    TipLoss     - Use the Prandtl tip-loss model? (flag) [unused when
     WakeMod=0 or 3]
True                    HubLoss     - Use the Prandtl hub-loss model? (flag) [unused when
     WakeMod=0 or 3]
True                    TanInd      - Include tangential induction in BEMT calculations?
    (flag) [unused when WakeMod=0 or 3]
True                    AIDrag      - Include the drag term in the axial-induction
    calculation? (flag) [unused when WakeMod=0 or 3]
True                    TIDrag      - Include the drag term in the tangential-induction
    calculation? (flag) [unused when WakeMod=0,3 or TanInd=FALSE]
"default"               IndToler    - Convergence tolerance for BEMT nonlinear solve
    residual equation {or "default"} (-) [unused when WakeMod=0 or 3]
100                     MaxIter     - Maximum number of iteration steps (-) [unused when
    WakeMod=0]
======  Dynamic Blade-Element/Momentum Theory Options  ====== [used only when WakeMod=1]
1                       DBEMT_Mod   - Type of dynamic BEMT (DBEMT) model {1=constant tau
    1, 2=time-dependent tau1} (-) [used only when WakeMod=2]
20                      tau1_const  - Time constant for DBEMT (s) [used only when WakeMod
    =2 and DBEMT_Mod=1]
======  OLAF -- cOnvecting LAgrangian Filaments (Free Vortex Wake) Theory Options
    ================== [used only when WakeMod=3]
IEA-10.0-198-RWT_OLAF.dat OLAFInputFileName - Input file for OLAF [used only when
    WakeMod=3]
======  Beddoes-Leishman Unsteady Airfoil Aerodynamics Options  ============ [used only
    when AFAeroMod=2]
3                       UAMod       - Unsteady Aero Model Switch (switch) {1=Baseline
    model (Original), 2=Gonzalez's variant (changes in Cn,Cc,Cm), 3=Minnema/Pierce
    variant (changes in Cc and Cm)} [used only when AFAeroMod=2]
True                    FLookup     - Flag to indicate whether a lookup for f' will be
    calculated (TRUE) or whether best-fit exponential equations will be used (FALSE); if
     FALSE S1-S4 must be provided in airfoil input files (flag) [used only when
    AFAeroMod=2]
======  Airfoil Information =========================================
1                       AFTabMod    - Interpolation method for multiple airfoil tables
    {1=1D interpolation on AoA (first table only); 2=2D interpolation on AoA and Re; 3=2
    D interpolation on AoA and UserProp} (-)
1                       InCol_Alfa  - The column in the airfoil tables that contains the
    angle of attack (-)
2                       InCol_Cl    - The column in the airfoil tables that contains the
    lift coefficient (-)
3                       InCol_Cd    - The column in the airfoil tables that contains the
    drag coefficient (-)
4                       InCol_Cm    - The column in the airfoil tables that contains the
    pitching-moment coefficient; use zero if there is no Cm column (-)
0                       InCol_Cpmin - The column in the airfoil tables that contains the
    Cpmin coefficient; use zero if there is no Cpmin column (-)
30                      NumAFfiles  - Number of airfoil files used (-)
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_00.dat"    AFNames            - Airfoil
    file names (NumAFfiles lines) (quoted strings)
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_01.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_02.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_03.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_04.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_05.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_06.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_07.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_08.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_09.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_10.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_11.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_12.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_13.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_14.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_15.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_16.dat"
```

```
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_17.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_18.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_19.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_20.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_21.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_22.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_23.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_24.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_25.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_26.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_27.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_28.dat"
"../Airfoils/IEA-10.0-198-RWT_AeroDyn15_Polar_29.dat"
======  Rotor/Blade Properties  ==============================================
True                   UseBlCm     - Include aerodynamic pitching moment in calculations
    ?  (flag)
"10MW_AeroDyn15_blade.dat" ADBlFile(1) - Name of file containing distributed aerodynamic
     properties for Blade #1 (-)
"10MW_AeroDyn15_blade.dat" ADBlFile(2) - Name of file containing distributed aerodynamic
     properties for Blade #2 (-) [unused if NumBl < 2]
"10MW_AeroDyn15_blade.dat" ADBlFile(3) - Name of file containing distributed aerodynamic
     properties for Blade #3 (-) [unused if NumBl < 3]
======  Tower Influence and Aerodynamics ======= [used only when TwrPotent/=0, TwrShadow
    /=0, or TwrAero=True]
11                    NumTwrNds   - Number of tower nodes used in the analysis  (-) [
    used only when TwrPotent/=0, TwrShadow/=0, or TwrAero=True]
TwrElev        TwrDiam        TwrCd         TwrTI (used only with TwrShadow=2)
(m)              (m)           (-)            (-)
26.00              8.300                  0.5                0.1
36.51              8.020                  0.5                0.1
47.01              7.740                  0.5                0.1
57.52              7.460                  0.5                0.1
68.02              7.190                  0.5                0.1
78.53              6.910                  0.5                0.1
89.03              6.630                  0.5                0.1
99.54              6.350                  0.5                0.1
110.04         6.070              0.5                0.1
120.55         5.790              0.5                0.1
131.63         5.500              0.5                0.1
======  Outputs  ============================================================
True                   SumPrint    - Generate a summary file listing input options and
    interpolated properties to "<rootname>.AD.sum"?  (flag)
9                      NBlOuts     - Number of blade node outputs [0 - 9] (-)
4, 7, 10, 13, 15, 18, 21, 24, 27 BlOutNd     - Blade nodes whose values will be output
    (-)
9                      NTwOuts     - Number of tower node outputs [0 - 9]  (-)
1, 2, 3, 4, 5, 6, 7, 8, 9                    TwOutNd     - Tower nodes whose values
    will be output  (-)
                   OutList             - The next line(s) contains a list of output
                       parameters.  See OutListParameters.xlsx for a listing of
                       available output channels, (-)
END of input file (the word "END" must appear in the first 3 columns of this last
    OutList line)
---------------------------------------------------------------------------
```

D.3 AeroDyn blade input file

```
------- AERODYN v15.00.* BLADE DEFINITION INPUT FILE -----------------------
Generated with AeroElasticSE FAST driver
======  Blade Properties =================================================
30          NumBlNds    - Number of blade nodes used in the analysis (-)
    BlSpn        BlCrvAC        BlSwpAC        BlCrvAng       BlTwist        BlChord
                 BlAFID
     (m)           (m)            (m)             (deg)          (deg)            (m)
                      (-)
 0.000000000000000e+00  0.000000000000000e+00  0.000000000000000e+00 -2.841341264563080e
     -01  1.200002806121996e+01  4.600000000000000e+00        1
 3.336162423461167e+00 -1.654421527553980e-02  0.000000000000000e+00 -3.377268435011394e
     -01  1.199964011912278e+01  4.602856719303900e+00        2
 6.672940546967894e+00 -3.933305588585017e-02  0.000000000000000e+00 -4.210207359220099e
     -01  1.199671430769378e+01  4.722267657550535e+00        3
```

```
 1.000900612310667e+01 -6.557715114405413e-02  0.000000000000000e+00 -4.651652887327193e
     -01  1.202241256026795e+01  5.008074089590719e+00        4
 1.334580407523511e+01 -9.350721003134796e-02  0.000000000000000e+00 -4.830551455935536e
     -01  1.156869232147211e+01  5.411514106583073e+00        5
 1.668232154052844e+01 -1.218385579937304e-01  0.000000000000000e+00 -4.877219150971492e
     -01  1.003883227305816e+01  5.800776847290640e+00        6
 2.001847591680350e+01 -1.503066639299398e-01  0.000000000000000e+00 -4.938541126732519e
     -01  8.076609005387132e+00  6.015708470169677e+00        7
 2.335448768472906e+01 -1.793477832512315e-01  0.000000000000000e+00 -5.017469868135497e
     -01  6.583589890889979e+00  5.982229064039409e+00        8
 2.669106852484301e+01 -2.087387242480996e-01  0.000000000000000e+00 -5.126107923364652e
     -01  5.661206923495265e+00  5.827155062245235e+00        9
 3.002744069380972e+01 -2.390482031574574e-01  0.000000000000000e+00 -5.391128507636488e
     -01  5.010257468512085e+00  5.608710832987120e+00       10
 3.336423513674197e+01 -2.715275862068965e-01  0.000000000000000e+00 -5.942422772358332e
     -01  4.447057061811484e+00  5.345638585017836e+00       11
 3.670036540293389e+01 -3.082550962088016e-01  0.000000000000000e+00 -6.817513298831223e
     -01  3.930526877993271e+00  5.053029053152982e+00       12
 4.003665517241379e+01 -3.509195402298850e-01  0.000000000000000e+00 -7.966951995003160e
     -01  3.440409048815762e+00  4.745833333333334e+00       13
 4.337350887089437e+01 -4.010418615230644e-01  0.000000000000000e+00 -9.417872586551786e
     -01  2.937699901176175e+00  4.434493039759803e+00       14
 4.670987073478967e+01 -4.606041704278749e-01  0.000000000000000e+00 -1.133138363624353e
     +00  2.390684243696569e+00  4.125517078895109e+00       15
 5.004619223459137e+01 -5.329988867770838e-01  0.000000000000000e+00 -1.385754310382543e
     +00  1.800014974504586e+00  3.822383518870486e+00       16
 5.338224074576885e+01 -6.219662141370261e-01  0.000000000000000e+00 -1.695430119589656e
     +00  1.181890536278555e+00  3.529777853199384e+00       17
 5.671861940572266e+01 -7.304127842993395e-01  0.000000000000000e+00 -2.059721193813486e
     +00  5.507512667346930e-01  3.252281841526046e+00       18
 6.005534136056972e+01 -8.618052848575714e-01  0.000000000000000e+00 -2.497308875779408e
     +00 -7.688715647492510e-02  2.991705734632684e+00       19
 6.339116532297232e+01 -1.021152044681885e+00  0.000000000000000e+00 -3.005546348335876e
     +00 -6.869854123077352e-01  2.748427838756678e+00       20
 6.672788765294773e+01 -1.211664404894328e+00  0.000000000000000e+00 -3.591966938519594e
     +00 -1.262800892452079e+00  2.523873414905450e+00       21
 7.006452203065135e+01 -1.439241794380588e+00  0.000000000000000e+00 -4.299034625217793e
     +00 -1.792359064289039e+00  2.317451053639846e+00       22
 7.340058382486387e+01 -1.711862556715063e+00  0.000000000000000e+00 -5.171394237073920e
     +00 -2.262491400035471e+00  2.128054548548095e+00       23
 7.673649718106726e+01 -2.040622669463747e+00  0.000000000000000e+00 -6.228703097646942e
     +00 -2.647842454204785e+00  1.957865831912941e+00       24
 8.007344200626959e+01 -2.435850156739812e+00  0.000000000000000e+00 -7.439338711743959e
     +00 -2.942933952928573e+00  1.802591379310345e+00       25
 8.340979293189226e+01 -2.904657231085948e+00  0.000000000000000e+00 -8.994870020906601e
     +00 -3.129337812859181e+00  1.660044089895351e+00       26
 8.674615298152030e+01 -3.479102038483521e+00  0.000000000000000e+00 -1.103253271356095e
     +01 -3.135739832811044e+00  1.521990683939798e+00       27
 9.008278302387268e+01 -4.181642891246685e+00  0.000000000000000e+00 -1.374480804418788e
     +01 -2.863327401352850e+00  1.342706976127321e+00       28
 9.341876858237546e+01 -5.064501302681990e+00  0.000000000000000e+00 -1.766379073748617e
     +01 -2.046041506422669e+00  1.050590574712644e+00       29
 9.675500000000000e+01 -6.206200000000000e+00  0.000000000000000e+00 -2.001170345264212e
     +01 -3.724225668350351e-02  9.619999999999999e-02       30
```

D.4 ElastoDyn input file

```
------- ELASTODYN v1.03.* INPUT FILE -----------------------------------------
Generated with AeroElasticSE FAST driver
----------------------- SIMULATION CONTROL -----------------------------------
False                   Echo        - Echo input data to "<RootName>.ech" (flag)
3                       Method      - Integration method: {1: RK4, 2: AB4, or 3: ABM4}
    (-)
0.005                   DT          - Integration time step (s)
----------------------- ENVIRONMENTAL CONDITION ------------------------------
9.81                    Gravity     - Gravitational acceleration (m/s^2)
----------------------- DEGREES OF FREEDOM -----------------------------------
True                    FlapDOF1    - First flapwise blade mode DOF (flag)
True                    FlapDOF2    - Second flapwise blade mode DOF (flag)
True                    EdgeDOF     - First edgewise blade mode DOF (flag)
```

```
False                   TeetDOF     - Rotor-teeter DOF (flag) [unused for 3 blades]
True                    DrTrDOF     - Drivetrain rotational-flexibility DOF (flag)
True                    GenDOF      - Generator DOF (flag)
True                    YawDOF      - Yaw DOF (flag)
True                    TwFADOF1    - First fore-aft tower bending-mode DOF (flag)
True                    TwFADOF2    - Second fore-aft tower bending-mode DOF (flag)
True                    TwSSDOF1    - First side-to-side tower bending-mode DOF (flag)
True                    TwSSDOF2    - Second side-to-side tower bending-mode DOF (flag)
True                    PtfmSgDOF   - Platform horizontal surge translation DOF (flag)
True                    PtfmSwDOF   - Platform horizontal sway translation DOF (flag)
True                    PtfmHvDOF   - Platform vertical heave translation DOF (flag)
True                    PtfmRDOF    - Platform roll tilt rotation DOF (flag)
True                    PtfmPDOF    - Platform pitch tilt rotation DOF (flag)
True                    PtfmYDOF    - Platform yaw rotation DOF (flag)
---------------------- INITIAL CONDITIONS ------------------------------------
0.0                     OoPDefl     - Initial out-of-plane blade-tip displacement (meters
    )
0.0                     IPDefl      - Initial in-plane blade-tip deflection (meters)
0.0                     BlPitch(1)  - Blade 1 initial pitch (degrees)
0.0                     BlPitch(2)  - Blade 2 initial pitch (degrees)
0.0                     BlPitch(3)  - Blade 3 initial pitch (degrees) [unused for 2
    blades]
0.0                     TeetDefl    - Initial or fixed teeter angle (degrees) [unused for
     3 blades]
0.0                     Azimuth     - Initial azimuth angle for blade 1 (degrees)
6.0                     RotSpeed    - Initial or fixed rotor speed (rpm)
0.0                     NacYaw      - Initial or fixed nacelle-yaw angle (degrees)
-0.225                  TTDspFA     - Initial fore-aft tower-top displacement (meters)
0.0                     TTDspSS     - Initial side-to-side tower-top displacement (meters
    )
-0.006                  PtfmSurge   - Initial or fixed horizontal surge translational
    displacement of platform (meters)
0.0                     PtfmSway    - Initial or fixed horizontal sway translational
    displacement of platform (meters)
-0.0156                 PtfmHeave   - Initial or fixed vertical heave translational
    displacement of platform (meters)
0.0                     PtfmRoll    - Initial or fixed roll tilt rotational displacement
    of platform (degrees)
0.0                     PtfmPitch   - Initial or fixed pitch tilt rotational displacement
     of platform (degrees)
0.0                     PtfmYaw     - Initial or fixed yaw rotational displacement of
    platform (degrees)
---------------------- TURBINE CONFIGURATION ---------------------------------
3                       NumBl       - Number of blades (-)
99.055                  TipRad      - The distance from the rotor apex to the blade tip (
    meters)
2.3                     HubRad      - The distance from the rotor apex to the blade root
    (meters)
-4.0                    PreCone(1)  - Blade 1 cone angle (degrees)
-4.0                    PreCone(2)  - Blade 2 cone angle (degrees)
-4.0                    PreCone(3)  - Blade 3 cone angle (degrees) [unused for 2 blades]
0.349                   HubCM       - Distance from rotor apex to hub mass [positive
    downwind] (meters)
0.0                     UndSling    - Undersling length [distance from teeter pin to the
    rotor apex] (meters) [unused for 3 blades]
0.0                     Delta3      - Delta-3 angle for teetering rotors (degrees) [
    unused for 3 blades]
0.0                     AzimB1Up    - Azimuth value to use for I/O when blade 1 points up
     (degrees)
-10.039                 OverHang    - Distance from yaw axis to rotor apex [3 blades] or
    teeter pin [2 blades] (meters)
3.55                    ShftGagL    - Distance from rotor apex [3 blades] or teeter pin
    [2 blades] to shaft strain gages [positive for upwind rotors] (meters)
-6.0                    ShftTilt    - Rotor shaft tilt angle (degrees)
-4.84                   NacCMxn     - Downwind distance from the tower-top to the nacelle
     CM (meters)
0.0                     NacCMyn     - Lateral  distance from the tower-top to the nacelle
     CM (meters)
3.39                    NacCMzn     - Vertical distance from the tower-top to the nacelle
     CM (meters)
-3.09528                NcIMUxn     - Downwind distance from the tower-top to the nacelle
     IMU (meters)
0.0                     NcIMUyn     - Lateral  distance from the tower-top to the nacelle
```

```
    IMU (meters)
2.23336               NcIMUzn     - Vertical distance from the tower-top to the nacelle
    IMU (meters)
2.96                  Twr2Shft    - Vertical distance from the tower-top to the rotor
    shaft (meters)
131.63                TowerHt     - Height of tower above ground level [onshore] or MSL
    [offshore] (meters)
26.0                  TowerBsHt   - Height of tower base above ground level [onshore]
    or MSL [offshore] (meters)
0.0                   PtfmCMxt    - Downwind distance from the ground level [onshore]
    or MSL [offshore] to the platform CM (meters)
0.0                   PtfmCMyt    - Lateral distance from the ground level [onshore] or
    MSL [offshore] to the platform CM (meters)
26.0                  PtfmCMzt    - Vertical distance from the ground level [onshore]
    or MSL [offshore] to the platform CM (meters)
26.0                  PtfmRefzt   - Vertical distance from the ground level [onshore]
    or MSL [offshore] to the platform reference point (meters)
---------------------- MASS AND INERTIA ----------------------------------
0.0                   TipMass(1)  - Tip-brake mass, blade 1 (kg)
0.0                   TipMass(2)  - Tip-brake mass, blade 2 (kg)
0.0                   TipMass(3)  - Tip-brake mass, blade 3 (kg) [unused for 2 blades]
81707.0               HubMass     - Hub mass (kg)
476512.0              HubIner     - Hub inertia about rotor axis [3 blades] or teeter
    axis [2 blades] (kg m^2)
3801700.0             GenIner     - Generator inertia about HSS (kg m^2)
545623.0              NacMass     - Nacelle mass (kg)
19006380.6            NacYIner    - Nacelle inertia about yaw axis (kg m^2)
93457.0               YawBrMass   - Yaw bearing mass (kg)
0.0                   PtfmMass    - Platform mass (kg)
0.0                   PtfmRIner   - Platform inertia for roll tilt rotation about the
    platform CM (kg m^2)
0.0                   PtfmPIner   - Platform inertia for pitch tilt rotation about the
    platform CM (kg m^2)
4.0513389E+07         PtfmYIner   - Platform inertia for yaw rotation about the
    platform CM (kg m^2)
---------------------- BLADE ---------------------------------------------
50                    BldNodes    - Number of blade nodes (per blade) used for analysis
    (-)
"10MW_ElastoDyn_blade.dat" BldFile1    - Name of file containing properties for blade 1
    (quoted string)
"10MW_ElastoDyn_blade.dat" BldFile2    - Name of file containing properties for blade 2
    (quoted string)
"10MW_ElastoDyn_blade.dat" BldFile3    - Name of file containing properties for blade 3
    (quoted string) [unused for 2 blades]
---------------------- ROTOR-TEETER --------------------------------------
0                     TeetMod     - Rotor-teeter spring/damper model {0: none, 1:
    standard, 2: user-defined from routine UserTeet} (switch) [unused for 3 blades]
0.0                   TeetDmpP    - Rotor-teeter damper position (degrees) [used only
    for 2 blades and when TeetMod=1]
0.0                   TeetDmp     - Rotor-teeter damping constant (N-m/(rad/s)) [used
    only for 2 blades and when TeetMod=1]
0.0                   TeetCDmp    - Rotor-teeter rate-independent Coulomb-damping
    moment (N-m) [used only for 2 blades and when TeetMod=1]
0.0                   TeetHStP    - Rotor-teeter hard-stop position (degrees) [used
    only for 2 blades and when TeetMod=1]
0.0                   TeetSStP    - Rotor-teeter soft-stop position (degrees) [used
    only for 2 blades and when TeetMod=1]
0.0                   TeetSSSp    - Rotor-teeter soft-stop linear-spring constant (N-m/
    rad) [used only for 2 blades and when TeetMod=1]
0.0                   TeetHSSp    - Rotor-teeter hard-stop linear-spring constant (N-m/
    rad) [used only for 2 blades and when TeetMod=1]
---------------------- DRIVETRAIN ----------------------------------------
100.0                 GBoxEff     - Gearbox efficiency (%)
1.0                   GBRatio     - Gearbox ratio (-)
2317025000.0          DTTorSpr    - Drivetrain torsional spring (N-m/rad)
9240560.0             DTTorDmp    - Drivetrain torsional damper (N-m/(rad/s))
---------------------- FURLING -------------------------------------------
False                 Furling     - Read in additional model properties for furling
    turbine (flag) [must currently be FALSE)
"unused"              FurlFile    - Name of file containing furling properties (quoted
    string) [unused when Furling=False]
---------------------- TOWER ---------------------------------------------
50                    TwrNodes    - Number of tower nodes used for analysis (-)
```

```
"10MW_ElastoDyn_tower.dat" TwrFile     - Name of file containing tower properties (
    quoted string)
---------------------- OUTPUT --------------------------------------------------
True                   SumPrint    - Print summary data to "<RootName>.sum" (flag)
1                      OutFile     - Switch to determine where output will be placed:
    {1: in module output file only; 2: in glue code output file only; 3: both} (
    currently unused)
True                   TabDelim    - Use tab delimiters in text tabular output file? (
    flag) (currently unused)
"ES20.12E3"             OutFmt      - Format used for text tabular output (except time).
      Resulting field should be 10 characters. (quoted string) (currently unused)
0.0                    TStart      - Time to begin tabular output (s) (currently unused)
1                      DecFact     - Decimation factor for tabular output {1: output
    every time step} (-) (currently unused)
9                      NTwGages    - Number of tower nodes that have strain gages for
    output [0 to 9] (-)
5, 10, 15, 20, 25, 30, 35, 40, 50                       TwrGagNd    - List of tower
    nodes that have strain gages [1 to TwrNodes] (-) [unused if NTwGages=0]
0                      NBlGages    - Number of blade nodes that have strain gages for
    output [0 to 9] (-)
6, 11, 16, 21, 26, 30  BldGagNd    - List of blade nodes that have strain gages [1 to
    BldNodes] (-) [unused if NBlGages=0]
                   OutList             - The next line(s) contains a list of output
                       parameters.  See OutListParameters.xlsx for a listing of
                       available output channels, (-)
END of input file (the word "END" must appear in the first 3 columns of this last
    OutList line)
---------------------------------------------------------------------------------
```

D.5 ElastoDyn blade input file

```
------- ELASTODYN V1.00.* TOWER INPUT FILE ---------------------------------
Generated with AeroElasticSE FAST driver
---------------------- TOWER PARAMETERS ------------------------------------
11                     NTwInpSt    - Number of input stations to specify tower geometry
35.5                   TwrFADmp(1) - Tower 1st fore-aft mode structural damping ratio (%
    )
35.5                   TwrFADmp(2) - Tower 2nd fore-aft mode structural damping ratio (%
    )
35.5                   TwrSSDmp(1) - Tower 1st side-to-side mode structural damping
    ratio (%)
35.5                   TwrSSDmp(2) - Tower 2nd side-to-side mode structural damping
    ratio (%)
---------------------- TOWER ADJUSTMUNT FACTORS ----------------------------
1.0                    FAStTunr(1) - Tower fore-aft modal stiffness tuner, 1st mode (-)
1.0                    FAStTunr(2) - Tower fore-aft modal stiffness tuner, 2nd mode (-)
1.0                    SSStTunr(1) - Tower side-to-side stiffness tuner, 1st mode (-)
1.0                    SSStTunr(2) - Tower side-to-side stiffness tuner, 2nd mode (-)
1.0                    AdjTwMa     - Factor to adjust tower mass density (-)
1.0                    AdjFASt     - Factor to adjust tower fore-aft stiffness (-)
1.0                    AdjSSSt     - Factor to adjust tower side-to-side stiffness (-)
---------------------- DISTRIBUTED TOWER PROPERTIES ------------------------
  HtFract       TMassDen         TwFAStif       TwSSStif
   (-)           (kg/m)           (Nm^2)         (Nm^2)
  0.000         15383.91            3.2182E+12     3.2182E+12
  0.099         14860.52            2.9008E+12     2.9008E+12
  0.199         13321.73            2.4236E+12     2.4236E+12
  0.298         11856.37            2.0052E+12     2.0052E+12
  0.398         11423.77            1.7936E+12     1.7936E+12
  0.497         10067.90            1.4611E+12     1.4611E+12
  0.597         8785.46         1.1748E+12         1.1748E+12
  0.696         7576.46         9.3018E+11         9.3018E+11
  0.796         5640.45         6.3444E+11         6.3444E+11
  0.895         4614.37         4.7280E+11         4.7280E+11
  1.000         4382.05         4.0493E+11         4.0493E+11
---------------------- TOWER FORE-AFT MODE SHAPES --------------------------
    -0.4466   TwFAM1Sh(2) - Mode 1, coefficient of x^2 term
     6.4341   TwFAM1Sh(3) -       , coefficient of x^3 term
   -11.8014   TwFAM1Sh(4) -       , coefficient of x^4 term
    10.1836   TwFAM1Sh(5) -       , coefficient of x^5 term
```

```
    -3.3697   TwFAM1Sh(6) -       , coefficient of x^6 term
     0.9048   TwFAM2Sh(2) - Mode 2, coefficient of x^2 term
    -3.9416   TwFAM2Sh(3) -       , coefficient of x^3 term
     9.2654   TwFAM2Sh(4) -       , coefficient of x^4 term
    -7.7438   TwFAM2Sh(5) -       , coefficient of x^5 term
     2.5152   TwFAM2Sh(6) -       , coefficient of x^6 term
---------------------- TOWER SIDE-TO-SIDE MODE SHAPES ----------------------
    -0.4382   TwSSM1Sh(2) - Mode 1, coefficient of x^2 term
     6.3432   TwSSM1Sh(3) -       , coefficient of x^3 term
   -11.6165   TwSSM1Sh(4) -       , coefficient of x^4 term
    10.0191   TwSSM1Sh(5) -       , coefficient of x^5 term
    -3.3075   TwSSM1Sh(6) -       , coefficient of x^6 term
     0.8872   TwSSM2Sh(2) - Mode 2, coefficient of x^2 term
    -4.5891   TwSSM2Sh(3) -       , coefficient of x^3 term
    10.5693   TwSSM2Sh(4) -       , coefficient of x^4 term
    -8.9942   TwSSM2Sh(5) -       , coefficient of x^5 term
     3.1268   TwSSM2Sh(6) -       , coefficient of x^6 term
```

D.6 InflowWind input file

```
------- InflowWind v3.01.* INPUT FILE -------------------------------------
Generated with AeroElasticSE FAST driver
---------------------------------------------------------------------------
False                  Echo        - Echo input data to <RootName>.ech (flag)
3                      WindType    - switch for wind file type (1=steady; 2=uniform; 3=
    binary TurbSim FF; 4=binary Bladed-style FF; 5=HAWC format; 6=User defined; 7=native
     Bladed FF)
0                      PropagationDir - Direction of wind propagation (meteoroligical
    rotation from aligned with X (positive rotates towards -Y) -- degrees)
0                      VFlowAng    - Upflow angle (degrees) (not used for native Bladed
    format WindType=7)
1                      NWindVel    - Number of points to output the wind velocity    (0
    to 9)
0.0                    WindVxiList - List of coordinates in the inertial X direction (m)
0.0                    WindVyiList - List of coordinates in the inertial Y direction (m)
131.63                 WindVziList - List of coordinates in the inertial Z direction (m)
================== Parameters for Steady Wind Conditions [used only for WindType = 1]
    =========================
12                     HWindSpeed  - Horizontal windspeed                            (m/
    s)
131.63                 RefHt       - Reference height for horizontal wind speed      (m)
0.2                    PLexp       - Power law exponent                              (-)
================== Parameters for Uniform wind file   [used only for WindType = 2]
    ============================
"none"                 Filename_Uni - Filename of time series data for uniform wind
    field.      (-)
119.0                  RefHt_Uni   - Reference height for horizontal wind speed
                   (m)
1.0                    RefLength   - Reference length for linear horizontal and vertical
     sheer (-)
================== Parameters for Binary TurbSim Full-Field files   [used only for
    WindType = 3] ==============
"../kaimal.bts"                 FileName_BTS - Name of the Full field wind file to use
    (.bts)
================== Parameters for Binary Bladed-style Full-Field files   [used only for
    WindType = 4] =========
"none"                 FilenameRoot - Rootname of the full-field wind file to use (.wnd,
     .sum)
False                  TowerFile   - Have tower file (.twr) (flag)
================== Parameters for HAWC-format binary files  [Only used with WindType =
    5] =====================
"none"                 FileName_u  - name of the file containing the u-component
    fluctuating wind (.bin)
"none"                 FileName_v  - name of the file containing the v-component
    fluctuating wind (.bin)
"none"                 FileName_w  - name of the file containing the w-component
    fluctuating wind (.bin)
2                      nx          - number of grids in the x direction (in the 3 files
    above) (-)
2                      ny          - number of grids in the y direction (in the 3 files
    above) (-)
```

```
2                      nz          - number of grids in the z direction (in the 3 files
    above) (-)
10                     dx          - distance (in meters) between points in the x
    direction    (m)
10                     dy          - distance (in meters) between points in the y
    direction    (m)
10                     dz          - distance (in meters) between points in the z
    direction    (m)
0.0                    RefHt_Hawc  - reference height; the height (in meters) of the
    vertical center of the grid (m)
-------------   Scaling parameters for turbulence   -----------------------
0                      ScaleMethod - Turbulence scaling method   [0 = none, 1 = direct
    scaling, 2 = calculate scaling factor based on a desired standard deviation]
1.0                    SFx         - Turbulence scaling factor for the x direction (-)
      [ScaleMethod=1]
1.0                    SFy         - Turbulence scaling factor for the y direction (-)
      [ScaleMethod=1]
1.0                    SFz         - Turbulence scaling factor for the z direction (-)
      [ScaleMethod=1]
1.0                    SigmaFx     - Turbulence standard deviation to calculate scaling
    from in x direction (m/s)    [ScaleMethod=2]
1.0                    SigmaFy     - Turbulence standard deviation to calculate scaling
    from in y direction (m/s)    [ScaleMethod=2]
1.0                    SigmaFz     - Turbulence standard deviation to calculate scaling
    from in z direction (m/s)    [ScaleMethod=2]
-------------   Mean wind profile parameters (added to HAWC-format files) --
0.0                    URef        - Mean u-component wind speed at the reference height
     (m/s)
0                      WindProfile - Wind profile type (0=constant;1=logarithmic,2=power
     law)
0.0                    PLExp_Hawc  - Power law exponent (-) (used for PL wind profile
    type only)
0.0                    Z0          - Surface roughness length (m) (used for LG wind
    profile type only)
0                      XOffset     - Initial offset in +x direction (shift of wind box)
    (-)
===================== OUTPUT ==========================================
False                  SumPrint    - Print summary data to <RootName>.IfW.sum (flag)
                                            OutList      - The next line(s) contains a
                                                list of output parameters.  See
                                                OutListParameters.xlsx for a listing of
                                                available output channels, (-)
END of input file (the word "END" must appear in the first 3 columns of this last
    OutList line)
---------------------------------------------------------------------------
```

D.7 ServoDyn input file

```
------- InflowWind v3.01.* INPUT FILE -------------------------------------
Generated with AeroElasticSE FAST driver
---------------------------------------------------------------------------
False                  Echo        - Echo input data to <RootName>.ech (flag)
3                      WindType    - switch for wind file type (1=steady; 2=uniform; 3=
    binary TurbSim FF; 4=binary Bladed-style FF; 5=HAWC format; 6=User defined; 7=native
     Bladed FF)
0                      PropagationDir - Direction of wind propagation (meteoroligical
    rotation from aligned with X (positive rotates towards -Y) -- degrees)
0                      VFlowAng    - Upflow angle (degrees) (not used for native Bladed
    format WindType=7)
1                      NWindVel    - Number of points to output the wind velocity    (0
    to 9)
0.0                    WindVxiList - List of coordinates in the inertial X direction (m)
0.0                    WindVyiList - List of coordinates in the inertial Y direction (m)
131.63                 WindVziList - List of coordinates in the inertial Z direction (m)
================== Parameters for Steady Wind Conditions [used only for WindType = 1]
    ===
12                     HWindSpeed  - Horizontal windspeed                            (m/
    s)
131.63                 RefHt       - Reference height for horizontal wind speed      (m)
0.2                    PLexp       - Power law exponent                              (-)
```

```
================== Parameters for Uniform wind file   [used only for WindType = 2] ===
"none"                 Filename_Uni - Filename of time series data for uniform wind
    field.      (-)
119.0                  RefHt_Uni   - Reference height for horizontal wind speed
                    (m)
1.0                    RefLength   - Reference length for linear horizontal and vertical
     sheer (-)
================== Parameters for Binary TurbSim Full-Field files   [used only for
    WindType = 3] ==============
"../kaimal.bts"                 FileName_BTS - Name of the Full field wind file to use
    (.bts)
================== Parameters for Binary Bladed-style Full-Field files   [used only for
    WindType = 4] =========
"none"                 FilenameRoot - Rootname of the full-field wind file to use (.wnd,
     .sum)
False                  TowerFile   - Have tower file (.twr) (flag)
================== Parameters for HAWC-format binary files  [Only used with WindType =
    5] =====================
"none"                 FileName_u  - name of the file containing the u-component
    fluctuating wind (.bin)
"none"                 FileName_v  - name of the file containing the v-component
    fluctuating wind (.bin)
"none"                 FileName_w  - name of the file containing the w-component
    fluctuating wind (.bin)
2                      nx          - number of grids in the x direction (in the 3 files
    above) (-)
2                      ny          - number of grids in the y direction (in the 3 files
    above) (-)
2                      nz          - number of grids in the z direction (in the 3 files
    above) (-)
10                     dx          - distance (in meters) between points in the x
    direction    (m)
10                     dy          - distance (in meters) between points in the y
    direction    (m)
10                     dz          - distance (in meters) between points in the z
    direction    (m)
0.0                    RefHt_Hawc  - reference height; the height (in meters) of the
    vertical center of the grid (m)
-------------   Scaling parameters for turbulence   ------------------------
0                      ScaleMethod - Turbulence scaling method   [0 = none, 1 = direct
    scaling, 2 = calculate scaling factor based on a desired standard deviation]
1.0                    SFx         - Turbulence scaling factor for the x direction (-)
      [ScaleMethod=1]
1.0                    SFy         - Turbulence scaling factor for the y direction (-)
      [ScaleMethod=1]
1.0                    SFz         - Turbulence scaling factor for the z direction (-)
      [ScaleMethod=1]
1.0                    SigmaFx     - Turbulence standard deviation to calculate scaling
    from in x direction (m/s)    [ScaleMethod=2]
1.0                    SigmaFy     - Turbulence standard deviation to calculate scaling
    from in y direction (m/s)    [ScaleMethod=2]
1.0                    SigmaFz     - Turbulence standard deviation to calculate scaling
    from in z direction (m/s)    [ScaleMethod=2]
-------------   Mean wind profile parameters (added to HAWC-format files) --
0.0                    URef        - Mean u-component wind speed at the reference height
     (m/s)
0                      WindProfile - Wind profile type (0=constant;1=logarithmic,2=power
     law)
0.0                    PLExp_Hawc  - Power law exponent (-) (used for PL wind profile
    type only)
0.0                    Z0          - Surface roughness length (m) (used for LG wind
    profile type only)
0                      XOffset     - Initial offset in +x direction (shift of wind box)
    (-)
====================== OUTPUT ==========================================
False                  SumPrint    - Print summary data to <RootName>.IfW.sum (flag)
                                         OutList      - The next line(s) contains a
                                             list of output parameters.  See
                                             OutListParameters.xlsx for a listing of
                                             available output channels, (-)
END of input file (the word "END" must appear in the first 3 columns of this last
    OutList line)
---------------------------------------------------------------------------
```

D.7.1 Structural control 1

```
------- STRUCTURAL CONTROL (StC) INPUT FILE --------------------------------
Input file for tuned mass damper, module by Matt Lackner, Meghan Glade, and Semyung Park
     (UMass)
---------------------- SIMULATION CONTROL ---------------------------------
True          Echo         - Echo input data to <RootName>.ech (flag)
---------------------- StC DEGREES OF FREEDOM ------------------------------
          4   StC_DOF_MODE - DOF mode (switch) {0: No StC or TLCD DOF; 1: StC_X_DOF, StC
              _Y_DOF, and/or StC_Z_DOF (three independent StC DOFs); 2: StC_XY_DOF (Omni
              -Directional StC); 3: TLCD; 4: Prescribed force/moment time series}
false         StC_X_DOF    - DOF on or off for StC X (flag) [Used only when StC_DOF_MODE
    =1]
false         StC_Y_DOF    - DOF on or off for StC Y (flag) [Used only when StC_DOF_MODE
    =1]
false         StC_Z_DOF    - DOF on or off for StC Z (flag) [Used only when StC_DOF_MODE
    =1]
---------------------- StC LOCATION ---------------------------------------- [relative
    to the reference origin of component attached to]
   -17.00000   StC_P_X      - At rest X position of StC (m)
    17.00000   StC_P_Y      - At rest Y position of StC (m)
   -74.50000   StC_P_Z      - At rest Z position of StC (m)
---------------------- StC INITIAL CONDITIONS ------------------------------ [used only
     when StC_DOF_MODE=1 or 2]
          0   StC_X_DSP    - StC X initial displacement (m) [relative to at rest
              position]
          0   StC_Y_DSP    - StC Y initial displacement (m) [relative to at rest
              position]
          0   StC_Z_DSP    - StC Z initial displacement (m) [relative to at rest
              position; used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC CONFIGURATION ----------------------------------- [used only
     when StC_DOF_MODE=1 or 2]
          0   StC_X_PSP    - Positive stop position (maximum X mass displacement) (m)
          0   StC_X_NSP    - Negative stop position (minimum X mass displacement) (m)
          0   StC_Y_PSP    - Positive stop position (maximum Y mass displacement) (m)
          0   StC_Y_NSP    - Negative stop position (minimum Y mass displacement) (m)
          0   StC_Z_PSP    - Positive stop position (maximum Z mass displacement) (m) [
              used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
          0   StC_Z_NSP    - Negative stop position (minimum Z mass displacement) (m) [
              used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC MASS, STIFFNESS, & DAMPING ---------------------- [used only
     when StC_DOF_MODE=1 or 2]
          0   StC_X_M      - StC X mass (kg) [must equal StC_Y_M for StC_DOF_MODE = 2]
         50   StC_Y_M      - StC Y mass (kg) [must equal StC_X_M for StC_DOF_MODE = 2]
          0   StC_Z_M      - StC Z mass (kg) [used only when StC_DOF_MODE=1 and StC_Z_
              DOF=TRUE]
          0   StC_XY_M     - StC Z mass (kg) [used only when StC_DOF_MODE=2]
       2300   StC_X_K      - StC X stiffness (N/m)
       2300   StC_Y_K      - StC Y stiffness (N/m)
          0   StC_Z_K      - StC Z stiffness (N/m) [used only when StC_DOF_MODE=1 and
              StC_Z_DOF=TRUE]
         35   StC_X_C      - StC X damping (N/(m/s))
         35   StC_Y_C      - StC Y damping (N/(m/s))
          0   StC_Z_C      - StC Z damping (N/(m/s)) [used only when StC_DOF_MODE=1 and
              StC_Z_DOF=TRUE]
          0   StC_X_KS     - Stop spring X stiffness (N/m)
          0   StC_Y_KS     - Stop spring Y stiffness (N/m)
          0   StC_Z_KS     - Stop spring Z stiffness (N/m) [used only when StC_DOF_MODE
              =1 and StC_Z_DOF=TRUE]
          0   StC_X_CS     - Stop spring X damping (N/(m/s))
          0   StC_Y_CS     - Stop spring Y damping (N/(m/s))
          0   StC_Z_CS     - Stop spring Z damping (N/(m/s)) [used only when StC_DOF_
              MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC USER-DEFINED SPRING FORCES ---------------------- [used only
     when StC_DOF_MODE=1 or 2]
False         Use_F_TBL    - Use spring force from user-defined table (flag)
         17   NKInpSt      - Number of spring force input stations
---------------------- StC SPRING FORCES TABLE ----------------------------- [used only
     when StC_DOF_MODE=1 or 2]
    X                F_X               Y              F_Y              Z              F_
        Z
   (m)               (N)              (m)             (N)             (m)             (N
       )
```

```
-6.0000000E+00  -4.8000000E+06  -6.0000000E+00  -4.8000000E+06  -6.0000000E+00
    -4.8000000E+06
-5.0000000E+00  -2.4000000E+06  -5.0000000E+00  -2.4000000E+06  -5.0000000E+00
    -2.4000000E+06
-4.5000000E+00  -1.2000000E+06  -4.5000000E+00  -1.2000000E+06  -4.5000000E+00
    -1.2000000E+06
-4.0000000E+00  -6.0000000E+05  -4.0000000E+00  -6.0000000E+05  -4.0000000E+00
    -6.0000000E+05
-3.5000000E+00  -3.0000000E+05  -3.5000000E+00  -3.0000000E+05  -3.5000000E+00
    -3.0000000E+05
-3.0000000E+00  -1.5000000E+05  -3.0000000E+00  -1.5000000E+05  -3.0000000E+00
    -1.5000000E+05
-2.5000000E+00  -1.0000000E+05  -2.5000000E+00  -1.0000000E+05  -2.5000000E+00
    -1.0000000E+05
-2.0000000E+00  -6.5000000E+04  -2.0000000E+00  -6.5000000E+04  -2.0000000E+00
    -6.5000000E+04
 0.0000000E+00   0.0000000E+00   0.0000000E+00   0.0000000E+00   0.0000000E+00
     0.0000000E+00
 2.0000000E+00   6.5000000E+04   2.0000000E+00   6.5000000E+04   2.0000000E+00
     6.5000000E+04
 2.5000000E+00   1.0000000E+05   2.5000000E+00   1.0000000E+05   2.5000000E+00
     1.0000000E+05
 3.0000000E+00   1.5000000E+05   3.0000000E+00   1.5000000E+05   3.0000000E+00
     1.5000000E+05
 3.5000000E+00   3.0000000E+05   3.5000000E+00   3.0000000E+05   3.5000000E+00
     3.0000000E+05
 4.0000000E+00   6.0000000E+05   4.0000000E+00   6.0000000E+05   4.0000000E+00
     6.0000000E+05
 4.5000000E+00   1.2000000E+06   4.5000000E+00   1.2000000E+06   4.5000000E+00
     1.2000000E+06
 5.0000000E+00   2.4000000E+06   5.0000000E+00   2.4000000E+06   5.0000000E+00
     2.4000000E+06
 6.0000000E+00   4.8000000E+06   6.0000000E+00   4.8000000E+06   6.0000000E+00
     4.8000000E+06
---------------------- StructCtrl CONTROL ----------------------------------  [used only
     when StC_DOF_MODE=1 or 2]
          0   StC_CMODE     - Control mode (switch) {0:none; 1: Semi-Active Control Mode
              ; 2: Active Control Mode}
          1   StC_SA_MODE   - Semi-Active control mode {1: velocity-based ground hook
              control; 2: Inverse velocity-based ground hook control; 3: displacement-
              based ground hook control 4: Phase difference Algorithm with Friction
              Force 5: Phase difference Algorithm with Damping Force} (-)
          0   StC_X_C_HIGH  - StC X high damping for ground hook control
          0   StC_X_C_LOW   - StC X low damping for ground hook control
          0   StC_Y_C_HIGH  - StC Y high damping for ground hook control
          0   StC_Y_C_LOW   - StC Y low damping for ground hook control
          0   StC_Z_C_HIGH  - StC Z high damping for ground hook control [used only when
               StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
          0   StC_Z_C_LOW   - StC Z low damping for ground hook control  [used only when
               StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
          0   StC_X_C_BRAKE - StC X high damping for braking the StC (Don't use it now.
              should be zero)
          0   StC_Y_C_BRAKE - StC Y high damping for braking the StC (Don't use it now.
              should be zero)
          0   StC_Z_C_BRAKE - StC Z high damping for braking the StC (Don't use it now.
              should be zero) [used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- TLCD ----------------------------------------------  [used only
     when StC_DOF_MODE=3]
   7.9325     L_X             - X TLCD total length (m)
   6.5929     B_X             - X TLCD horizontal length (m)
   2.0217     area_X          - X TLCD cross-sectional area of vertical column (m^2)
   0.913      area_ratio_X    - X TLCD cross-sectional area ratio (vertical column area
       divided by horizontal column area) (-)
   2.5265     headLossCoeff_X - X TLCD head loss coeff (-)
     1000     rho_X           - X TLCD liquid density (kg/m^3)
   3.5767     L_Y             - Y TLCD total length (m)
   2.1788     B_Y             - Y TLCD horizontal length (m)
   1.2252     area_Y          - Y TLCD cross-sectional area of vertical column (m^2)
   2.7232     area_ratio_Y    - Y TLCD cross-sectional area ratio (vertical column area
       divided by horizontal column area) (-)
   0.6433     headLossCoeff_Y - Y TLCD head loss coeff (-)
     1000     rho_Y           - Y TLCD liquid density (kg/m^3)
---------------------- PRESCRIBED TIME SERIES ------------------------------  [used only
```

```
     when StC_DOF_MODE=4]
          1   PrescribedForcesCoord- Prescribed forces are in global or local
              coordinates (switch) {1: global; 2: local}
"seismic_forces_masslessJacket.dat"  PrescribedForcesFile   - Time series force and
    moment (7 columns of time, FX, FY, FZ, MX, MY, MZ)
---------------------------------------------------------------------------
```

D.7.2 Structural control 9

```
------- STRUCTURAL CONTROL (StC) INPUT FILE --------------------------------
Input file for tuned mass damper, module by Matt Lackner, Meghan Glade, and Semyung Park
     (UMass)
---------------------- SIMULATION CONTROL ----------------------------------
True          Echo         - Echo input data to <RootName>.ech (flag)
---------------------- StC DEGREES OF FREEDOM ------------------------------
          4   StC_DOF_MODE - DOF mode (switch) {0: No StC or TLCD DOF; 1: StC_X_DOF, StC
              _Y_DOF, and/or StC_Z_DOF (three independent StC DOFs); 2: StC_XY_DOF (Omni
              -Directional StC); 3: TLCD; 4: Prescribed force/moment time series}
false         StC_X_DOF    - DOF on or off for StC X (flag) [Used only when StC_DOF_MODE
    =1]
false         StC_Y_DOF    - DOF on or off for StC Y (flag) [Used only when StC_DOF_MODE
    =1]
false         StC_Z_DOF    - DOF on or off for StC Z (flag) [Used only when StC_DOF_MODE
    =1]
---------------------- StC LOCATION ----------------------------------------  [relative
    to the reference origin of component attached to]
   17.00000   StC_P_X      - At rest X position of StC (m)
   17.00000   StC_P_Y      - At rest Y position of StC (m)
  -74.50000   StC_P_Z      - At rest Z position of StC (m)
---------------------- StC INITIAL CONDITIONS ------------------------------ [used only
    when StC_DOF_MODE=1 or 2]
          0   StC_X_DSP    - StC X initial displacement (m) [relative to at rest
              position]
          0   StC_Y_DSP    - StC Y initial displacement (m) [relative to at rest
              position]
          0   StC_Z_DSP    - StC Z initial displacement (m) [relative to at rest
              position; used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC CONFIGURATION -----------------------------------  [used only
     when StC_DOF_MODE=1 or 2]
          0   StC_X_PSP    - Positive stop position (maximum X mass displacement) (m)
          0   StC_X_NSP    - Negative stop position (minimum X mass displacement) (m)
          0   StC_Y_PSP    - Positive stop position (maximum Y mass displacement) (m)
          0   StC_Y_NSP    - Negative stop position (minimum Y mass displacement) (m)
          0   StC_Z_PSP    - Positive stop position (maximum Z mass displacement) (m) [
              used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
          0   StC_Z_NSP    - Negative stop position (minimum Z mass displacement) (m) [
              used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC MASS, STIFFNESS, & DAMPING ----------------------  [used only
     when StC_DOF_MODE=1 or 2]
          0   StC_X_M      - StC X mass (kg) [must equal StC_Y_M for StC_DOF_MODE = 2]
         50   StC_Y_M      - StC Y mass (kg) [must equal StC_X_M for StC_DOF_MODE = 2]
          0   StC_Z_M      - StC Z mass (kg) [used only when StC_DOF_MODE=1 and StC_Z_
              DOF=TRUE]
          0   StC_XY_M     - StC Z mass (kg) [used only when StC_DOF_MODE=2]
       2300   StC_X_K      - StC X stiffness (N/m)
       2300   StC_Y_K      - StC Y stiffness (N/m)
          0   StC_Z_K      - StC Z stiffness (N/m) [used only when StC_DOF_MODE=1 and
              StC_Z_DOF=TRUE]
         35   StC_X_C      - StC X damping (N/(m/s))
         35   StC_Y_C      - StC Y damping (N/(m/s))
          0   StC_Z_C      - StC Z damping (N/(m/s)) [used only when StC_DOF_MODE=1 and
              StC_Z_DOF=TRUE]
          0   StC_X_KS     - Stop spring X stiffness (N/m)
          0   StC_Y_KS     - Stop spring Y stiffness (N/m)
          0   StC_Z_KS     - Stop spring Z stiffness (N/m) [used only when StC_DOF_MODE
              =1 and StC_Z_DOF=TRUE]
          0   StC_X_CS     - Stop spring X damping (N/(m/s))
          0   StC_Y_CS     - Stop spring Y damping (N/(m/s))
          0   StC_Z_CS     - Stop spring Z damping (N/(m/s)) [used only when StC_DOF_
              MODE=1 and StC_Z_DOF=TRUE]
```

```
---------------------- StC USER-DEFINED SPRING FORCES ----------------------  [used only
     when StC_DOF_MODE=1 or 2]
False         Use_F_TBL    - Use spring force from user-defined table (flag)
         17   NKInpSt      - Number of spring force input stations
---------------------- StC SPRING FORCES TABLE ----------------------------  [used only
     when StC_DOF_MODE=1 or 2]
    X               F_X              Y              F_Y              Z              F_
        Z
   (m)              (N)              (m)            (N)              (m)            (N
      )
-6.0000000E+00  -4.8000000E+06  -6.0000000E+00  -4.8000000E+06  -6.0000000E+00
    -4.8000000E+06
-5.0000000E+00  -2.4000000E+06  -5.0000000E+00  -2.4000000E+06  -5.0000000E+00
    -2.4000000E+06
-4.5000000E+00  -1.2000000E+06  -4.5000000E+00  -1.2000000E+06  -4.5000000E+00
    -1.2000000E+06
-4.0000000E+00  -6.0000000E+05  -4.0000000E+00  -6.0000000E+05  -4.0000000E+00
    -6.0000000E+05
-3.5000000E+00  -3.0000000E+05  -3.5000000E+00  -3.0000000E+05  -3.5000000E+00
    -3.0000000E+05
-3.0000000E+00  -1.5000000E+05  -3.0000000E+00  -1.5000000E+05  -3.0000000E+00
    -1.5000000E+05
-2.5000000E+00  -1.0000000E+05  -2.5000000E+00  -1.0000000E+05  -2.5000000E+00
    -1.0000000E+05
-2.0000000E+00  -6.5000000E+04  -2.0000000E+00  -6.5000000E+04  -2.0000000E+00
    -6.5000000E+04
 0.0000000E+00   0.0000000E+00   0.0000000E+00   0.0000000E+00   0.0000000E+00
     0.0000000E+00
 2.0000000E+00   6.5000000E+04   2.0000000E+00   6.5000000E+04   2.0000000E+00
     6.5000000E+04
 2.5000000E+00   1.0000000E+05   2.5000000E+00   1.0000000E+05   2.5000000E+00
     1.0000000E+05
 3.0000000E+00   1.5000000E+05   3.0000000E+00   1.5000000E+05   3.0000000E+00
     1.5000000E+05
 3.5000000E+00   3.0000000E+05   3.5000000E+00   3.0000000E+05   3.5000000E+00
     3.0000000E+05
 4.0000000E+00   6.0000000E+05   4.0000000E+00   6.0000000E+05   4.0000000E+00
     6.0000000E+05
 4.5000000E+00   1.2000000E+06   4.5000000E+00   1.2000000E+06   4.5000000E+00
     1.2000000E+06
 5.0000000E+00   2.4000000E+06   5.0000000E+00   2.4000000E+06   5.0000000E+00
     2.4000000E+06
 6.0000000E+00   4.8000000E+06   6.0000000E+00   4.8000000E+06   6.0000000E+00
     4.8000000E+06
---------------------- StructCtrl CONTROL ---------------------------------  [used only
     when StC_DOF_MODE=1 or 2]
          0   StC_CMODE     - Control mode (switch) {0:none; 1: Semi-Active Control Mode
              ; 2: Active Control Mode}
          1   StC_SA_MODE   - Semi-Active control mode {1: velocity-based ground hook
              control; 2: Inverse velocity-based ground hook control; 3: displacement-
              based ground hook control 4: Phase difference Algorithm with Friction
              Force 5: Phase difference Algorithm with Damping Force} (-)
          0   StC_X_C_HIGH  - StC X high damping for ground hook control
          0   StC_X_C_LOW   - StC X low damping for ground hook control
          0   StC_Y_C_HIGH  - StC Y high damping for ground hook control
          0   StC_Y_C_LOW   - StC Y low damping for ground hook control
          0   StC_Z_C_HIGH  - StC Z high damping for ground hook control [used only when
               StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
          0   StC_Z_C_LOW   - StC Z low damping for ground hook control  [used only when
               StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
          0   StC_X_C_BRAKE - StC X high damping for braking the StC (Don't use it now.
              should be zero)
          0   StC_Y_C_BRAKE - StC Y high damping for braking the StC (Don't use it now.
              should be zero)
          0   StC_Z_C_BRAKE - StC Z high damping for braking the StC (Don't use it now.
              should be zero) [used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- TLCD -----------------------------------------------  [used only
     when StC_DOF_MODE=3]
   7.9325     L_X             - X TLCD total length (m)
   6.5929     B_X             - X TLCD horizontal length (m)
   2.0217     area_X          - X TLCD cross-sectional area of vertical column (m^2)
   0.913      area_ratio_X    - X TLCD cross-sectional area ratio (vertical column area
       divided by horizontal column area) (-)
```

```
   2.5265     headLossCoeff_X - X TLCD head loss coeff (-)
     1000     rho_X           - X TLCD liquid density (kg/m^3)
   3.5767     L_Y             - Y TLCD total length (m)
   2.1788     B_Y             - Y TLCD horizontal length (m)
   1.2252     area_Y          - Y TLCD cross-sectional area of vertical column (m^2)
   2.7232     area_ratio_Y    - Y TLCD cross-sectional area ratio (vertical column area
       divided by horizontal column area) (-)
   0.6433     headLossCoeff_Y - Y TLCD head loss coeff (-)
     1000     rho_Y           - Y TLCD liquid density (kg/m^3)
---------------------- PRESCRIBED TIME SERIES ------------------------------  [used only
     when StC_DOF_MODE=4]
        1   PrescribedForcesCoord- Prescribed forces are in global or local
            coordinates (switch) {1: global; 2: local}
"seismic_forces_masslessJacket.dat"  PrescribedForcesFile   - Time series force and
    moment (7 columns of time, FX, FY, FZ, MX, MY, MZ)
----------------------------------------------------------------------------
```

D.7.3 Structural control 33

```
------- STRUCTURAL CONTROL (StC) INPUT FILE --------------------------------
Input file for tuned mass damper, module by Matt Lackner, Meghan Glade, and Semyung Park
     (UMass)
---------------------- SIMULATION CONTROL ----------------------------------
True          Echo         - Echo input data to <RootName>.ech (flag)
---------------------- StC DEGREES OF FREEDOM ------------------------------
          4   StC_DOF_MODE - DOF mode (switch) {0: No StC or TLCD DOF; 1: StC_X_DOF, StC
              _Y_DOF, and/or StC_Z_DOF (three independent StC DOFs); 2: StC_XY_DOF (Omni
              -Directional StC); 3: TLCD; 4: Prescribed force/moment time series}
false         StC_X_DOF    - DOF on or off for StC X (flag) [Used only when StC_DOF_MODE
    =1]
false         StC_Y_DOF    - DOF on or off for StC Y (flag) [Used only when StC_DOF_MODE
    =1]
false         StC_Z_DOF    - DOF on or off for StC Z (flag) [Used only when StC_DOF_MODE
    =1]
---------------------- StC LOCATION ----------------------------------------  [relative
    to the reference origin of component attached to]
   -17.00000   StC_P_X      - At rest X position of StC (m)
   -17.00000   StC_P_Y      - At rest Y position of StC (m)
   -74.50000   StC_P_Z      - At rest Z position of StC (m)
---------------------- StC INITIAL CONDITIONS ------------------------------  [used only
     when StC_DOF_MODE=1 or 2]
          0   StC_X_DSP    - StC X initial displacement (m) [relative to at rest
              position]
          0   StC_Y_DSP    - StC Y initial displacement (m) [relative to at rest
              position]
          0   StC_Z_DSP    - StC Z initial displacement (m) [relative to at rest
              position; used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC CONFIGURATION ----------------------------------- [used only
    when StC_DOF_MODE=1 or 2]
          0   StC_X_PSP    - Positive stop position (maximum X mass displacement) (m)
          0   StC_X_NSP    - Negative stop position (minimum X mass displacement) (m)
          0   StC_Y_PSP    - Positive stop position (maximum Y mass displacement) (m)
          0   StC_Y_NSP    - Negative stop position (minimum Y mass displacement) (m)
          0   StC_Z_PSP    - Positive stop position (maximum Z mass displacement) (m) [
              used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
          0   StC_Z_NSP    - Negative stop position (minimum Z mass displacement) (m) [
              used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC MASS, STIFFNESS, & DAMPING ----------------------  [used only
     when StC_DOF_MODE=1 or 2]
          0   StC_X_M      - StC X mass (kg) [must equal StC_Y_M for StC_DOF_MODE = 2]
         50   StC_Y_M      - StC Y mass (kg) [must equal StC_X_M for StC_DOF_MODE = 2]
          0   StC_Z_M      - StC Z mass (kg) [used only when StC_DOF_MODE=1 and StC_Z_
              DOF=TRUE]
          0   StC_XY_M     - StC Z mass (kg) [used only when StC_DOF_MODE=2]
       2300   StC_X_K      - StC X stiffness (N/m)
       2300   StC_Y_K      - StC Y stiffness (N/m)
          0   StC_Z_K      - StC Z stiffness (N/m) [used only when StC_DOF_MODE=1 and
              StC_Z_DOF=TRUE]
         35   StC_X_C      - StC X damping (N/(m/s))
         35   StC_Y_C      - StC Y damping (N/(m/s))
```

```
         0   StC_Z_C      - StC Z damping (N/(m/s)) [used only when StC_DOF_MODE=1 and
             StC_Z_DOF=TRUE]
         0   StC_X_KS     - Stop spring X stiffness (N/m)
         0   StC_Y_KS     - Stop spring Y stiffness (N/m)
         0   StC_Z_KS     - Stop spring Z stiffness (N/m) [used only when StC_DOF_MODE
             =1 and StC_Z_DOF=TRUE]
         0   StC_X_CS     - Stop spring X damping (N/(m/s))
         0   StC_Y_CS     - Stop spring Y damping (N/(m/s))
         0   StC_Z_CS     - Stop spring Z damping (N/(m/s)) [used only when StC_DOF_
             MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC USER-DEFINED SPRING FORCES ----------------------  [used only
     when StC_DOF_MODE=1 or 2]
False        Use_F_TBL    - Use spring force from user-defined table (flag)
        17   NKInpSt      - Number of spring force input stations
---------------------- StC SPRING FORCES TABLE ----------------------------  [used only
     when StC_DOF_MODE=1 or 2]
    X              F_X              Y              F_Y              Z              F_
        Z
   (m)              (N)              (m)              (N)              (m)              (N
       )
-6.0000000E+00  -4.8000000E+06  -6.0000000E+00  -4.8000000E+06  -6.0000000E+00
    -4.8000000E+06
-5.0000000E+00  -2.4000000E+06  -5.0000000E+00  -2.4000000E+06  -5.0000000E+00
    -2.4000000E+06
-4.5000000E+00  -1.2000000E+06  -4.5000000E+00  -1.2000000E+06  -4.5000000E+00
    -1.2000000E+06
-4.0000000E+00  -6.0000000E+05  -4.0000000E+00  -6.0000000E+05  -4.0000000E+00
    -6.0000000E+05
-3.5000000E+00  -3.0000000E+05  -3.5000000E+00  -3.0000000E+05  -3.5000000E+00
    -3.0000000E+05
-3.0000000E+00  -1.5000000E+05  -3.0000000E+00  -1.5000000E+05  -3.0000000E+00
    -1.5000000E+05
-2.5000000E+00  -1.0000000E+05  -2.5000000E+00  -1.0000000E+05  -2.5000000E+00
    -1.0000000E+05
-2.0000000E+00  -6.5000000E+04  -2.0000000E+00  -6.5000000E+04  -2.0000000E+00
    -6.5000000E+04
 0.0000000E+00   0.0000000E+00   0.0000000E+00   0.0000000E+00   0.0000000E+00
     0.0000000E+00
 2.0000000E+00   6.5000000E+04   2.0000000E+00   6.5000000E+04   2.0000000E+00
     6.5000000E+04
 2.5000000E+00   1.0000000E+05   2.5000000E+00   1.0000000E+05   2.5000000E+00
     1.0000000E+05
 3.0000000E+00   1.5000000E+05   3.0000000E+00   1.5000000E+05   3.0000000E+00
     1.5000000E+05
 3.5000000E+00   3.0000000E+05   3.5000000E+00   3.0000000E+05   3.5000000E+00
     3.0000000E+05
 4.0000000E+00   6.0000000E+05   4.0000000E+00   6.0000000E+05   4.0000000E+00
     6.0000000E+05
 4.5000000E+00   1.2000000E+06   4.5000000E+00   1.2000000E+06   4.5000000E+00
     1.2000000E+06
 5.0000000E+00   2.4000000E+06   5.0000000E+00   2.4000000E+06   5.0000000E+00
     2.4000000E+06
 6.0000000E+00   4.8000000E+06   6.0000000E+00   4.8000000E+06   6.0000000E+00
     4.8000000E+06
---------------------- StructCtrl CONTROL ---------------------------------  [used only
     when StC_DOF_MODE=1 or 2]
         0   StC_CMODE     - Control mode (switch) {0:none; 1: Semi-Active Control Mode
             ; 2: Active Control Mode}
         1   StC_SA_MODE   - Semi-Active control mode {1: velocity-based ground hook
             control; 2: Inverse velocity-based ground hook control; 3: displacement-
             based ground hook control 4: Phase difference Algorithm with Friction
             Force 5: Phase difference Algorithm with Damping Force} (-)
         0   StC_X_C_HIGH  - StC X high damping for ground hook control
         0   StC_X_C_LOW   - StC X low damping for ground hook control
         0   StC_Y_C_HIGH  - StC Y high damping for ground hook control
         0   StC_Y_C_LOW   - StC Y low damping for ground hook control
         0   StC_Z_C_HIGH  - StC Z high damping for ground hook control [used only when
              StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
         0   StC_Z_C_LOW   - StC Z low damping for ground hook control  [used only when
              StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
         0   StC_X_C_BRAKE - StC X high damping for braking the StC (Don't use it now.
             should be zero)
         0   StC_Y_C_BRAKE - StC Y high damping for braking the StC (Don't use it now.
```

```
            should be zero)
        0   StC_Z_C_BRAKE - StC Z high damping for braking the StC (Don't use it now.
            should be zero) [used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- TLCD ------------------------------------------------  [used only
     when StC_DOF_MODE=3]
   7.9325     L_X            - X TLCD total length (m)
   6.5929     B_X            - X TLCD horizontal length (m)
   2.0217     area_X         - X TLCD cross-sectional area of vertical column (m^2)
   0.913      area_ratio_X   - X TLCD cross-sectional area ratio (vertical column area
       divided by horizontal column area) (-)
   2.5265     headLossCoeff_X - X TLCD head loss coeff (-)
     1000     rho_X          - X TLCD liquid density (kg/m^3)
   3.5767     L_Y            - Y TLCD total length (m)
   2.1788     B_Y            - Y TLCD horizontal length (m)
   1.2252     area_Y         - Y TLCD cross-sectional area of vertical column (m^2)
   2.7232     area_ratio_Y   - Y TLCD cross-sectional area ratio (vertical column area
       divided by horizontal column area) (-)
   0.6433     headLossCoeff_Y - Y TLCD head loss coeff (-)
     1000     rho_Y          - Y TLCD liquid density (kg/m^3)
---------------------- PRESCRIBED TIME SERIES ------------------------------  [used only
     when StC_DOF_MODE=4]
        1   PrescribedForcesCoord- Prescribed forces are in global or local
            coordinates (switch) {1: global; 2: local}
"seismic_forces_masslessJacket.dat"  PrescribedForcesFile   - Time series force and
    moment (7 columns of time, FX, FY, FZ, MX, MY, MZ)
----------------------------------------------------------------------------
```

D.7.4 Structural control 41

```
------- STRUCTURAL CONTROL (StC) INPUT FILE ----------------------------
Input file for tuned mass damper, module by Matt Lackner, Meghan Glade, and Semyung Park
     (UMass)
---------------------- SIMULATION CONTROL ----------------------------------
True           Echo         - Echo input data to <RootName>.ech (flag)
---------------------- StC DEGREES OF FREEDOM ------------------------------
        4   StC_DOF_MODE - DOF mode (switch) {0: No StC or TLCD DOF; 1: StC_X_DOF, StC
            _Y_DOF, and/or StC_Z_DOF (three independent StC DOFs); 2: StC_XY_DOF (Omni
            -Directional StC); 3: TLCD; 4: Prescribed force/moment time series}
false          StC_X_DOF    - DOF on or off for StC X (flag) [Used only when StC_DOF_MODE
    =1]
false          StC_Y_DOF    - DOF on or off for StC Y (flag) [Used only when StC_DOF_MODE
    =1]
false          StC_Z_DOF    - DOF on or off for StC Z (flag) [Used only when StC_DOF_MODE
    =1]
---------------------- StC LOCATION ----------------------------------------  [relative
    to the reference origin of component attached to]
   17.00000   StC_P_X      - At rest X position of StC (m)
  -17.00000   StC_P_Y      - At rest Y position of StC (m)
  -74.50000   StC_P_Z      - At rest Z position of StC (m)
---------------------- StC INITIAL CONDITIONS ------------------------------ [used only
    when StC_DOF_MODE=1 or 2]
        0   StC_X_DSP    - StC X initial displacement (m) [relative to at rest
            position]
        0   StC_Y_DSP    - StC Y initial displacement (m) [relative to at rest
            position]
        0   StC_Z_DSP    - StC Z initial displacement (m) [relative to at rest
            position; used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC CONFIGURATION -----------------------------------  [used only
     when StC_DOF_MODE=1 or 2]
        0   StC_X_PSP    - Positive stop position (maximum X mass displacement) (m)
        0   StC_X_NSP    - Negative stop position (minimum X mass displacement) (m)
        0   StC_Y_PSP    - Positive stop position (maximum Y mass displacement) (m)
        0   StC_Y_NSP    - Negative stop position (minimum Y mass displacement) (m)
        0   StC_Z_PSP    - Positive stop position (maximum Z mass displacement) (m) [
            used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
        0   StC_Z_NSP    - Negative stop position (minimum Z mass displacement) (m) [
            used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC MASS, STIFFNESS, & DAMPING ----------------------  [used only
     when StC_DOF_MODE=1 or 2]
        0   StC_X_M      - StC X mass (kg) [must equal StC_Y_M for StC_DOF_MODE = 2]
       50   StC_Y_M      - StC Y mass (kg) [must equal StC_X_M for StC_DOF_MODE = 2]
```

```
         0   StC_Z_M      - StC Z mass (kg) [used only when StC_DOF_MODE=1 and StC_Z_
             DOF=TRUE]
         0   StC_XY_M     - StC Z mass (kg) [used only when StC_DOF_MODE=2]
      2300   StC_X_K      - StC X stiffness (N/m)
      2300   StC_Y_K      - StC Y stiffness (N/m)
         0   StC_Z_K      - StC Z stiffness (N/m) [used only when StC_DOF_MODE=1 and
             StC_Z_DOF=TRUE]
        35   StC_X_C      - StC X damping (N/(m/s))
        35   StC_Y_C      - StC Y damping (N/(m/s))
         0   StC_Z_C      - StC Z damping (N/(m/s)) [used only when StC_DOF_MODE=1 and
             StC_Z_DOF=TRUE]
         0   StC_X_KS     - Stop spring X stiffness (N/m)
         0   StC_Y_KS     - Stop spring Y stiffness (N/m)
         0   StC_Z_KS     - Stop spring Z stiffness (N/m) [used only when StC_DOF_MODE
             =1 and StC_Z_DOF=TRUE]
         0   StC_X_CS     - Stop spring X damping (N/(m/s))
         0   StC_Y_CS     - Stop spring Y damping (N/(m/s))
         0   StC_Z_CS     - Stop spring Z damping (N/(m/s)) [used only when StC_DOF_
             MODE=1 and StC_Z_DOF=TRUE]
---------------------- StC USER-DEFINED SPRING FORCES ----------------------  [used only
     when StC_DOF_MODE=1 or 2]
False        Use_F_TBL    - Use spring force from user-defined table (flag)
        17   NKInpSt      - Number of spring force input stations
---------------------- StC SPRING FORCES TABLE ----------------------------  [used only
     when StC_DOF_MODE=1 or 2]
    X               F_X              Y              F_Y              Z              F_
        Z
   (m)              (N)             (m)             (N)             (m)             (N
      )
-6.0000000E+00  -4.8000000E+06  -6.0000000E+00  -4.8000000E+06  -6.0000000E+00
    -4.8000000E+06
-5.0000000E+00  -2.4000000E+06  -5.0000000E+00  -2.4000000E+06  -5.0000000E+00
    -2.4000000E+06
-4.5000000E+00  -1.2000000E+06  -4.5000000E+00  -1.2000000E+06  -4.5000000E+00
    -1.2000000E+06
-4.0000000E+00  -6.0000000E+05  -4.0000000E+00  -6.0000000E+05  -4.0000000E+00
    -6.0000000E+05
-3.5000000E+00  -3.0000000E+05  -3.5000000E+00  -3.0000000E+05  -3.5000000E+00
    -3.0000000E+05
-3.0000000E+00  -1.5000000E+05  -3.0000000E+00  -1.5000000E+05  -3.0000000E+00
    -1.5000000E+05
-2.5000000E+00  -1.0000000E+05  -2.5000000E+00  -1.0000000E+05  -2.5000000E+00
    -1.0000000E+05
-2.0000000E+00  -6.5000000E+04  -2.0000000E+00  -6.5000000E+04  -2.0000000E+00
    -6.5000000E+04
 0.0000000E+00   0.0000000E+00   0.0000000E+00   0.0000000E+00   0.0000000E+00
     0.0000000E+00
 2.0000000E+00   6.5000000E+04   2.0000000E+00   6.5000000E+04   2.0000000E+00
     6.5000000E+04
 2.5000000E+00   1.0000000E+05   2.5000000E+00   1.0000000E+05   2.5000000E+00
     1.0000000E+05
 3.0000000E+00   1.5000000E+05   3.0000000E+00   1.5000000E+05   3.0000000E+00
     1.5000000E+05
 3.5000000E+00   3.0000000E+05   3.5000000E+00   3.0000000E+05   3.5000000E+00
     3.0000000E+05
 4.0000000E+00   6.0000000E+05   4.0000000E+00   6.0000000E+05   4.0000000E+00
     6.0000000E+05
 4.5000000E+00   1.2000000E+06   4.5000000E+00   1.2000000E+06   4.5000000E+00
     1.2000000E+06
 5.0000000E+00   2.4000000E+06   5.0000000E+00   2.4000000E+06   5.0000000E+00
     2.4000000E+06
 6.0000000E+00   4.8000000E+06   6.0000000E+00   4.8000000E+06   6.0000000E+00
     4.8000000E+06
---------------------- StructCtrl CONTROL ---------------------------------  [used only
     when StC_DOF_MODE=1 or 2]
         0   StC_CMODE     - Control mode (switch) {0:none; 1: Semi-Active Control Mode
             ; 2: Active Control Mode}
         1   StC_SA_MODE   - Semi-Active control mode {1: velocity-based ground hook
             control; 2: Inverse velocity-based ground hook control; 3: displacement-
             based ground hook control 4: Phase difference Algorithm with Friction
             Force 5: Phase difference Algorithm with Damping Force} (-)
         0   StC_X_C_HIGH  - StC X high damping for ground hook control
         0   StC_X_C_LOW   - StC X low damping for ground hook control
```

```
        0   StC_Y_C_HIGH  - StC Y high damping for ground hook control
        0   StC_Y_C_LOW   - StC Y low damping for ground hook control
        0   StC_Z_C_HIGH  - StC Z high damping for ground hook control [used only when
             StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
        0   StC_Z_C_LOW   - StC Z low damping for ground hook control  [used only when
             StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
        0   StC_X_C_BRAKE - StC X high damping for braking the StC (Don't use it now.
            should be zero)
        0   StC_Y_C_BRAKE - StC Y high damping for braking the StC (Don't use it now.
            should be zero)
        0   StC_Z_C_BRAKE - StC Z high damping for braking the StC (Don't use it now.
            should be zero) [used only when StC_DOF_MODE=1 and StC_Z_DOF=TRUE]
---------------------- TLCD -------------------------------------------------  [used only
     when StC_DOF_MODE=3]
   7.9325     L_X            - X TLCD total length (m)
   6.5929     B_X            - X TLCD horizontal length (m)
   2.0217     area_X         - X TLCD cross-sectional area of vertical column (m^2)
   0.913      area_ratio_X   - X TLCD cross-sectional area ratio (vertical column area
       divided by horizontal column area) (-)
   2.5265     headLossCoeff_X - X TLCD head loss coeff (-)
     1000     rho_X          - X TLCD liquid density (kg/m^3)
   3.5767     L_Y            - Y TLCD total length (m)
   2.1788     B_Y            - Y TLCD horizontal length (m)
   1.2252     area_Y         - Y TLCD cross-sectional area of vertical column (m^2)
   2.7232     area_ratio_Y   - Y TLCD cross-sectional area ratio (vertical column area
       divided by horizontal column area) (-)
   0.6433     headLossCoeff_Y - Y TLCD head loss coeff (-)
     1000     rho_Y          - Y TLCD liquid density (kg/m^3)
---------------------- PRESCRIBED TIME SERIES ------------------------------  [used only
     when StC_DOF_MODE=4]
        1   PrescribedForcesCoord- Prescribed forces are in global or local
            coordinates (switch) {1: global; 2: local}
"seismic_forces_masslessJacket.dat"  PrescribedForcesFile   - Time series force and
    moment (7 columns of time, FX, FY, FZ, MX, MY, MZ)
---------------------------------------------------------------------------
```

D.8 HydroDyn input file

```
------- HydroDyn v2.03.* Input File ----------------------------------------
INNWIND.EU 10 MW Offshore Wind Turbine with Reference Jacket HydroDyn input properties
                    Jan Haefele (Leibniz Universitaet Hannover), j.haefele@isd.uni-
    hannover.de
True            Echo           - Echo the input file data (flag)
---------------------- ENVIRONMENTAL CONDITIONS -----------------------------
          1025   WtrDens        - Water density (kg/m^3)
          48.5   WtrDpth        - Water depth (meters)
                         0   MSL2SWL        - Offset between still-water level and mean
                             sea level (meters) [positive upward; must be zero if
                             HasWAMIT=TRUE]
---------------------- WAVES -----------------------------------------------
             2   WaveMod        - Incident wave kinematics model {0: none=still water,
                 1: regular (periodic), 1P#: regular (periodic) with user-specified
                 phase, 2: JONSWAP/Pierson-Moskowitz spectrum (irregular), 3: White
                 noise spectrum (irregular), 4: user-defined spectrum from routine
                 UserWaveSpctrm (irregular), 5: GH Bladed wave data [option 5 is invalid
                  for HasWAMIT = TRUE]} (switch)
             0   WaveStMod      - Model for stretching incident wave kinematics to
                 instantaneous free surface {0: none=no stretching, 1: vertical
                 stretching, 2: extrapolation stretching, 3: Wheeler stretching} (switch
                 ) [unused when WaveMod=0 or when HasWAMIT = TRUE]
          2000   WaveTMax       - Analysis time for incident wave calculations (sec) [
             unused when WaveMod=0] [determines WaveDOmega=2Pi/WaveTMax in the IFFT]
                 0.1   WaveDT         - Time step for incident wave calculations     (
                     sec) [unused when WaveMod=0] [0.1<=WaveDT<=1.0 recommended] [
                     determines WaveOmegaMax=Pi/WaveDT in the IFFT]
           8.0   WaveHs         - Significant wave height of incident waves (meters) [
              used only when WaveMod=1, 2, or 3]
            12   WaveTp         - Peak-spectral period of incident waves       (sec) [
               used only when WaveMod=1 or 2]
        2.3892   WavePkShp      - Peak-shape parameter of incident wave spectrum (-) or
            DEFAULT (string) [used only when WaveMod=2] [use 1.0 for Pierson-Moskowitz]
```

```
            0   WvLowCOff      - Low  cut-off frequency or lower frequency limit of the
                 wave spectrum beyond which the wave spectrum is zeroed (rad/s) [used
                only when WaveMod=2, 3, or 4]
         62.8   WvHiCOff       - High cut-off frequency or upper frequency limit of the
              wave spectrum beyond which the wave spectrum is zeroed (rad/s) [used only
              when WaveMod=2, 3, or 4]
            0   WaveDir        - Incident wave propagation heading direction
                                        (degrees) [unused when WaveMod=0 or 5]
            0   WaveDirMod     - Directional spreading function {0: none, 1: COS2S}
                                 (-)       [only used when WaveMod=2,3,4]
            1   WaveDirSpread  - Wave direction spreading coefficient ( > 0 )
                                       (-)       [only used when WaveMod=2,3,4 and
                WaveDirMod=1]
            1   WaveNDir       - Number of wave directions
                                                           (-)       [odd number only,
                may be adjusted within HydroDyn]
           90   WaveDirRange   - Range of wave directions (full range: WaveDir +/- 1/2*
               WaveDirRange) (degrees) [only used when WaveMod=2,3,4 and WaveDirMod=1]
    123456789   WaveSeed(1)    - First  random seed of incident waves [-2147483648 to
        2147483647]    (-)       [unused when WaveMod=0 or 5]
   1011121314   WaveSeed(2)    - Second random seed of incident waves [-2147483648 to
       2147483647]    (-)       [unused when WaveMod=0 or 5]
False           WaveNDAmp      - Flag for normally distributed amplitudes (flag)
""              WvKinFile      - Root name of GH Bladed files containing wave data
          (quoted string)       [used only when WaveMod=5]
            1   NWaveElev      - Number of points where the incident wave elevations
                can be computed (-)       [maximum of 9 output locations]
            0   WaveElevxi     - List of xi-coordinates for points where the incident
                wave elevations can be output (meters) [NWaveElev points, separated by
                commas or white space; usused if NWaveElev = 0]
            0   WaveElevyi     - List of yi-coordinates for points where the incident
                wave elevations can be output (meters) [NWaveElev points, separated by
                commas or white space; usused if NWaveElev = 0]
---------------------- 2ND-ORDER WAVES -------- [unused with WaveMod=0 or 6]
False           WvDiffQTF      - Full difference-frequency 2nd-order wave kinematics (
    flag)
False           WvSumQTF       - Full summation-frequency  2nd-order wave kinematics (
    flag)
            0   WvLowCOffD     - Low  frequency cutoff used in the difference-
                frequencies (rad/s) [Only used with a difference-frequency method]
          3.5   WvHiCOffD      - High frequency cutoff used in the difference-
              frequencies (rad/s) [Only used with a difference-frequency method]
          0.1   WvLowCOffS     - Low  frequency cutoff used in the summation-
              frequencies  (rad/s) [Only used with a summation-frequency  method]
          3.5   WvHiCOffS      - High frequency cutoff used in the summation-
              frequencies  (rad/s) [Only used with a summation-frequency  method]
---------------------- CURRENT --------------------------------------------- [unused
    with WaveMod=6]
            0   CurrMod        - Current profile model {0: none=no current, 1: standard
                , 2: user-defined from routine UserCurrent} (switch)
            0   CurrSSV0       - Sub-surface current velocity at still water level  (m/
                s) [used only when CurrMod=1]
"DEFAULT"       CurrSSDir      - Sub-surface current heading direction (degrees) or
    DEFAULT (string) [used only when CurrMod=1]
           20   CurrNSRef      - Near-surface current reference depth            (
               meters) [used only when CurrMod=1]
            0   CurrNSV0       - Near-surface current velocity at still water level (m/
                s) [used only when CurrMod=1]
            0   CurrNSDir      - Near-surface current heading direction         (
                degrees) [used only when CurrMod=1]
            0   CurrDIV        - Depth-independent current velocity                 (m/
                s) [used only when CurrMod=1]
            0   CurrDIDir      - Depth-independent current heading direction    (
                degrees) [used only when CurrMod=1]
---------------------- FLOATING PLATFORM ----------------------------------- [unused
    with WaveMod=6]
            0   PotMod         - Potential-flow model {0: none=no potential flow, 1:
                frequency-to-time-domain transforms based on WAMIT output, 2: fluid-
                impulse theory (FIT)} (switch)
0                               ExtnMod
            0   RdtnMod        - Radiation memory-effect model {0: no memory-effect
                calculation, 1: convolution, 2: state-space} (switch) [only used when
                PotMod=1; STATE-SPACE REQUIRES *.ss INPUT FILE]
```

```
            0   RdtnTMax       - Analysis time for wave radiation kernel calculations (
                sec) [only used when PotMod=1 and RdtnMod>0; determines RdtnDOmega=Pi/
                RdtnTMax in the cosine transform; MAKE SURE THIS IS LONG ENOUGH FOR THE
                 RADIATION IMPULSE RESPONSE FUNCTIONS TO DECAY TO NEAR-ZERO FOR THE
                GIVEN PLATFORM!]
            0   RdtnDT         - Time step for wave radiation kernel calculations (sec)
                 [only used when PotMod=1 and RdtnMod=1; DT<=RdtnDT<=0.1 recommended;
                determines RdtnOmegaMax=Pi/RdtnDT in the cosine transform]
0                               NBody
0                               NBodyMod
""              PotFile        - Root name of potential-flow model data; WAMIT output
    files containing the linear, nondimensionalized, hydrostatic restoring matrix (.hst)
    , frequency-dependent hydrodynamic added mass matrix and damping matrix (.1), and
    frequency- and direction-dependent wave excitation force vector per unit wave
    amplitude (.3) (quoted string) [MAKE SURE THE FREQUENCIES INHERENT IN THESE WAMIT
    FILES SPAN THE PHYSICALLY-SIGNIFICANT RANGE OF FREQUENCIES FOR THE GIVEN PLATFORM;
    THEY MUST CONTAIN THE ZERO- AND INFINITE-FREQUENCY LIMITS!]
            1   WAMITULEN      - Characteristic body length scale used to
                redimensionalize WAMIT output (meters) [only used when PotMod=1]
                  0.0   PtfmRefxt - The xt offset of the body reference point(s) from
                      (0,0,0) (meters) [1 to NBody] [only used when PotMod=1]
                  0.0   PtfmRefyt - The yt offset of the body reference point(s) from
                      (0,0,0) (meters) [1 to NBody] [only used when PotMod=1]
                  0.0   PtfmRefzt - The zt offset of the body reference point(s) from
                      (0,0,0) (meters) [1 to NBody] [only used when PotMod=1. If NBody-
                      Mod=2,PtfmRefzt=0.0]
                  0.0   PtfmRefztRot - The rotation about zt of the body reference
                      frame(s) from xt/yt (degrees) [1 to NBody] [only used when PotMod
                      =1]
            0   PtfmVol0       - Displaced volume of water when the platform is in its
                undisplaced position (m^3) [only used when PotMod=1; USE THE SAME VALUE
                 COMPUTED BY WAMIT AS OUTPUT IN THE .OUT FILE!]
            0   PtfmCOBxt      - The xt offset of the center of buoyancy (COB) from the
                 platform reference point (meters)  [only used when PotMod=1]
            0   PtfmCOByt      - The yt offset of the center of buoyancy (COB) from the
                 platform reference point (meters)  [only used when PotMod=1]
---------------------- 2ND-ORDER FLOATING PLATFORM FORCES ------------------ [unused
    with WaveMod=0 or 6, or PotMod=0 or 2]
            0   MnDrift        - Mean-drift 2nd-order forces computed
                                               {0: None; [7, 8, 9, 10, 11, or
                12]: WAMIT file to use} [Only one of MnDrift, NewmanApp, or DiffQTF can
                 be non-zero]
            0   NewmanApp      - Mean- and slow-drift 2nd-order forces computed with
                Newman's approximation {0: None; [7, 8, 9, 10, 11, or 12]: WAMIT file
                to use} [Only one of MnDrift, NewmanApp, or DiffQTF can be non-zero.
                Used only when WaveDirMod=0]
            0   DiffQTF        - Full difference-frequency 2nd-order forces computed
                with full QTF          {0: None; [10, 11, or 12]: WAMIT file to use}
                         [Only one of MnDrift, NewmanApp, or DiffQTF can be non-zero]
            0   SumQTF         - Full summation -frequency 2nd-order forces computed
                with full QTF          {0: None; [10, 11, or 12]: WAMIT file to use}
---------------------- PLATFORM ADDITIONAL STIFFNESS AND DAMPING  ----------
0 AddF0 - Additional preload (N, N-m) [If NBodyMod=1, one size 6*NBody x 1 vector; if
    NBodyMod>1, NBody size 6 x 1 vectors]
0
0
0
0
0
            0             0             0             0             0             0
                AddCLin  - Additional linear stiffness (N/m, N/rad, N-m/m, N-m/rad)
            0             0             0             0             0             0
            0             0             0             0             0             0
            0             0             0             0             0             0
            0             0             0             0             0             0
            0             0             0             0             0             0
            0             0             0             0             0             0
                AddBLin  - Additional linear damping(N/(m/s), N/(rad/s), N-m/(m/s), N-m
                /(rad/s))
            0             0             0             0             0             0
            0             0        1.115919E+6        0             0             0
            0             0             0             0             0             0
            0             0             0             0             0             0
```

```
            0             0             0             0             0             0
            0             0             0             0             0             0
                AddBQuad - Additional quadratic drag(N/(m/s)^2, N/(rad/s)^2, N-m(m/s)
                ^2, N-m/(rad/s)^2)
            0             0             0             0             0             0
            0             0             0             0             0             0
            0             0             0             0             0             0
            0             0             0             0             0             0
            0             0             0             0             0             0
---------------------- AXIAL COEFFICIENTS ---------------------------------
            1   NAxCoef        - Number of axial coefficients (-)
AxCoefID  AxCd     AxCa     AxCp
   (-)    (-)      (-)      (-)
    1     0.00     1.00     1.00
---------------------- MEMBER JOINTS ---------------------------------------
            62   NJoints        - Number of joints (-)   [must be exactly 0 or at least
                2]
JointID   Jointxi     Jointyi     Jointzi  JointAxID   JointOvrlp   [JointOvrlp= 0: do
    nothing at joint, 1: eliminate overlaps by calculating super member]
   (-)     (m)         (m)         (m)        (-)        (switch)
1       -17     17      -48.6   1       0
2       -17     17      -47.5   1       0
3       -16.80366412    16.80366412     -46.214 1       0
4       -13.58381679    13.58381679     -25.124 1       0
5       -10.98229008    10.98229008     -8.084  1       0
6       -8.881526718    8.881526718     5.676   1       0
7       -7.183206107    7.183206107     16.8    1       0
8       -7      7       18      1       0
9       17      17      -48.6   1       0
10      17      17      -47.5   1       0
11      16.80366412     16.80366412     -46.214 1       0
12      13.58381679     13.58381679     -25.124 1       0
13      10.98229008     10.98229008     -8.084  1       0
14      8.881526718     8.881526718     5.676   1       0
15      7.183206107     7.183206107     16.8    1       0
16      7       7       18      1       0
17      -15.02315348    0       -34.513 1       0
18      -12.14530387    0       -15.704 1       0
19      -9.820821823    0       -0.509  1       0
20      -7.942595455    0       11.901  1       0
21      0       15.02315348     -34.513 1       0
22      0       12.14530387     -15.704 1       0
23      0       9.820821823     -0.509  1       0
24      0       7.942595455     11.901  1       0
25      15.02315348     0       -34.513 1       0
26      12.14530387     0       -15.704 1       0
27      9.820821823     0       -0.509  1       0
28      7.942595455     0       11.901  1       0
29      -16.80366412    0       -46.214 1       0
30      16.80366412     0       -46.214 1       0
31      0       16.80366412     -46.214 1       0
32      0       -16.80366412    -46.214 1       0
33      -17     -17     -48.6   1       0
34      -17     -17     -47.5   1       0
35      -16.80366412    -16.80366412    -46.214 1       0
36      -13.58381679    -13.58381679    -25.124 1       0
37      -10.98229008    -10.98229008    -8.084  1       0
38      -8.881526718    -8.881526718    5.676   1       0
39      -7.183206107    -7.183206107    16.8    1       0
40      -7      -7      18      1       0
41      17      -17     -48.6   1       0
42      17      -17     -47.5   1       0
43      16.80366412     -16.80366412    -46.214 1       0
44      13.58381679     -13.58381679    -25.124 1       0
45      10.98229008     -10.98229008    -8.084  1       0
46      8.881526718     -8.881526718    5.676   1       0
47      7.183206107     -7.183206107    16.8    1       0
48      7       -7      18      1       0
49      0       -15.02315348    -34.513 1       0
50      0       -12.14530387    -15.704 1       0
51      0       -9.820821823    -0.509  1       0
52      0       -7.942595455    11.901  1       0
53      0       0       18      1       0
```

```
54      -6.541984733    6.541984733     22      1       0
55      -2.711754506    2.711754506     26      1       0
56      2.711754506     2.711754506     26      1       0
57      6.541984733     6.541984733     22      1       0
58      -6.541984733    -6.541984733    22      1       0
59      -2.711754506    -2.711754506    26      1       0
60      2.711754506     -2.711754506    26      1       0
61      6.541984733     -6.541984733    22      1       0
62      0       0       26      1       0
---------------------- MEMBER CROSS-SECTION PROPERTIES ---------------------
             13   NPropSets      - Number of member property sets (-)
PropSetID    PropD         PropThck
   (-)        (m)            (m)
1       1.4     0.12
2       1.4     0.07
3       1.4     0.042
4       1.4     0.042
5       1.4     0.042
6       1.4     0.066
7       1.04    0.02
8       1.06    0.03
9       0.936   0.018
10      0.84    0.02
11      0.832   0.016
12      1.4     0.08
13      8.3     0.07
---------------------- SIMPLE HYDRODYNAMIC COEFFICIENTS (model 1) ----------
SimplCd    SimplCdMG    SimplCa    SimplCaMG    SimplCp    SimplCpMG   SimplAxCa
    SimplAxCaMG  SimplAxCa  SimplAxCaMG   SimplAxCp   SimplAxCpMG
  (-)         (-)         (-)         (-)         (-)         (-)         (-)
      (-)         (-)         (-)         (-)         (-)
  1.00        1.00        1.00        1.00        1.00        1.00        1.00
      1.00        1.00        1.00        1.00        1.00
---------------------- DEPTH-BASED HYDRODYNAMIC COEFFICIENTS (model 2) -----
            0   NCoefDpth       - Number of depth-dependent coefficients (-)
Dpth      DpthCd   DpthCdMG   DpthCa   DpthCaMG       DpthCp   DpthCpMG   DpthAxCa
    DpthAxCaMG       DpthAxCp   DpthAxCpMG
(m)       (-)       (-)        (-)      (-)            (-)       (-)          (-)
    (-)              (-)         (-)
---------------------- MEMBER-BASED HYDRODYNAMIC COEFFICIENTS (model 3) ----
            0   NCoefMembers       - Number of member-based coefficients (-)
MemberID    MemberCd1     MemberCd2     MemberCdMG1   MemberCdMG2    MemberCa1
    MemberCa2    MemberCaMG1   MemberCaMG2    MemberCp1     MemberCp2    MemberCpMG1
    MemberCpMG2   MemberAxCa1   MemberAxCa2  MemberAxCaMG1 MemberAxCaMG2  MemberAxCp1
    MemberAxCp2   MemberAxCpMG1   MemberAxCpMG2
   (-)         (-)           (-)           (-)           (-)           (-)           (-)
                  (-)           (-)           (-)           (-)           (-)
       (-)           (-)           (-)           (-)           (-)           (-)
                 (-)           (-)           (-)
-------------------- MEMBERS -------------------------------------------------
          117   NMembers       - Number of members (-)
MemberID  MJointID1  MJointID2  MPropSetID1  MPropSetID2  MDivSize   MCoefMod  PropWAMIT
       [MCoefMod=1: use simple coeff table, 2: use depth-based coeff table, 3: use
    member-based coeff table] [ PropWAMIT = TRUE if member is modeled in WAMIT]
  (-)        (-)        (-)         (-)          (-)         (m)       (switch)   (flag)
1       1       2       1       1       1       1       FALSE
2       2       3       1       1       1       1       FALSE
3       3       4       2       2       1       1       FALSE
4       4       5       3       3       1       1       FALSE
5       5       6       4       4       1       1       FALSE
6       6       7       5       5       1       1       FALSE
7       7       8       6       6       1       1       FALSE
8       9       10      1       1       1       1       FALSE
9       10      11      1       1       1       1       FALSE
10      11      12      2       2       1       1       FALSE
11      12      13      3       3       1       1       FALSE
12      13      14      4       4       1       1       FALSE
13      14      15      5       5       1       1       FALSE
14      15      16      6       6       1       1       FALSE
15      33      34      1       1       1       1       FALSE
16      34      35      1       1       1       1       FALSE
17      35      36      2       2       1       1       FALSE
18      36      37      3       3       1       1       FALSE
```

19	37	38	4	4	1	1	FALSE
20	38	39	5	5	1	1	FALSE
21	39	40	6	6	1	1	FALSE
22	41	42	1	1	1	1	FALSE
23	42	43	1	1	1	1	FALSE
24	43	44	2	2	1	1	FALSE
25	44	45	3	3	1	1	FALSE
26	45	46	4	4	1	1	FALSE
27	46	47	5	5	1	1	FALSE
28	47	48	6	6	1	1	FALSE
29	3	29	7	7	1	1	FALSE
30	29	35	7	7	1	1	FALSE
31	3	17	8	8	1	1	FALSE
32	35	17	8	8	1	1	FALSE
33	4	17	8	8	1	1	FALSE
34	36	17	8	8	1	1	FALSE
35	4	18	9	9	1	1	FALSE
36	36	18	9	9	1	1	FALSE
37	5	18	9	9	1	1	FALSE
38	37	18	9	9	1	1	FALSE
39	5	19	10	10	0.2	1	FALSE
40	37	19	10	10	0.2	1	FALSE
41	6	19	10	10	0.2	1	FALSE
42	38	19	10	10	0.2	1	FALSE
43	6	20	11	11	1	1	FALSE
44	38	20	11	11	1	1	FALSE
45	7	20	11	11	1	1	FALSE
46	39	20	11	11	1	1	FALSE
47	11	30	7	7	1	1	FALSE
48	30	43	7	7	1	1	FALSE
49	11	25	8	8	1	1	FALSE
50	43	25	8	8	1	1	FALSE
51	12	25	8	8	1	1	FALSE
52	44	25	8	8	1	1	FALSE
53	12	26	9	9	1	1	FALSE
54	44	26	9	9	1	1	FALSE
55	13	26	9	9	1	1	FALSE
56	45	26	9	9	1	1	FALSE
57	13	27	10	10	0.2	1	FALSE
58	45	27	10	10	0.2	1	FALSE
59	14	27	10	10	0.2	1	FALSE
60	46	27	10	10	0.2	1	FALSE
61	14	28	11	11	1	1	FALSE
62	46	28	11	11	1	1	FALSE
63	15	28	11	11	1	1	FALSE
64	47	28	11	11	1	1	FALSE
65	3	31	7	7	1	1	FALSE
66	11	31	7	7	1	1	FALSE
67	3	21	8	8	1	1	FALSE
68	11	21	8	8	1	1	FALSE
69	4	21	8	8	1	1	FALSE
70	12	21	8	8	1	1	FALSE
71	4	22	9	9	1	1	FALSE
72	12	22	9	9	1	1	FALSE
73	5	22	9	9	1	1	FALSE
74	13	22	9	9	1	1	FALSE
75	5	23	10	10	0.2	1	FALSE
76	13	23	10	10	0.2	1	FALSE
77	6	23	10	10	0.2	1	FALSE
78	14	23	10	10	0.2	1	FALSE
79	6	24	11	11	1	1	FALSE
80	14	24	11	11	1	1	FALSE
81	7	24	11	11	1	1	FALSE
82	15	24	11	11	1	1	FALSE
83	35	32	7	7	1	1	FALSE
84	43	32	7	7	1	1	FALSE
85	35	49	8	8	1	1	FALSE
86	43	49	8	8	1	1	FALSE
87	36	49	8	8	1	1	FALSE
88	44	49	8	8	1	1	FALSE
89	36	50	9	9	1	1	FALSE
90	44	50	9	9	1	1	FALSE
91	37	50	9	9	1	1	FALSE

```
92      45      50      9       9       1       1       FALSE
93      37      51      10      10      0.2     1       FALSE
94      45      51      10      10      0.2     1       FALSE
95      38      51      10      10      0.2     1       FALSE
96      46      51      10      10      0.2     1       FALSE
97      38      52      11      11      1       1       FALSE
98      46      52      11      11      1       1       FALSE
99      39      52      11      11      1       1       FALSE
100     47      52      11      11      1       1       FALSE
101     8       53      12      12      1       1       FALSE
102     16      53      12      12      1       1       FALSE
103     40      53      12      12      1       1       FALSE
104     48      53      12      12      1       1       FALSE
105     53      62      13      13      1       1       FALSE
106     8       54      12      12      1       1       FALSE
107     16      57      12      12      1       1       FALSE
108     40      58      12      12      1       1       FALSE
109     48      61      12      12      1       1       FALSE
110     55      62      12      12      1       1       FALSE
111     56      62      12      12      1       1       FALSE
112     59      62      12      12      1       1       FALSE
113     60      62      12      12      1       1       FALSE
114     54      55      12      12      1       1       FALSE
115     57      56      12      12      1       1       FALSE
116     58      59      12      12      1       1       FALSE
117     61      60      12      12      1       1       FALSE
------------------------ FILLED MEMBERS -------------------------------------
            1   NFillGroups     - Number of filled member groups (-) [If FillDens =
                DEFAULT, then FillDens = WtrDens; FillFSLoc is related to MSL2SWL]
FillNumM FillMList    FillFSLoc     FillDens
(-)      (-)             (m)           (kg/m^3)
66      1       2       3       4       5       8       9       10      11      12
    29      30      31      32      33      34      35      36      37      38      39
          40      41      42      47      48      49      50      51      52      53
          54      55      56      57      58      59      60      65      66      67
          68      69      70      71      72      73      74      75      76      77
          78      83      84      85      86      87      88      89      90      91
          92      93      94      95      96      0       1025
---------------------- MARINE GROWTH ----------------------------------------
            0   NMGDepths      - Number of marine-growth depths specified (-)
MGDpth     MGThck        MGDens
(m)        (m)          (kg/m^3)
---------------------- MEMBER OUTPUT LIST -----------------------------------
            0   NMOutputs      - Number of member outputs (-) [must be < 10]
MemberID   NOutLoc    NodeLocs [NOutLoc < 10; node locations are normalized distance
    from the start of the member, and must be >=0 and <= 1] [unused if NMOutputs=0]
  (-)        (-)        (-)
---------------------- JOINT OUTPUT LIST ------------------------------------
            0   NJOutputs      - Number of joint outputs [Must be < 10]
   0          JOutLst        - List of JointIDs which are to be output (-)[unused if
       NJOutputs=0]
---------------------- OUTPUT -----------------------------------------------
True             HDSum          - Output a summary file [flag]
False            OutAll         - Output all user-specified member and joint loads (only
     at each member end, not interior locations) [flag]
            2   OutSwtch       - Output requested channels to: [1=Hydrodyn.out, 2=
                GlueCode.out, 3=both files]
"ES20.12E3"       OutFmt         - Output format for numerical results (quoted string) [
    not checked for validity!]
"A11"            OutSFmt        - Output format for header strings (quoted string) [not
    checked for validity!]
---------------------- OUTPUT CHANNELS --------------------------------------
END of output channels and end of file. (the word "END" must appear in the first 3
    columns of this line)
```

D.9 SubDyn input file

```
----------- SubDyn v1.03.x MultiMember Support Structure Input File --------
INNWIND.EU 10MW Reference (Steel) Jacket SubDyn input properties
-------------------------- SIMULATION CONTROL  -----------------------------
```

```
true              Echo         - Echo input data to "<rootname>.SD.ech" (flag)
0.005             SDdeltaT     - Local Integration Step. If "default", the glue-code
    integration step will be used.
             3    IntMethod    - Integration Method [1/2/3/4 = RK4/AB4/ABM4/AM2].
True              SttcSolve    - Solve dynamics about static equilibrium point
False                     GuyanLoadCorection - Include extra moment from lever arm at
    interface and rotate FEM for floating
-------------------- FEA and CRAIG-BAMPTON PARAMETERS-----------------------
             3    FEMMod       - FEM switch: element model in the FEM. [1= Euler-Bernoulli
                  (E-B);  2=Tapered E-B (unavailable);  3= 2-node Timoshenko;  4= 2-node
                  tapered Timoshenko (unavailable)]
             5    NDiv         - Number of sub-elements per member
True              CBMod        - [T/F] If True perform C-B reduction, else full FEM dofs
    will be retained. If True, select Nmodes to retain in C-B reduced system.
            13    Nmodes       - Number of internal modes to retain (ignored if CBMod=
                 False). If Nmodes=0 --> Guyan Reduction.
             5    JDampings    - Damping Ratios for each retained mode (% of critical) If
                  Nmodes>0, list Nmodes structural damping ratios for each retained mode
                  (% of critical), or a single damping ratio to be applied to all
                  retained modes. (last entered value will be used for all remaining
                  modes).
                         1       GuyanDampMod Guyan damping [0=none, 1=Rayleigh Damping,
                              2= user specified 6x6 matrix]
   0.158963354,0.005616645       RayleighDamp Mass and stiffness proportional damping
       coefficients ((alpha,beta) Rayleigh damping) [only if GuyanDampMod=1]
                         6       GuyanDampSize - Guyan damping matrix size (square, 6x6)
                              [only if GuyanDampMod=2]
0             0             0             0             0             0
0             0             0             0             0             0
0             0             0             0             0             0
0             0             0             0             0             0
0             0             0             0             0             0
0             0             0             0             0             0
---- STRUCTURE JOINTS: joints connect structure members ---
            62    NJoints      - Number of joints (-)
JointID         JointXss        JointYss        JointZss                JointType
            JointDirX               JointDirY               JointDirZ
            JointStiff              [Coordinates of Member joints in SS-Coordinate
    System]
(-)              (m)             (m)                     (m)                     (-)
                     (-)                     (-)                     (-)
            (Nm/rad)
1       -17     17      -48.5   1       0       0       0       0
2       -17     17      -47.5   1       0       0       0       0
3       -16.80366412    16.80366412     -46.214 1  0  0  0  0
4       -13.58381679    13.58381679     -25.124 1  0  0  0  0
5       -10.98229008    10.98229008     -8.084  1  0  0  0  0
6       -8.881526718    8.881526718     5.676   1  0  0  0  0
7       -7.183206107    7.183206107     16.8    1  0  0  0  0
8       -7      7       18      1       0       0       0       0
9       17      17      -48.5   1       0       0       0       0
10      17      17      -47.5   1       0       0       0       0
11      16.80366412     16.80366412     -46.214 1       0  0    0  0
12      13.58381679     13.58381679     -25.124 1       0  0    0  0
13      10.98229008     10.98229008     -8.084  1       0  0    0  0
14      8.881526718     8.881526718     5.676   1       0  0    0  0
15      7.183206107     7.183206107     16.8    1       0  0    0  0
16      7       7       18      1       0       0       0       0
17      -15.02315348    0       -34.513 1       0       0       0       0
18      -12.14530387    0       -15.704 1       0       0       0       0
19      -9.820821823    0       -0.509  1       0       0       0       0
20      -7.942595455    0       11.901  1       0       0       0       0
21      0       15.02315348     -34.513 1       0       0       0       0
22      0       12.14530387     -15.704 1       0       0       0       0
23      0       9.820821823     -0.509  1       0       0       0       0
24      0       7.942595455     11.901  1       0       0       0       0
25      15.02315348     0       -34.513 1       0       0       0       0
26      12.14530387     0       -15.704 1       0       0       0       0
27      9.820821823     0       -0.509  1       0       0       0       0
28      7.942595455     0       11.901  1       0       0       0       0
29      -16.80366412    0       -46.214 1       0       0       0       0
30      16.80366412     0       -46.214 1       0       0       0       0
31      0       16.80366412     -46.214 1       0       0       0       0
```

```
32      0       -16.80366412    -46.214 1       0       0       0       0
33      -17     -17     -48.5   1       0       0       0       0
34      -17     -17     -47.5   1       0       0       0       0
35      -16.80366412    -16.80366412    -46.214 1       0       0       0       0
36      -13.58381679    -13.58381679    -25.124 1       0       0       0       0
37      -10.98229008    -10.98229008    -8.084  1       0       0       0       0
38      -8.881526718    -8.881526718    5.676   1       0       0       0       0
39      -7.183206107    -7.183206107    16.8    1       0       0       0       0
40      -7      -7      18      1       0       0       0       0
41      17      -17     -48.5   1       0       0       0       0
42      17      -17     -47.5   1       0       0       0       0
43      16.80366412     -16.80366412    -46.214 1       0       0       0       0
44      13.58381679     -13.58381679    -25.124 1       0       0       0       0
45      10.98229008     -10.98229008    -8.084  1       0       0       0       0
46      8.881526718     -8.881526718    5.676   1       0       0       0       0
47      7.183206107     -7.183206107    16.8    1       0       0       0       0
48      7       -7      18      1       0       0       0       0
49      0       -15.02315348    -34.513 1       0       0       0       0
50      0       -12.14530387    -15.704 1       0       0       0       0
51      0       -9.820821823    -0.509  1       0       0       0       0
52      0       -7.942595455    11.901  1       0       0       0       0
53      0       0       18      1       0       0       0       0
54      -6.389313       6.389313        22      1       0       0       0       0
55      -2.9344931      2.9344931       26      1       0       0       0       0
56      2.9344931       2.9344931       26      1       0       0       0       0
57      6.3893123       6.389313        22      1       0       0       0       0
58      -6.389313       -6.389313       22      1       0       0       0       0
59      -2.9344931      -2.9344931      26      1       0       0       0       0
60      2.9344931       -2.9344931      26      1       0       0       0       0
61      6.389313        -6.389313       22      1       0       0       0       0
62      0       0       26      1       0       0       0       0
-- BASE REACTION JOINTS: 1/0 for Locked/Free DOF @ each Reaction Node ------
            4   NReact      - Number of Joints with reaction forces; be sure to remove
                all rigid motion DOFs of the structure  (else det([K])=[0])
RJointID   RctTDXss    RctTDYss    RctTDZss    RctRDXss    RctRDYss    RctRDZss
    SSIfile [Global Coordinate System]
  (-)       (flag)      (flag)      (flag)      (flag)      (flag)      (flag)
      (string)
1       0         0         0         0         0         0    "SSI.txt"
9       0         0         0         0         0         0        "SSI.txt"
33      0         0         0         0         0         0        "SSI.txt"
41      0         0         0         0         0         0        "SSI.txt"
------- INTERFACE JOINTS: 1/0 for Locked (to the TP)/Free DOF @each Interface Joint (
    only Locked-to-TP implemented thus far (=rigid TP)) ---------
            1   NInterf     - Number of interface joints locked to the Transition Piece
                 (TP):  be sure to remove all rigid motion dofs
IJointID   ItfTDXss    ItfTDYss    ItfTDZss    ItfRDXss    ItfRDYss    ItfRDZss     [
    Global Coordinate System]
  (-)       (flag)      (flag)      (flag)      (flag)      (flag)      (flag)
  62          1           1           1           1           1           1
----------------------------------- MEMBERS -------------------------------
           117   NMembers    - Number of frame members
MemberID   MJointID1   MJointID2   MPropSetID1   MPropSetID2    MType     COSMID
  (-)         (-)         (-)          (-)           (-)          (-)          (-)
1       1       2       1       1       1
2       2       3       1       1       1
3       3       4       2       2       1
4       4       5       3       3       1
5       5       6       4       4       1
6       6       7       5       5       1
7       7       8       6       6       1
8       9       10      1       1       1
9       10      11      1       1       1
10      11      12      2       2       1
11      12      13      3       3       1
12      13      14      4       4       1
13      14      15      5       5       1
14      15      16      6       6       1
15      33      34      1       1       1
16      34      35      1       1       1
17      35      36      2       2       1
18      36      37      3       3       1
19      37      38      4       4       1
```

20	38	39	5	5	1
21	39	40	6	6	1
22	41	42	1	1	1
23	42	43	1	1	1
24	43	44	2	2	1
25	44	45	3	3	1
26	45	46	4	4	1
27	46	47	5	5	1
28	47	48	6	6	1
29	3	29	7	7	1
30	29	35	7	7	1
31	3	17	8	8	1
32	35	17	8	8	1
33	4	17	8	8	1
34	36	17	8	8	1
35	4	18	9	9	1
36	36	18	9	9	1
37	5	18	9	9	1
38	37	18	9	9	1
39	5	19	10	10	1
40	37	19	10	10	1
41	6	19	10	10	1
42	38	19	10	10	1
43	6	20	11	11	1
44	38	20	11	11	1
45	7	20	11	11	1
46	39	20	11	11	1
47	11	30	7	7	1
48	30	43	7	7	1
49	11	25	8	8	1
50	43	25	8	8	1
51	12	25	8	8	1
52	44	25	8	8	1
53	12	26	9	9	1
54	44	26	9	9	1
55	13	26	9	9	1
56	45	26	9	9	1
57	13	27	10	10	1
58	45	27	10	10	1
59	14	27	10	10	1
60	46	27	10	10	1
61	14	28	11	11	1
62	46	28	11	11	1
63	15	28	11	11	1
64	47	28	11	11	1
65	3	31	7	7	1
66	11	31	7	7	1
67	3	21	8	8	1
68	11	21	8	8	1
69	4	21	8	8	1
70	12	21	8	8	1
71	4	22	9	9	1
72	12	22	9	9	1
73	5	22	9	9	1
74	13	22	9	9	1
75	5	23	10	10	1
76	13	23	10	10	1
77	6	23	10	10	1
78	14	23	10	10	1
79	6	24	11	11	1
80	14	24	11	11	1
81	7	24	11	11	1
82	15	24	11	11	1
83	35	32	7	7	1
84	43	32	7	7	1
85	35	49	8	8	1
86	43	49	8	8	1
87	36	49	8	8	1
88	44	49	8	8	1
89	36	50	9	9	1
90	44	50	9	9	1
91	37	50	9	9	1
92	45	50	9	9	1

```
93      37      51      10      10      1
94      45      51      10      10      1
95      38      51      10      10      1
96      46      51      10      10      1
97      38      52      11      11      1
98      46      52      11      11      1
99      39      52      11      11      1
100     47      52      11      11      1
101     8       53      12      12      1
102     16      53      12      12      1
103     40      53      12      12      1
104     48      53      12      12      1
105     53      62      13      13      1
106     8       54      12      12      1
107     16      57      12      12      1
108     40      58      12      12      1
109     48      61      12      12      1
110     55      62      12      12      1
111     56      62      12      12      1
112     59      62      12      12      1
113     60      62      12      12      1
114     54      55      12      12      1
115     57      56      12      12      1
116     58      59      12      12      1
117     61      60      12      12      1
------------------ MEMBER X-SECTION PROPERTY data 1/2 [isotropic material for now: use
    this table for circular-tubular elements] ------------------------
             13   NPropSets   - Number of structurally unique x-sections (i.e. how many
                 groups of X-sectional properties are utilized throughout all of the
                 members)
PropSetID     YoungE          ShearG          MatDens          XsecD           XsecT
  (-)         (N/m2)          (N/m2)          (kg/m3)           (m)             (m)
1       2.10E+11        8.08E+10        7850    1.40    0.12
2       2.10E+11        8.08E+10        7850    1.40    0.07
3       2.10E+11        8.08E+10        7850    1.40    0.042
4       2.10E+11        8.08E+10        7850    1.40    0.042
5       2.10E+11        8.08E+10        7850    1.40    0.042
6       2.10E+11        8.08E+10        7850    1.40    0.066
7       2.10E+11        8.08E+10        7850    1.04    0.02
8       2.10E+11        8.08E+10        7850    1.06    0.03
9       2.10E+11        8.08E+10        7850    0.936   0.018
10      2.10E+11        8.08E+10        7850    0.84    0.02
11      2.10E+11        8.08E+10        7850    0.832   0.016
12      1.05E+12        4.04E+11        7850    1.40    0.08
13      1.05E+12        4.04E+11        7850    8.3     0.08
------------------ MEMBER X-SECTION PROPERTY data 2/2 [isotropic material for now: use
    this table if any section other than circular, however provide COSM(i,j) below]
    ------------------------
             0   NXPropSets  - Number of structurally unique non-circular x-sections (if
                  0 the following table is ignored)
PropSetID     YoungE          ShearG          MatDens          XsecA           XsecAsx
          XsecAsy       XsecJxx       XsecJyy        XsecJ0
  (-)         (N/m2)          (N/m2)          (kg/m3)          (m2)             (m2)
              (m2)           (m4)          (m4)           (m4)
-------------------------- CABLE PROPERTIES ---------------------------------
0 NCablePropSets - Number of cable cable properties
PropSetID EA MatDens T0 CtrlChannel
(-) (N) (kg/m) (N) (-)
----------------------- RIGID LINK PROPERTIES ------------------------------
0 NRigidPropSets - Number of rigid link properties
PropSetID MatDens
(-) (kg/m)
---------------------- MEMBER COSINE MATRICES COSM(i,j) ------------------------
             0   NCOSMs      - Number of unique cosine matrices (i.e., of unique member
                 alignments including principal axis rotations); ignored if NXPropSets=0
                    or 9999 in any element below
COSMID    COSM11    COSM12    COSM13    COSM21    COSM22    COSM23    COSM31    COSM32
       COSM33
 (-)        (-)       (-)       (-)       (-)       (-)       (-)       (-)       (-)
           (-)
------------------------ JOINT ADDITIONAL CONCENTRATED MASSES---------------
             0   NCmass      - Number of joints with concentrated masses; Global
                 Coordinate System
```

```
CMJointID      JMass           JMXX             JMYY             JMZZ         JMXY
            JMXZ                  JMYZ     MCGX           MCGY     MCGZ
  (-)           (kg)          (kg*m^2)         (kg*m^2)         (kg*m^2)       (kg*m^2)
        (kg*m^2)    (kg*m^2)        (kg*m^2)  (kg*m^2)  (kg*m^2)
---------------------------- OUTPUT: SUMMARY & OUTFILE ---------------------
False           SDSum       - Output a Summary File (flag).It contains: matrices K,M
    and C-B reduced M_BB, M-BM, K_BB, K_MM(OMG^2), PHI_R, PHI_L. It can also contain
    COSMs if requested.
False           OutCOSM     - Output cosine matrices with the selected output member
    forces (flag)
False           OutAll      - [T/F] Output all members' end forces
             1   OutSwtch    - [1/2/3] Output requested channels to: 1=<rootname>.SD.out
                 ;  2=<rootname>.out (generated by FAST);  3=both files.
True            TabDelim    - Generate a tab-delimited output in the <rootname>.SD.out
    file
             1   OutDec      - Decimation of output in the <rootname>.SD.out file
"ES20.12E3"       OutFmt      - Output format for numerical results in the <rootname>.SD
    .out file
"A11"           OutSFmt     - Output format for header strings in the <rootname>.SD.out
     file
------------------------- MEMBER OUTPUT LIST ------------------------------
             1   NMOutputs   - Number of members whose forces/displacements/velocities/
                 accelerations will be output (-) [Must be <= 9].
MemberID   NOutCnt    NodeCnt [NOutCnt=how many nodes to get output for [< 10]; NodeCnt
    are local ordinal numbers from the start of the member, and must be >=1 and <= NDiv
    +1] If NMOutputs=0 leave blank as well.
  (-)        (-)        (-)
  1                     1                       1
------------------------- SDOutList: The next line(s) contains a list of output
    parameters that will be output in <rootname>.SD.out or <rootname>.out. ------
"ReactFXss, ReactFYss, ReactFZss" - Base reactions: fore-aft shear, side-to-side shear,
    side-to-side moment, fore-aft moment, yaw moment, vertical force
"M1N1TDxss"
"ReactMXss, ReactMYss, ReactMZss"
END of output channels and end of file. (the word "END" must appear in the first 3
    columns of this line)
```

D.9.1 SSI stiffness file

```
!---------------- Pile Head K and M elements -------------------!
!Equivalent Stiffness Constants: Kxx, Kyy, Kzz, Kxtx, Kxty..Kztx,Kzty,Kztz  in any order
    ; max 21 elements
4.69155E+08                    Kxx
0.00000E+00                    Kxy
4.69155E+08                    Kyy
0.00000E+00                    Kxz
0.00000E+00                    Kyz
2.44494E+09                    Kzz
0.00000E+00                    Kxtx
1.93452E+09                    Kytx
0.00000E+00                    Kztx
1.52446E+10                    Ktxtx
-1.93452E+09                   Kxty
0.00000E+00                    Kyty
0.00000E+00                    Kzty
0.00000E+00                    Ktxty
1.52446E+10                    Ktyty
0.00000E+00                    Kxtz
0.00000E+00                    Kytz
0.00000E+00                    Kztz
0.00000E+00                    Ktxtz
0.00000E+00                    Ktytz
3.96802E+09                    Ktztz
```

D.10 TurbSim input file

```
---------TurbSim v2.00.* Input File------------------------
for Certification Test #1 (Kaimal Spectrum, formatted FF files).
```

```
---------Runtime Options-----------------------------------
False         Echo            - Echo input data to <RootName>.ech (flag)
     123456   RandSeed1       - First random seed  (-2147483648 to 2147483647)
     789012   RandSeed2       - Second random seed (-2147483648 to 2147483647) for
         intrinsic pRNG, or an alternative pRNG: "RanLux" or "RNSNLW"
False         WrBHHTP         - Output hub-height turbulence parameters in binary form?
      (Generates RootName.bin)
False         WrFHHTP         - Output hub-height turbulence parameters in formatted
    form?  (Generates RootName.dat)
False         WrADHH          - Output hub-height time-series data in AeroDyn form?  (
    Generates RootName.hh)
True          WrADFF          - Output full-field time-series data in TurbSim/AeroDyn
    form? (Generates RootName.bts)
False         WrBLFF          - Output full-field time-series data in BLADED/AeroDyn
    form?  (Generates RootName.wnd)
True          WrADTWR         - Output tower time-series data? (Generates RootName.twr)
False         WrFMTFF         - Output full-field time-series data in formatted (
    readable) form?  (Generates RootName.u, RootName.v, RootName.w)
False         WrACT           - Output coherent turbulence time steps in AeroDyn form? (
    Generates RootName.cts)
True          Clockwise       - Clockwise rotation looking downwind? (used only for full
    -field binary files - not necessary for AeroDyn)
          0   ScaleIEC        - Scale IEC turbulence models to exact target standard
              deviation? [0=no additional scaling; 1=use hub scale uniformly; 2=use
              individual scales]
--------Turbine/Model Specifications-----------------------
         60   NumGrid_Z       - Vertical grid-point matrix dimension
         60   NumGrid_Y       - Horizontal grid-point matrix dimension
      0.005   TimeStep        - Time step [seconds]
        300   AnalysisTime    - Length of analysis time series [seconds] (program will
            add time if necessary: AnalysisTime = MAX(AnalysisTime, UsableTime+GridWidth
            /MeanHHWS) )
                200   UsableTime      - Usable length of output time series [seconds] (
                    program will add GridWidth/MeanHHWS seconds unless UsableTime is "
                    ALL")
     131.63   HubHt           - Hub height [m] (should be > 0.5*GridHeight)
        210   GridHeight      - Grid height [m]
        210   GridWidth       - Grid width [m] (should be >= 2*(RotorRadius+ShaftLength)
            )
          0   VFlowAng        - Vertical mean flow (uptilt) angle [degrees]
          0   HFlowAng        - Horizontal mean flow (skew) angle [degrees]
--------Meteorological Boundary Conditions-------------------
"IECKAI"      TurbModel       - Turbulence model ("IECKAI","IECVKM","GP_LLJ","NWTCUP","
    SMOOTH","WF_UPW","WF_07D","WF_14D","TIDAL","API","USRINP","TIMESR", or "NONE")
"unused"      UserFile        - Name of the file that contains inputs for user-defined
    spectra or time series inputs (used only for "USRINP" and "TIMESR" models)
          3   IECstandard     - Number of IEC 61400-x standard (x=1,2, or 3 with
              optional 61400-1 edition number (i.e. "1-Ed2") )
"C"           IECturbc        - IEC turbulence characteristic ("A", "B", "C" or the
    turbulence intensity in percent) ("KHTEST" option with NWTCUP model, not used for
    other models)
"NTM"         IEC_WindType    - IEC turbulence type ("NTM"=normal, "xETM"=extreme
    turbulence, "xEWM1"=extreme 1-year wind, "xEWM50"=extreme 50-year wind, where x=wind
     turbine class 1, 2, or 3)
"default"     ETMc            - IEC Extreme Turbulence Model "c" parameter [m/s]
"default"     WindProfileType - Velocity profile type ("LOG";"PL"=power law;"JET";"H2L"=
    Log law for TIDAL model;"API";"USR";"TS";"IEC"=PL on rotor disk, LOG elsewhere; or "
    default")
"unused"      ProfileFile     - Name of the file that contains input profiles for
    WindProfileType="USR" and/or TurbModel="USRVKM" [-]
     131.63   RefHt           - Height of the reference velocity (URef) [m]
         11   URef            - Mean (total) velocity at the reference height [m/s] (or
             "default" for JET velocity profile) [must be 1-hr mean for API model;
             otherwise is the mean over AnalysisTime seconds]
        350   ZJetMax         - Jet height [m] (used only for JET velocity profile,
            valid 70-490 m)
"default"     PLExp           - Power law exponent [-] (or "default")
"default"     Z0              - Surface roughness length [m] (or "default")
--------Non-IEC Meteorological Boundary Conditions------------
"default"     Latitude        - Site latitude [degrees] (or "default")
       0.05   RICH_NO         - Gradient Richardson number [-]
"default"     UStar           - Friction or shear velocity [m/s] (or "default")
"default"     ZI              - Mixing layer depth [m] (or "default")
```

```
"default"     PC_UW           - Hub mean u'w' Reynolds stress [m^2/s^2] (or "default" or
      "none")
"default"     PC_UV           - Hub mean u'v' Reynolds stress [m^2/s^2] (or "default" or
      "none")
"default"     PC_VW           - Hub mean v'w' Reynolds stress [m^2/s^2] (or "default" or
      "none")
--------Spatial Coherence Parameters----------------------------
"default"     SCMod1           - u-component coherence model ("GENERAL","IEC","API","
    NONE", or "default")
"default"     SCMod2           - v-component coherence model ("GENERAL","IEC","NONE", or
      "default")
"default"     SCMod3           - w-component coherence model ("GENERAL","IEC","NONE", or
      "default")
"default"     InCDec1          - u-component coherence parameters for general or IEC
    models [-, m^-1] (e.g. "10.0  0.3e-3" in quotes) (or "default")
"default"     InCDec2          - v-component coherence parameters for general or IEC
    models [-, m^-1] (e.g. "10.0  0.3e-3" in quotes) (or "default")
"default"     InCDec3          - w-component coherence parameters for general or IEC
    models [-, m^-1] (e.g. "10.0  0.3e-3" in quotes) (or "default")
"default"     CohExp           - Coherence exponent for general model [-] (or "default")
--------Coherent Turbulence Scaling Parameters-------------------
".\EventData"    CTEventPath     - Name of the path where event data files are located
"random"      CTEventFile     - Type of event files ("LES", "DNS", or "RANDOM")
true          Randomize       - Randomize the disturbance scale and locations? (true/
    false)
         1   DistScl         - Disturbance scale [-] (ratio of event dataset height to
             rotor disk). (Ignored when Randomize = true.)
       0.5   CTLy            - Fractional location of tower centerline from right [-] (
           looking downwind) to left side of the dataset. (Ignored when Randomize =
           true.)
       0.5   CTLz            - Fractional location of hub height from the bottom of the
            dataset. [-] (Ignored when Randomize = true.)
        30   CTStartTime     - Minimum start time for coherent structures in RootName.
            cts [seconds]
```

D.11 BModes input file

```
======================   BModes v3.00 Main Input File  ==================
10MW Tower
--------- General parameters ---------------------------------------------------
true      Echo        Echo input file contents to *.echo file if true.
2         beam_type   1: blade, 2: tower (-)
0.0       romg:       rotor speed, automatically set to zero for tower modal analysis (
    rpm)
1.0       romg_mult:  rotor speed muliplicative factor (-)
131.63    radius:     rotor tip radius measured along coned blade axis, OR tower height
    above ground level [onshore] or MSL [offshore](m)
26.0      hub_rad:    hub radius measured along coned blade axis OR tower rigid-base
    height (m)
0.0       precone:    built-in precone angle, automatically set to zero for a tower (deg
    )
0.0       bl_thp:     blade pitch setting, automatically set to zero for a tower (deg)
2         hub_conn:   hub-to-blade or tower-base boundary condition [1: cantilevered; 2:
     free-free; 3: only axial and torsion constraints] (-)
20        modepr:     number of modes to be printed (-)
t         TabDelim    (true: tab-delimited output tables; false: space-delimited tables)
f         mid_node_tw  (true: output twist at mid-node of elements; false: no mid-node
    outputs)

--------- Blade-tip or tower-top mass properties ------------------------------
866555.062         tip_mass    blade-tip or tower-top mass (kg)
0                          cm_loc      tip-mass c.m. offset from the tower axis measured
     along x-tower axis (m)
0                      cm_axial    tip-mass c.m. offset tower tip measures axially along
     the z axis (m)
2.40016659E+8      ixx_tip     blade lag mass moment of inertia about the tip-section x
    reference axis (kg-m^2)
1.42102115E+8      iyy_tip     blade flap mass moment of inertia about the tip-section y
     reference axis (kg-m^2)
1.118464128E+8     izz_tip     torsion mass moment of inertia about the tip-section z
    reference axis (kg-m^2)
```

```
0                 ixy_tip     cross product of inertia about x and y reference axes(kg-
    m^2)
0                 izx_tip     cross product of inertia about z and x reference axes(kg-
    m^2)
0                 iyz_tip     cross product of inertia about y and z reference axes(kg-
    m^2)

--------- Distributed-property identifiers ----------------------------------
1         id_mat:     material_type [1: isotropic; non-isotropic composites option not
    yet available]
'Tower_props.dat'   : sec_props_file   name of beam section properties file (-)

Property scaling factors.............................
1.0       sec_mass_mult:   mass density multiplier (-)
1.0       flp_iner_mult:   blade flap or tower f-a inertia multiplier (-)
1.0       lag_iner_mult:   blade lag or tower s-s inertia multiplier (-)
1.0       flp_stff_mult:   blade flap or tower f-a bending stiffness multiplier (-)
1.0       edge_stff_mult:  blade lag or tower s-s bending stiffness multiplier (-)
1.0       tor_stff_mult:   torsion stiffness multiplier (-)
1.0       axial_stff_mult: axial stiffness multiplier (-)
1.0       cg_offst_mult:   cg offset multiplier (-)
1.0       sc_offst_mult:   shear center multiplier (-)
1.0       tc_offst_mult:   tension center multiplier (-)

--------- Finite element discretization ---------------------------------------
50        nselt:     no of blade or tower elements (-)
Distance of element boundary nodes from blade or flexible-tower root (normalized wrt
    blade or tower length), el_loc()
0.0000  0.0200  0.0400  0.0600  0.0800  0.1000  0.1200  0.1400  0.1600  0.1800  0.2000
    0.2200  0.2400  0.2600  0.2800  0.3000  0.3200  0.3400  0.3600  0.3800  0.4000
    0.4200  0.4400  0.4600  0.4800  0.5000  0.5200  0.5400  0.5600  0.5800  0.6000
    0.6200  0.6400  0.6600  0.6800  0.7000  0.7200  0.7400  0.7600  0.7800  0.8000
    0.8200  0.8400  0.8600  0.8800  0.9000  0.9200  0.9400  0.9600  0.9800  1.0000

------ Properties of tower support subsystem (read only if beam_type is 2) ---
1          tow_support: : aditional tower support [0: no additional support; 1: floating
    -platform or monopile with or without tension wires] (-)
0.0        draft        : depth of tower base from the ground or the MSL (mean sea level
    ) (m)
0.0        cm_pform     : distance of platform c.m. below the MSL (m)
0          mass_pform   : platform mass (kg)
Platform mass inertia 3X3 matrix (i_matrix_pform):
0.   0.   0.
0.   0.   0.
0.   0.   0.
-26.0        ref_msl    : distance of platform reference point below the MSL (m)
Platform-reference-point-referred hydrodynamic 6X6 matrix (hydro_M):
 6.111259E+05   0.000000E+00   0.000000E+00   0.000000E+00  -4.330388E+06   0.000000E+00
 0.000000E+00   6.111259E+05   0.000000E+00   4.330388E+06   0.000000E+00   0.000000E+00
 0.000000E+00   0.000000E+00   5.312108E+05   0.000000E+00   0.000000E+00   0.000000E+00
 0.000000E+00   4.330388E+06   0.000000E+00   5.293475E+07   0.000000E+00   0.000000E+00
-4.330388E+06   0.000000E+00   0.000000E+00   0.000000E+00   5.293475E+07   0.000000E+00
 0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00   2.189853E+07
Platform-reference-point-referred hydrodynamic 6X6 stiffness matrix (hydro_K):
 1.447757E+08   0.000000E+00   0.000000E+00   0.000000E+00  -2.444356E+09   0.000000E+00
 0.000000E+00   1.447757E+08   0.000000E+00   2.444356E+09   0.000000E+00   0.000000E+00
 0.000000E+00   0.000000E+00   1.659739E+09   0.000000E+00   0.000000E+00   0.000000E+00
 0.000000E+00   2.444356E+09   0.000000E+00   1.893857E+11   0.000000E+00   0.000000E+00
-2.444356E+09   0.000000E+00   0.000000E+00   0.000000E+00   1.893857E+11   0.000000E+00
 0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00   3.653442E+10
Mooring-system 6X6 stiffness matrix (mooring_K):
0.0     0.0     0.0     0.0     0.0      0.0
0.0     0.0     0.0     0.0     0.0      0.0
0.0     0.0     0.0     0.0     0.0      0.0
0.0     0.0     0.0     0.0     0.0      0.0
0.0     0.0     0.0     0.0     0.0      0.0
0.0     0.0     0.0     0.0     0.0      0.0

Distributed (hydrodynamic) added-mass per unit length along a flexible portion of the
    tower length:
0           n_secs_m_distr: number of sections at which added mass per unit length is
    specified (-)
0.  0.    : z_distr_m [row array of size n_added_m_pts; section locations wrt the
```

```
    flexible tower base over which distributed mass is specified] (m)
0.  0.    : distr_m [row array of size n_added_m_pts; added distributed masses per unit
    length] (kg/m)

Distributed elastic stiffness per unit length along a flexible portion of the tower
    length:
0           n_secs_k_distr: number of points at which distributed stiffness per unit
    length is specified (-)
0       1       2       3       4       5       6       7       8       9       10
    11      12      13      14      15      16      17      18      19      20      21
          22      23      24      25      26      27      28      29      30      31
          32      33      34      35      36  : z_distr_k [row array of size n_added_m_
    pts; section locations wrt the flexible tower base over which distributed stiffness
    is specified] (m)
595318000.0     1165208000      1129400000      1095553000      1059931000
    1024493000      989209000       953643000       918718000       883287000
    847803000       812541000       777187000       741870000       706616000
    671440000       636229000       600957000       565919000       530470000
    495081000       459574000       385327000       305479000       280059000
    254125000       227500000       200112000       171927000       143115000
    114173000       80184000        52237000        35561000        20912000
    9000000 1156000   : distr_k [row array of size n_added_m_pts; distributed stiffness
    per unit length] (N/m^2)

Tension wires data
0          n_attachments: no of wire-attachment locations on tower [0: no tension wires]
    (-)
3 3        n_wires:       no of wires attached at each location (must be 3 or higher) (-)
6 9        node_attach:   node numbers of attacments location (node number must be more
    than 1 and less than nselt+2) (-)
0.e0 0.e0 wire_stfness:  wire spring constant in each set (see users' manual) (N/m)
0. 0.      th_wire:       angle of tension wires (wrt the horizontal ground plane) at
    each attachment point (deg)

END of Main Input File Data ***************************************************
```

D.11.1 Tower properties

```
Tower section properties
11         n_secs:     number of tower sections at which properties are specified (-)

sec_loc  str_tw  tw_iner   mass_den  flp_iner    edge_iner  flp_stff     edge_stff   tor
    _stff   axial_stff  cg_offst  sc_offst tc_offst
(-)       (deg)     (deg)     (kg/m)     (kg-m)             (kg-m)     (Nm^2)      (Nm^2)
         (Nm^2)       (N)         (m)        (m)       (m)
0                       0       0       15383.91        2.6052E+05       2.6052E+05
    3.2182E+12      3.2182E+12      4.5974E+12      8.6091E+19      0       0       0
0.099498249     0       0       14860.52        2.3482E+05      2.3482E+05      2.9008E
    +12      2.9008E+12      4.1439E+12      7.7600E+19      0       0       0
0.198901827     0       0       13321.73        1.9620E+05      1.9620E+05      2.4236E
    +12      2.4236E+12      3.4623E+12      6.4835E+19      0       0       0
0.298400076     0       0       11856.37        1.6232E+05      1.6232E+05      2.0052E
    +12      2.0052E+12      2.8645E+12      5.3642E+19      0       0       0
0.397803654     0       0       11423.77        1.4520E+05      1.4520E+05      1.7936E
    +12      1.7936E+12      2.5623E+12      4.7982E+19      0       0       0
0.497301903     0       0       10067.90        1.1828E+05      1.1828E+05      1.4611E
    +12      1.4611E+12      2.0873E+12      3.9088E+19      0       0       0
0.596705481     0       0       8785.46         9.5100E+04      9.5100E+04      1.1748E
    +12      1.1748E+12      1.6782E+12      3.1427E+19      0       0       0
0.69620373      0       0       7576.46         7.5301E+04      7.5301E+04      9.3018E
    +11      9.3018E+11      1.3288E+12      2.4884E+19      0       0       0
0.795607309     0       0       5640.45         5.1360E+04      5.1360E+04      6.3444E
    +11      6.3444E+11      9.0635E+11      1.6972E+19      0       0       0
0.895105557     0       0       4614.37         3.8274E+04      3.8274E+04      4.7280E
    +11      4.7280E+11      6.7543E+11      1.2648E+19      0       0       0
1                       0       0       4382.05         3.2780E+04       3.2780E+04
    4.0493E+11      4.0493E+11      5.7847E+11      1.0832E+19      0       0       0


**Note: If the above data represents TOWER properties, the following are overwritten:
```

```
str_tw is set to zero
tw_iner is set to zero
cg_offst is set to zero
sc_offst is set to zero
tc_offst is set to zero
edge_iner is set equal to flp_iner
edge_stff is set equal to flp_stff
```