Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Gjermund Smedsland

Fatigue Estimation of Offshore Structure by Monitored Data and Machine Learning

Master's thesis in Marine Technology
Supervisor: Svein Sævik
Co-supervisor: Ole Gabrielsen
June 2022

NTNU
Norwegian University of
Science and Technology

Gjermund Smedsland

Fatigue Estimation of Offshore Structure by Monitored Data and Machine Learning

Master's thesis in Marine Technology
Supervisor: Svein Sævik
Co-supervisor: Ole Gabrielsen
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU
Norwegian University of
Science and Technology

NTNU Trondheim
Norwegian University of Science and Technology *Faculty of Engineering*
Department of Marine Technology

MASTER THESIS WORK SPRING 2022
for

Stud. Tech. Gjermund Smedsland

Fatigue Estimation of Offshore Structure by Monitored Data and Machine Learning

Estimering av havkonstruksjonsutmatting med bruk av måledata og maskinlæring

Following the excessive developments of low-cost sensor systems, on-line monitoring of offshore structures has become an important factor in lifecycle asset management. When it comes to offshore structure fatigue, the measured data need to be converted to hot-spot stress in order to evaluate the Miner sum based on available S-N curves for the specific structural detail. The traditional way of doing this is by performing time consuming FE analysis. An alternative approach to improve computation efficiency is by applying machine learning algorithms, which is the motivation for the present master thesis, The master thesis work is to be carried out as a continuation of the project performed during Fall 2021 and will include the following items:

1. Supplementary literature review into fatigue analysis of offshore structures and machine learning algorithms, including relevant guidelines and standards in order to further support the methodology description outlined during the project work.
2. Familiarize with the established FE model provided by DnV/Aker BP.
3. Familiarize with the acceleration and weather data provided by DnV/Aker BP.
4. Train and validate the machine learning model
5. Conclusions and recommendations for further work

It is assumed that necessary FE models and monitored data are provided by DnV/Aker BP

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

In the master's thesis, the candidate shall present his personal contribution to the resolution of problems within the scope of the master's thesis work

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

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

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

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

The supervisors may require that the candidate, in an early stage of the work, presents a written plan for the completion of the work.

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

The report shall be submitted in electronic format (.pdf):
- Signed by the candidate
- The text defining the scope shall be included (this document)
- Drawings and/or computer models that are not suited to be part of the report in terms of appendices shall be provided on separate (.zip) files.

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

Thesis supervisors:

Prof. Svein Sævik, NTNU, svein.savik@ntnu.no
Team Leader Ole Gabrielsen, Ole.Gabrielsen@dnv.com

Deadline: As decided on the web

Trondheim, January 2022

Svein Sævik

Gjermund Smedsland: 15.03.2022 Gjermund Smedsland

Preface

This master's thesis is the closing submission for a master's degree within Marine Structures, and summarizes the work performed in TMR4930 Marine technology - Master's Thesis at the Norwegian University of Science and Technology (NTNU). The work was carried out in the time period from January 2022 to June 2022, and is a continuation of the project report written during the Autumn of 2021, mainly contributing to develop a method for the master's thesis. The work has been a continuous process carried out under supervision of Ole Gabrielsen, who suggested this thesis, and Professor Svein Sævik, who helped defining the scope.

The motivation behind the topic of this thesis was to familiarize with machine learning, and investigate the applicability related to common problems faced within the offshore industry. By doing so, the goal was to establish a foundation within several aspects considered important for the future, as finite element modelling and -analyses, and data processing.

I would like to express my gratitude to Ole Gabrielsen for giving me the opportunity to write a thesis in collaboration with DNV. Further, I appreciate his ability to propose a task based on my personal wishes, and for his guidance throughout this fall. Additionally, I would like to thank my supervisor at NTNU, Svein Sævik. His broad field of expertise has proven valuable, and his wish to be my supervisor, despite the late confrontation, was a prerequisite for the realization of this thesis. I also wish to thank Atle Nøsen from Aker BP for providing monitored data and sufficient structural information to develop a meaningful thesis. Despite their tight schedules, their willingness to prioritize the guidance hours has been helpful and admirable.

Lastly, I would like to thank my fellow students Borgar, Eirik, Henrik, Hermann and Snorre for creating a supportive work environment.

Trondheim, 9th June 2022

Gjermund Smedsland

Abstract

As an increasing amount of offshore platforms are passing their initially intended design life, fatigue failure is responsible for a substantial amount of repairs in the North Sea. Over the past couple of years, monitoring instruments has become increasingly accurate and inexpensive, where, in addition to wave elevation, displacements of large structures can be determined by using a limited number of sensors. By combining monitored data with machine learning, a trained neural network may estimate the real-time fatigue damage, and contribute to reduce the usage of finite element analyses.

This thesis presents a method on how a neural network can be built, trained and tested to predict the accumulated fatigue damage, within reasonable accuracy, in critical joints on a jacket platform located in the North Sea. The neural network was created by use of the programming language Python, and more specifically through the libraries TensorFlow and Keras.

To properly train- and test the neural network, labeled data needs to be constructed through state-of-the-art fatigue damage calculation methods. For this purpose, a finite element model and on-site measurements of structural displacements and waves was provided by Aker BP. In addition, measurements of wind speed and -direction was extracted from Meteorologisk institutt. Subsequently, a sufficient amount of dynamic time domain fatigue simulations was performed in USFOS, on a stripped version of the original finite element model. To apply site-relevant environmental actions, statistical evaluations of the monitored wave elevation time-series and extracted wind speed was made. The labeled dataset was constructed by combining the environmental actions with the statistical evaluations of the simulated displacement mimicking the monitored. To optimize the training of the neural network, data cleaning and -scaling was performed.

A quite simple feedforward neural network was developed, and provided promising indications that a feedforward neural network indeed is capable of quite accurate predictions of fatigue damage, based on a finite element model subjected to both irregular waves and wind from different headings. The robustness was confirmed by evaluating predictions made with different types of activation functions and optimizers, in addition to removal of certain input features. For the applied neural network, data scaling and removal of outliers was a prerequisite to obtain accurate results. Hence, the limitation in terms of industrial application seems to lie within the accuracy of the performed finite element analyses and fatigue calculations.

Sammendrag

Som en konsekvens av at et økende antall offshore-plattformer passerer sin initielt tiltenkte levetid, er utmattingssvikt ansvarlig for et betydelig antall reparasjoner i Nordsjøen. I løpet av de siste årene har monitoreringsinstrumenter blitt stadig mer nøyaktige og rimelige, hvor man, i tillegg til bølgehøyde, kan måle forskyvninger til store strukturer ved bruk av et begrenset antall sensorer. Ved å kombinere monitorerte data med maskinlæring kan behovet for elementmetodeanalyser reduseres, og et trent nevralt nettverk kan estimere utmattingsskader i sanntid.

Denne oppgaven presenterer en metode for hvordan et nevralt nettverk kan bygges, trenes og testes for å predikere akkumulert utmattingsskade, rimelig nøyaktig, i kritiske knutepunkt på en jacket-plattform lokalisert i Nordsjøen. Det nevrale nettverket ble laget ved bruk av programmeringsspråket Python, og mer spesifikt gjennom bibliotekene TensorFlow og Keras.

For å trene og teste det nevrale nettverket, må merket data konstrueres ved hjelp av moderne metoder for beregning av utmattingsskader. For dette formålet ble en elementmodell samt målinger av strukturelle forskyvninger og bølgehøyde levert av Aker BP. I tillegg ble målinger av vindstyrke og -retning hentet ut fra Meteorologisk institutt. Et tilstrekkelig antall dynamiske utmattingssimuleringer i tidsdomenet ble utført i USFOS, på en strippet versjon av den originale elementmodellen. For å påføre modellen stedsrelevante miljølaster ble det utført statistiske evalueringer av monitorert bølgehøyde og uthentet vindhastighet. Det merkede datasettet ble konstruert gjennom å kombinere miljølastene og statistiske evalueringer av de simulerte forskyvningene som etterlignet de monitorerte. For å optimalisere treningen av det nevrale nettverket ble det foretatt datarensing og -skalering.

Et fremovermatende nevralt nettverk ble laget, og ga gode indikasjoner på at et nevralt nettverk er i stand til ganske nøyaktige prediksjoner av utmattingsskader, basert på en elementmodell påført både irregulære bølger og vind fra forskjellige retninger. Robustheten ble bekreftet gjennom evaluering av prediksjoner gjort ved bruk av ulike typer aktiveringsfunksjoner og optimeringsmetoder, i tillegg til fjerning av enkelte inputverdier. For det anvendte nevrale nettverket var dataskalering og fjerning av avvik en forutsetning for å oppnå nøyaktige resultater. Det later derfor til at begrensningen når det gjelder eventuell bruk i industrien ligger i nøyaktigheten av de utførte elementmodellanalysene og utmattingsberegningene.

Table of Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CDF	Cumulative Distribution Function
DL	Deep Learning
DNV	Det Norske Veritas
FEA	Finite Element Analysis
FEM	Finite Element Model
FFNN	Feedforward Neural Network
FFT	Fast Fourier transform
FLS	Fatigue Limit State
GPS	Global Positioning System
MAE	Mean absolute error
MDOF	Multiple Degree of Freedom
ML	Machine Learning
MSE	Mean squared error
NN	Neural Network
NTNU	Norwegian University of Science and Technology
SCF	Stress Concentration Factor
SDOF	Single Degree of Freedom
SHM	Structural Health Monitoring
SN	Stress-cycle relationship
ULS	Ultimate Limit State

Fatigue symbols

$\bar{a}$	Intercept between the design SN-curve and $log(N)$ axis
β, α, γ	Joint geometry parameters
ΔS	Constant amplitude stress range
ϕ	Angular position around the circumference of the tube
σ	Local stress
σ_1 - σ_8	Hot-spot stresses along the circumference of a tubular beam
τ, ζ, θ	Joint geometry parameters
D	Fatigue damage
k	Thickness exponent
m	Negative inverse slope of the SN-curve
N_i	Number of cycles to failure at constant amplitude stress range i
n_i	Experienced number of cycles at interval i
R	R-ratio or stress ratio
S	Global stress
t	Thickness of tubular member
t_{ref}	Reference thickness of tubular member

Monitoring- and general symbols

α_1	Mass-proportional coefficient in Rayleigh damping
α_2	Stiffness-proportional coefficient in Rayleigh damping
$\bar{c}_i$	Modal damping for mode i in Rayleigh damping
$\bar{k}_i$	Modal stiffness for mode i in Rayleigh damping
$\bar{m}_i$	Modal mass for mode i in Rayleigh damping
β	Direction of regular wave components
$\mathbf{\Phi}$	Eigenvectors in Rayleigh damping
$\ddot{X}(\omega)$	Measured accelerations from accelerometers
$\ddot{X}_0(\omega)$	True acceleration signal from accelerometers
Δt	Time increment
γ	Peak enhancement factor in JONSWAP spectra
λ	damping ratio in Rayleigh damping
λ_d	Scale parameter in Weibull distribution

λ_w	Wavelength
$\ddot{\mathbf{x}}$	Acceleration vector for multi degree of freedom system
$\dot{\mathbf{x}}$	Velocity vector for multi degree of freedom system
$\mathbf{C}$	Damping matrix for multi degree of freedom system
$\mathbf{F}$	Force vector for multi degree of freedom system
$\mathbf{K}$	Stiffness matrix for multi degree of freedom system
$\mathbf{M}$	Mass matrix for multi degree of freedom system
$\mathbf{x}$	Displacement vector for multi degree of freedom system
ω	Angular frequency
ω_i	Eigenfrequency for mode i in Rayleigh damping
ϕ	Phase angle
σ	Standard deviation
σ^2	Variance
ε_x	Engineering strain
ζ	Wave surface elevation
ζ_a	Wave amplitude
a_n	Water particle acceleration normal to the pipe longitudinal axis
C_d	Drag coefficient
C_m	Inertia coefficient
d	Cylinder diameter
dF	Force component on a small element length in Morison's equation
ds	Element length
ds_0	Initial element length
E	Total energy in a sea state
E_x	Green strain
g	Gravity acceleration
h	water depth
H_s	Significant wave height
k	Wave number
k_d	Shape parameter in Weibull distribution
$m^{(n)}$	Spectral moments

$N(\omega)$	Uncorrelated noise from accelerometers
$R_{\zeta\zeta}$	Autocorrelation function for wave elevation
$S_{\zeta\zeta}(\omega)$	Wave elevation spectrum
T	Wave period
t	Time
T_p	Peak period
u_n	Water particle velocity normal to the pipe longitudinal axis
v_w	Wind velocity
ρ	Water density
ρ_w	Wind density

Machine learning symbols

γ	decay rate
$\hat{\mathbf{y}}i$	Vector containing values predicted by the neural network
$\mathbf{w}$	Vector containing neuron weights
$\mathbf{y}$	Vector containing true values
$\mathcal{L}$	Loss function
θ	Update parameter
ε	Global learning rate
g_t	gradient of a differentiable function at iteration step t
m_t	Moving average of the gradient in Adam
r_t	adjustment parameter in RMSProp
t	Time/iteration step
v_t	Moving average of the squared gradient in Adam
w_o	Bias connected to neurons
x	Neuron value
y	Output from a neuron passed through activation function
z	Scalar passed through activation function

Chapter 1

Introduction

1.1 Background and motivation

From an economical perspective it is often considered beneficial to extend the lifetime of already existing platforms. As a result, an increasing amount of offshore platforms are passing their intended design life. Knowing that fatigue failure is responsible for a substantial amount of repairs in the North Sea, it is a vital assessment to ensure structural integrity (Stacey and Sharp, 1997).

Structural integrity management needs to be performed regularly by the operator to ensure safety on offshore jacket platforms, which can be done by inspections of the exposed platform joints and/or monitoring. Lack of accessibility inside parts of the underwater jacket structure provides a need for inspections to be performed from the outside by divers or remotely operated vehicles, causing high inspection cost per joint (Moan, 2005). Continuous measurement of accelerations through one or several accelerometers may be used to discover structural defects by considering severe changes in response spectra, which implies that small defects as small cracks will not be captured.

Characteristic for offshore structures are the exposure to cyclic loads as a consequence of incoming wind, current and waves. Therefore, fatigue performance of the welded joints in a jacket structure is a design driving criterion (Dong et al., 2012). The current practice is to calculate the damage contribution by use of appropriate SN-curves and stress concentration factors (SCFs) for specific structural details. In order to do so, the hot-spot stresses needs to be found through finite element analysis (FEA). Currently, linear approaches are often applied, because nonlinear approaches are complex and computationally demanding (Shabakhty and Khansari, 2019). Nevertheless, traditional frequency domain analyses cannot properly handle nonlinear effects from multiple sources (Jia et al., 2008).

1.2 Literature review

This section aims to cover the state-of-the-art methods considered especially relevant to quantify the fatigue damage of a jacket structure subjected to dynamic loading, and subsequently estimate the fatigue damage by use of machine learning (ML). Firstly, methods on fatigue calculation will be presented. Secondly, some literature on how properties of monitoring equipment influences the acceleration measurements will be covered. Lastly,

research providing information on how ML can be used for fatigue assessment will be presented.

1.2.1 Finite element analyses for fatigue damage

State-of-the-art methods for fatigue calculation

Random loads are inherent in offshore structures due to wave loading. The resulting fatigue damage takes the form of a multi-step process. Hot-spot stresses at critical locations are derived though superimposing the contribution from axial-, in-plane- and out-of-plane actions (Dong et al., 2012). The hot-spot stress can be defined through FEA of the structure, and quantified in terms of a SCF before the time domain analyses takes place (Potvin et al., 1977).

Subsequently, time domain analyses needs to be performed to quantify the stress ranges in various representative environmental conditions in the long term period. The stress ranges are sorted into bins, normally by use of Rainflow counting (Chaudhury and Dover, 1985), and the fatigue damage is determined through application of appropriate SN-curves and Miner summation (Moan, 2005).

Quantification of stress concentration factors

The ability to quantify the SCFs is a prerequisite to assess fatigue damage in a sufficient manner. For some tubular joints, the maximum stress at the hot-spot can be 20 times as high as the nominal stress (Potvin et al., 1977). Modern methods of determining SCF's utilizes FEA with very fine mesh consisting of shell elements. N'Diaye et al. (2007) investigated how the SCF varied in the vicinity of the weld fillet in a tubular T-joint subjected to both static and cyclic loading, and obtained results which compared well with similar studies. They concluded that the SCF was higher on the brace- than the chord member, with a maximum obtained SCF on the brace equal to 9.62.

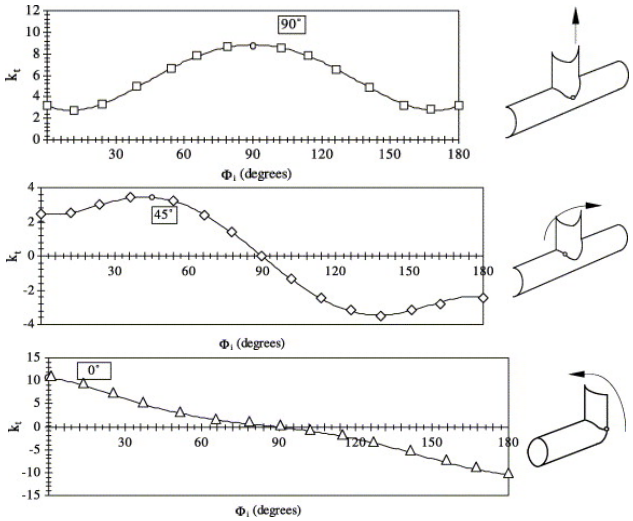


Figure 1.1: Variations of SCF across the brace fillet weld with respect to angular position, Φ, originated at the crown toe point, for axial-, in-plane- and out-of-plane loading (N'Diaye et al., 2007).

For multi-planar tubular DKT-joints, Ahmadi and Lotfollahi-Yaghin (2013) found that the SCF generally was significantly higher in the middle brace than in the outer braces.

Dynamic time domain finite element analyses

Mendes et al. (2021) performed both static- and dynamic analyses on a jacket structure, and concluded that the dynamic analysis provided slightly higher fatigue damage to the structure in general, but not all parts. For the dynamic analyses, several investigations of both linear and non-linear dynamic behavior of jacket-type platforms was performed. Among them, several shown that the motions for jacket structures typically are small, and that Morison's formula sufficiently predicts the quasi-static loads subjected to the jacket (Shabakhty and Khansari, 2019).

Jia et al. (2008) evaluated fatigue of a jacket through nonlinear time domain analyses for cases where the pile-soil information were both present and unavailable. They found that among the selected positions, the fatigue damage decreased with increasing water depth, and that the most critical joints were located roughly around 11 m below the still water level. Therefore, for a tall jacket platform with a natural flexural period of above 3.5 s, fully fixed boundary conditions proved sufficiently accurate for fatigue damage assessment on the upper part. The study also found that inertia effects of the structure, topside installations and equipment, influenced the fatigue damage significantly. Their analysis procedure is summarized below:

1. Establish the finite element model (FEM).
2. Divide scatter diagram into representative sea states.

   22 blocks with relevant probability.

   8 wave directions with relevant probability.
3. Set up damping model for the structure.
4. Include hydrodynamic coefficients, C_d and C_m, include buoyancy effects and define the splash zone.
5. Dynamic analyses.

   Start with the first sea state and wave direction, and proceed through the 176 analyses.

   Document the force-time history for selected elements.
6. Calculate the SCFs.
7. Pick a suitable SN-curve.
8. Calculate the fatigue damage for each sea state from the force-time histories.

   Calculate hot-spot stresses.

   Calculate stress ranges and number of cycles using Rainflow counting.
9. Calculate the one year fatigue damage for each hot-spot of a joint.
10. The respective damages for the 176 sea states are multiplied with the probability for each sea state and accumulated to the one year fatigue damage. The maximum damage among those hot-spots are selected as the one year fatigue damage.
11. subtract the prior installation and service damage.

1.2.2 Fatigue estimations through machine learning

As computational power has increased dramatically over the past couple of years, ML has become more viable to use. Therefore, it is appropriate to explore the influence of ML for the offshore industry. Yan et al. (2019) used ML to perform a probabilistic fatigue analysis of a bridge subjected to truck loading. Initially, the deterministic responses of the bridge during overloading was found by use of FEA in ANSYS. Subsequently, a feedforward neural network (FFNN) was developed, trained and tested to be used in combination with Monte Carlo simulations to predict the fatigue failure probability. Monte Carlo simulations was used to create additional data, with a limited amount of FEA. The achieved ML results corresponded very well with the computed FEA results, with a maximum absolute error of about 0.35%.

Luna et al. (2020) trained and tested different artificial neural networks (ANNs) to reduce the tower fatigue damage on an onshore wind turbine subjected to wind load. The input for the ANNs was received data from the tower top oscillation velocity and the feedback loop providing the expected instantaneous damage. The resulting fatigue damage was the output. The hyperbolic tangent function was as activation function for the hidden layers, and a 80%-20% split between test data and validation data was used. While determination of the optimal architecture of the neural network (NN) was beyond the scope of the paper, computational power was a limitation, and it was of importance to minimize the complexity of the ANN while still maintaining good accuracy. They found that nonlinear autoregressive networks with exogenous input was the best option to predict the tower fatigue, and obtained accurate results for estimated fatigue damage.

To achieve accurate results, they found it important to filter for outliers as they may originate from noise, and contribute to false trends. Although the ML-model showed promising results with wind speed and damage correction through feedback as the only input, they concluded that additional input features could help improving the results.

Figueiredo et al. (2011) used ML-algorithms to detect structural damage undergoing operational- and environmental variability. As data from most structures only are available from the undamaged condition, they sought to utilize unsupervised learning allowing acceleration time-series to be the only input. One of the algorithms was a combination of supervised learning and unsupervised learning, so called semi-supervised learning. Supervised learning should obtain the environmental- and operational conditions, while unsupervised learning should detect damage. The accuracy of the model when applied new data was uncertain, and the behaviour of the model if any structural damage should change the dynamic response was also unclear.

1.2.3 Structural health monitoring

Monitoring instruments has in recent years become increasingly accurate and inexpensive, relatively speaking, and thus more viable to use for assessment of structural integrity. Structural monitoring systems of offshore structures usually consists of a set of sensors, such as GPS's, wave radars and accelerometers. Displacements of large structures can be determined by using a limited number of sensors, typically located on the topside to ease the installation, measuring velocities or accelerations. According to Hansen et al. (2011), well-designed sensors are capable of providing accurate information for frequencies above 0.1 Hz, which is quite accurate, but also prevents capture of the complete response. Even though monitoring of offshore structures has proven to be increasingly accurate, a perfect

method is yet to be established (Tang et al., 2015). In relation to offshore structures, monitoring can be used to assess typical finite element input parameters influencing the modal parameters as mass, center of gravity (CoG), damping and various stiffnesses: element, joint and soil.

Noise

The quality of the signal will inevitably reduce as inherent noise and processing noise are introduced. In general, any signal from an integrated sensor signal can be expressed as the sum between the true acceleration signal, $\ddot{X}_0$, and the polluted uncorrelated noise, N (Hansen et al., 2011). In the frequency domain, where ω is the angular frequency, the equation for an acceleration signal becomes:

$$\ddot{X}(\omega) = \ddot{X}_0(\omega) + N(\omega) \tag{1.1}$$

In many situations it is desirable to integrate the acceleration to quantify the velocity and displacements. The consequence of performing integration on Equation 1.1 is that the noise gets amplified as a function of ω, where frequencies close to direct current are particularly exposed. Hence, filtering algorithms are applied to minimize the influence of the noise (Hansen et al., 2011).

Axioms of structural health monitoring

In relation to the development of structural health monitoring (SHM), some fundamental axioms for SHM has been established. The axioms considered relevant for this thesis are summarized below (Worden et al., 2007):

- Axiom II. Damage assessment requires a comparison between two system states.
- Axiom III: Identification of damage existence and -location can be done by an unsupervised ML-model, while the severity can generally only be done by supervised models.
- Axiom IVa: Sensors are incapable of measuring damage.
- Axiom V: The required properties of the SHM are dictated by the time-scales associated with damage initiation and evolution.
- Axiom VI: There is a trade-off between an algorithms damage sensitivity and noise rejection capability.
- Axiom VII: The magnitude of the detectable damage due to changes in the system is inversely proportional to frequency range of excitation.

1.3 Objectives

This Master's thesis aims to investigate whether it is possible to build and train a supervised ML-model to calculate the accumulated fatigue damage of a jacket structure on real-time monitored data within acceptable accuracy. Today's standard approach requires time-consuming FEA in the time domain to properly assess the accumulated fatigue damage. Although a supervised ML-model consisting of a NN needs tailored FEA to generate a dataset used for training and testing, only a limited amount has to be performed, consequently decreasing the long-term time consumption.

In order to obtain accurate predictions, the NN needs to be trained on a sufficient dataset. Hence, the thesis further aims to develop a possible tabulation method by use of a calibrated FEM and on-site measurements of structural accelerations, waves and wind. A prerequisite for the information contained in the dataset is that it should be accessible directly from on-site measurements, and that no information available only through simulations should be included. The NN should take in statistical properties of the measured waves, -displacements and wind, as well as joint- and connection indexations, and return the one-year fatigue damage given the experienced sea state.

1.4 Thesis outline

- **Chapter 2** introduces relevant theory within dynamic time domain simulations in USFOS, fatigue of tubular joints and ML.
- **Chapter 3** describes the applied method to create the dataset used for training and testing, and the development of the ML-model. In addition, intermediate results supported by directly applied theory and immediate observations will be presented and discussed.
- **Chapter 4** presents the results obtained from training and testing the NN, including discussion of the obtained results and applied method.
- **Chapter 5** provides a conclusion based on the results and discussion, and lastly recommends further work.

Chapter 2

Theory

In order to create and evaluate the results properly, it is important to be aware of the calculation techniques utilized in this thesis. Initially, this section aims to provide the necessary theoretical background on how time domain simulations are performed in USFOS. Subsequently, the theory describing how fatigue calculations are performed in the USFOS-module Fatal is presented. Lastly, relevant theory explaining fundamental ML concepts will be covered.

2.1 Basis for USFOS fatigue analysis

2.1.1 Dynamic equation of motion

The motion of a multiple degree of freedom (MDOF) system of size N subjected to external loading can be described by the dynamic equation of motion in Equation 2.1.

$$\boldsymbol{M}\ddot{\boldsymbol{x}}(t) + \boldsymbol{C}\dot{\boldsymbol{x}}(t) + \boldsymbol{K}\boldsymbol{x}(t) = \boldsymbol{F}(t) \tag{2.1}$$

Here, $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}$ is the damping matrix, $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{F}$ is the load vector and $\boldsymbol{x}$ is the displacement vector. Equation 2.1 can be solved both in the time domain and in the frequency domain. However, solutions in the frequency domain requires linearity, and cannot properly handle nonlinear effects. In the time domain, the solution procedure can be done either by numerical integration methods or difference formulations of N uncoupled equations.

2.1.2 Finite element modelling

Jacket structures are usually modelled by beam elements (Amdahl, 2009). As illustrated in Figure 2.1, the general three dimensional beam element in USFOS consists of one node at each end, where each node contains three translational- and three rotational degrees of freedom.

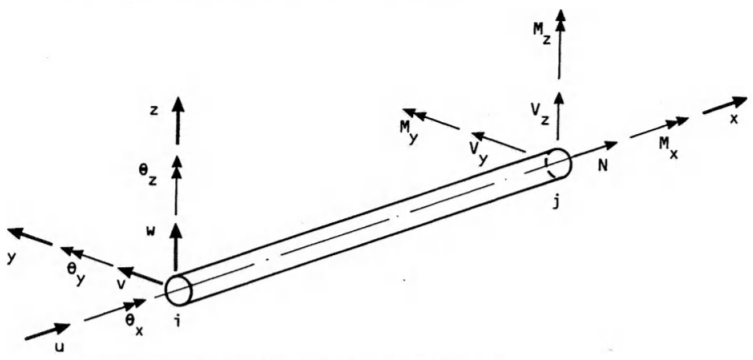


Figure 2.1: General three dimensional beam element in USFOS (Søreide et al., 1994).

An advantage in USFOS which reduces the computational effort is that large, complex structures can be modelled by relatively few elements, since the structural members often are modelled by a single element. However, a consequence is that geometrical nonlinearities on element level are needed in order to account for local collapse modes.

The USFOS analysis model is called the Idealized Structural Unit method, and is based upon the updated Lagrangian formulation on a global level. This implies that the element reference axes are updated, for each time step, throughout deformation. In other words, the discrete equations are based on the current deformation. As a result of the sparse number of beam elements, large deflections are incorporated by second-order strain terms on element level, meaning that a total Lagrangian formulation is implemented locally (Søreide et al., 1994).

The total Lagrangian formulation is characterized by utilizing Green strain, implying that the corresponding stress component is the 2^{nd} Piola-Kirchhoff stress (Moan, 2003). The Green strain, E_x, is given in Equation 2.2, where ds and ds_0 are infinitesimal line segments in current and initial configuration, respectively.

$$E_x = \frac{ds^2 - ds_0^2}{2ds_0^2} \tag{2.2}$$

Utilizing the engineering strain given in Equation 2.3, the Green strain can be rewritten according to Equation 2.4.

$$\varepsilon_x = \frac{ds - ds_0}{ds_0} \tag{2.3}$$

$$E_x = \varepsilon_x + \frac{1}{2}\varepsilon_x^2 \tag{2.4}$$

2.1.3 Time domain method

Dynamic time domain analyses can be performed by defining a time period and a time increment, Δt. The time increment is usually set to be constant during the entire time period, such that $\Delta t = c$. Numerical stability is ensured as long as the time increment is less than a prescribed fraction of the eigenperiod of the structure. The solution procedure for each time step takes place through assuming a certain behaviour of the motion, where

the solution accuracy is inversely proportional to the length of the time increment. The consequence is that the time domain method is computationally demanding, but versatile in the sense that it is able to capture nonlinear effects.

Numerical Integration

Consider the time dependent equation of motion for a single degree of freedom (SDOF) system, given in Equation 2.5

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t), \tag{2.5}$$

where the force, $F(t)$, is a continuous function with angular frequency ω and phase ϕ:

$$F(t) = F_0 sin(\omega t + \phi) = \sum_{j=1}^{n} F_{0,j} sin(\omega_j t + \phi_j) \tag{2.6}$$

The principle is that dynamic equilibrium is found between discrete points between the current time step, i, and the next time step, $(i+1)$. Dynamic equilibrium states that the acceleration, $\ddot{x}_{i+1}$, needs to satisfy Equation 2.7.

$$\ddot{x}_{i+1} = \frac{1}{m}(F_{i+1} - c\dot{x}_{i+1} - kx_{i+1}), \tag{2.7}$$

where x_i, $\dot{x}_i$, $\ddot{x}_i$, F_i and F_{i+1} are known. Common for all methods based on numerical integration is that velocity and displacement are found by integrating the acceleration twice:

$$\dot{x}_{i+1} = \dot{x}_i + \int_0^c \ddot{x}(t)dt \tag{2.8}$$

$$x_{i+1} = x_i + \int_0^c \dot{x}(t)dt \tag{2.9}$$

Subsequently, an assumption needs to be made regarding how the acceleration will vary over the interval, and thereby allowing computation of Equation 2.8 and Equation 2.9. The numerical time integration in USFOS is based on the HHT-α method, and the reader is referred to Chapter 14.4 in the USFOS theory manual (Søreide et al., 1994). Solving the MDOF system of size N in Equation 2.1 is equivalent to solving the N uncoupled equations by the method described above (Langen, 1999).

2.1.4 Irregular waves

For time domain fatigue simulations, irregular waves tends to give the best representation of reality (USFOS, 2010). Irregular sea is generated by superimposing a finite set of discrete regular waves. This is illustrated in Figure 2.2.

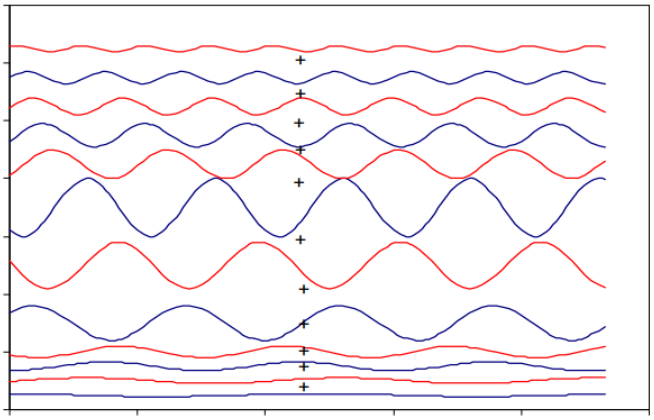


Figure 2.2: Illustration of irregular waves composed by superposition of regular waves (USFOS, 2010).

The superimposing of regular waves is done through Fast Fourier Transform (FFT) of the wave energy spectrum. Mathematically, this becomes as in Equation 2.10.

$$\zeta(x,y,t) = \sum_{j=1}^{N} \zeta_{a,j} cos\left(\omega_j t - k_j cos(\beta x) - k_j sin(\beta y) - \phi_j\right) \tag{2.10}$$

where:

- ζ is the surface elevation at coordinate (x, y) and time t.
- $\zeta_{a,j}$ is the amplitude of harmonic wave component j.
- k_j is the wave number for wave component j.
- ϕ_j is the random phase angle, uniformly distributed between 0 and 2π, for wave component j.
- β is the direction of the wave components.

2.1.5 Spectral density

The total energy, E, in a sea state, accumulated by adding N harmonic wave components, is given by Equation 2.11, where ρ is the water density and g is the gravity acceleration.

$$\frac{E}{\rho g} = \sum_{n=1}^{N} \frac{1}{2} \zeta_{a,n}^2(\omega_n) \tag{2.11}$$

By introducing a wave spectrum, $S_{\zeta\zeta}(\omega)$, the area inside a small frequency interval, $\Delta\omega$, is equal to the energy of all the wave components within this frequency interval (Myrhaug, 2019):

$$\frac{1}{2}\zeta_{a,n}^2 = S_{\zeta\zeta}(\omega_n)\Delta\omega \tag{2.12}$$

By inserting Equation 2.12 in Equation 2.11, and let $N \to \infty$ such that $\Delta\omega \to 0$, the total energy can be rewritten according to Equation 2.13.

$$\frac{E}{\rho g} = \sum_{n=1}^{N} S_{\zeta\zeta}(\omega_n)\Delta\omega = \int_0^{\infty} S_{\zeta\zeta}(\omega_n)\Delta\omega \tag{2.13}$$

2.1.6 Morison force calculation

Wave loads are calculated by use of Morison's equation, which provides the wave force on a slender fixed cylindrical element. The equation is composed of a linear inertia-term and a second order nonlinear drag-term. For a small element with length ds, the corresponding force, dF, becomes as in Equation 2.14:

$$dF = \left\{ \rho \frac{\pi}{4} d^2 C_m a_n + \frac{1}{2} \rho C_d d u_n |u_n| \right\} ds \tag{2.14}$$

where ρ is the water density, d is the diameter of the cylinder, C_m is the inertia coefficient included added mass, a_n is the water particle acceleration normal to the pipe longitudinal axis, C_d is the drag coefficient and u_n is the water particle velocity normal to the pipe longitudinal axis.

It is important to emphasize that Equation 2.14 is only sufficiently accurate for slender members, defined as $\frac{\lambda_w}{d} > 5$, where λ_w is the wavelength. This criterion is generally fulfilled by jacket structures, including the one used in this thesis. According to linear theory, deep water is defined as $\frac{\lambda_w}{h} < \frac{1}{2}$, where h is the water depth. With a maximum wave period allowed for the simulations, $T_{max} = 18$ s, deep water cannot reasonably be assumed for all wave periods. Therefore, the definitions of the wave number, k, and the dispersion relation becomes as shown in Equation 2.15 and Equation 2.16, respectively.

$$k = \frac{2\pi}{\lambda_w} \tag{2.15}$$

$$\omega^2 = \left(\frac{2\pi}{T}\right)^2 = kg \cdot tanh(kh) \tag{2.16}$$

Here, ω is the angular frequency and g is the gravity acceleration. The solution for λ_w in Equation 2.16 needs to be found iteratively. By considering the wave period used as the lower limit for the simulations, $T_{min} = 4$ s, gives $\lambda_w = 24.8$ m. Knowing that the largest diameter used in the jacket model is 3.2 m, the minimum wave length to diameter ratio becomes as shown in Equation 2.17.

$$\frac{\lambda_w}{d} = \frac{24.8}{3.2} = 7.8 > 5 \tag{2.17}$$

For non-slender members however, diffraction forces becomes important (Pettersen, 2007). The total force acting on a structural member is found by integrating Equation 2.14 over the member length along the longitudinal axis. The total hydrodynamic loads acting on the structure are integrated from the sea bottom to the instantaneous wetted surface (USFOS, 2010).

2.2 Fatigue of welded tubular joints

Fatigue damage is caused by cyclic loads over time, usually causing stresses smaller than the yield strength of the material, and is therefore a cycle by cycle process of damage accumulation. Fatigue damage becomes important for offshore structures, as the cyclic behaviour of waves contributes to a large degree of dynamic loading. In fatigue assessments of tubular joints, only the dynamic loading from the braces is considered, and the stresses in the chord are neglected. This is because the chord in most cases is subjected to static loading, whereas environmental dynamic forces governs the brace loading (Berge and Ås, 2017). Cyclic stress loading acting on a structural component, following a sinusoidal pattern, can be described by the parameters shown in Figure 2.3.

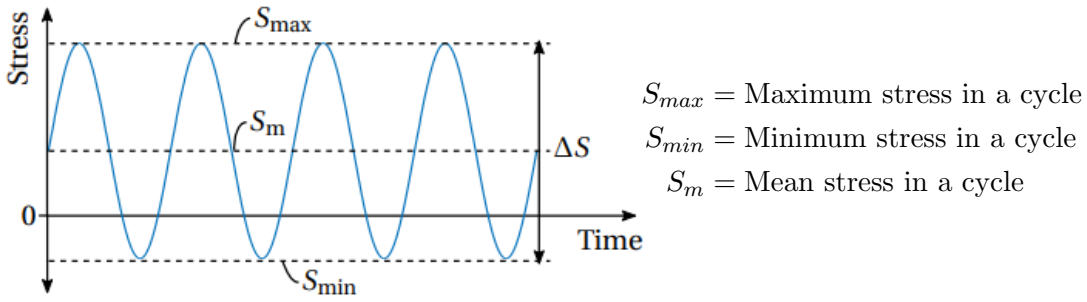


Figure 2.3: Cyclic load history with symbols (Berge and Ås, 2017).

From Figure 2.3 it can also be seen that the stress range is defined according to Equation 2.18, while the stress ratio, commonly called the R-ratio, is defined as in Equation 2.19.

$$\Delta S = S_{max} - S_{min} \tag{2.18}$$

$$R = \frac{S_{min}}{S_{max}} \tag{2.19}$$

Utilizing Equation 2.18 and Equation 2.19, the mean stress can be expressed as in Equation 2.20.

$$S_m = \frac{\Delta S}{2}\left(\frac{R+1}{R-1}\right) \tag{2.20}$$

The stress range is often the primary contributor to fatigue damage, while the mean stress is considered secondary. Increased tensile mean stress, composed of both residual stresses and stresses induced by external load, lowers the fatigue strength of the structure. According to Berge and Ås (2017), residual stresses in welded structures are found to be large. For welds transverse to the loading axis, in which the whole cross-section is heated simultaneously during the welding process, specimen used for SN-testing struggles to accurately represent residual stresses of full scale structures. A relaxation of the residual stresses will happen for large stress ranges, as the hot-spot stress cannot exceed the yield stress of the material. Residual stresses will therefore be more important for high-cycle fatigue (Zhang and Moan, 2006).

For offshore structures, the fatigue stresses tends to be in the high-cycle range, from around 10^5 to 10^7 cycles, where the stresses are elastic. The low-cycle range is less important, and generally not given in design standards. In the high-cycle range strain-cycle data tends to follow a log-linear relationship, seen in Equation 2.21, commonly denoted the SN-curve (Berge and Ås, 2017).

$$N(\Delta S)^m = \text{Constant} \tag{2.21}$$

Here, N denotes the number of cycles to failure and ΔS is the constant amplitude stress range. Plotted on a log-log format, Equation 2.21 becomes a straight line where m is the negative inverse slope. A fatigue limit may occasionally be defined, implying that stress ranges below this threshold does not contribute to fatigue damage.

2.2.1 SN-curves for welded tubular joints

SN-curves are found from analysis on test data, and given as two standard deviations below the mean regression line to ensure conservatism. The standard equation for an SN-curve for tubular joints in seawater with cathodic protection, plotted on a log-log format, is given according to Equation 2.22 where $\bar{a}$ is the intercept between the design SN-curve and $log(N)$ axis.

$$logN = log\bar{a} - m\left[\Delta S\left(\frac{t}{t_{ref}}\right)^k\right] \tag{2.22}$$

The difference relative to Equation 2.21 is the thickness correction term, where t is the thickness of the tube to be studied, t_{ref} is the reference thickness proposed by DNV as 16 mm for tubular joints, and k is the thickness exponent[1] determined from the weld classification (DNV, 2014). Note that the correction term in Equation 2.22 entails that a solitary thickness increase of the fatigued specimen causes a decrease in fatigue strength.

The SN-curve depends heavily on the weld details. Smooth surfaces without sharp notches and non-welded details will generally provide better structural integrity in terms of fatigue. DNV RP-C203 proposes different classes for the welds based on considerations of the weld itself, geometrical properties of the local structure and local direction of the load (DNV, 2014).

[1] Only applied as a reduction factor. Not used for $t > t_{ref}$ (Berge and Ås, 2017).

2.2.2 Stress concentration factor

In order to investigate how stress is distributed through critical areas of the jacket structure, it is crucial to understand what a SCF is. The SCF is defined as the scaling factor between the nominal stress, σ_{nom}, and the maximum local stress, σ_{max}, which implies that the stress experienced at a given location differs from what it would have been if the structure was uniform. This can be seen in Equation 2.23:

$$SCF = \frac{\sigma_{max}}{\sigma_{nom}} \tag{2.23}$$

The SCF can be divided into a global and a local SCF, whereas for a jacket structure, the global SCF is of most interest. In other words, the peak stress from the weld geometry is of minor significance (Berge and Ås, 2017).

The *global SCF* is due to sudden changes in geometry, and varies with load type and -direction, the joint types and geometrical properties of the joints. The *local SCF* follows directly from the weld classification, and is included in the tabulated SN-curve seen in Table 3.9. Subsequently, the total SCF is given as the product between the global- and local SCF:

$$SCF_{total} = SCF_{global} \cdot SCF_{local} \tag{2.24}$$

Although state-of-the-art methods utilizes FEA to determine SCFs, it can be very time consuming. DNV provides equations for determination of SCFs for tubular joints, based on the parameters seen in Figure 2.4. The geometrical parameters used to quantify the SCFs are defined in Equation 2.25.

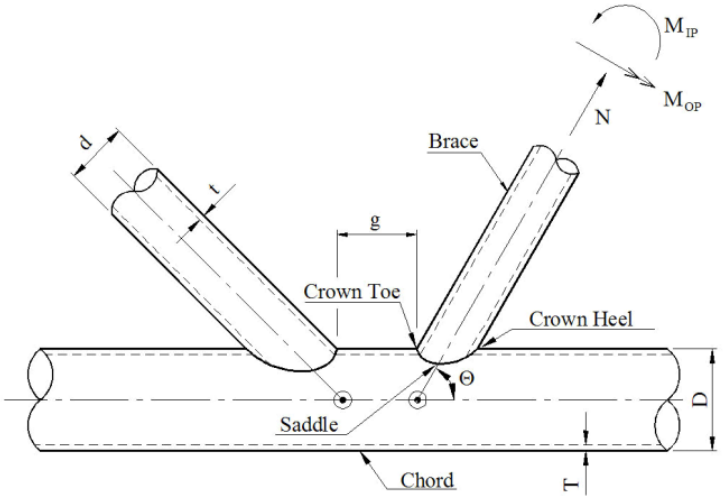


Figure 2.4: Geometrical definitions for a common tubular joint (DNV, 2014).

$$\beta = \frac{d}{D}, \quad \alpha = \frac{2L}{D}, \quad \gamma = \frac{D}{2T}, \quad \tau = \frac{t}{T}, \quad \zeta = \frac{g}{D} \tag{2.25}$$

Here, L is the length of the chord section considered a part of the joint and θ is the angle between the brace and chord. Although the equations in Equation 2.25 are a useful tool,

they are only valid in the ranges given in Equation 2.26.

$$\begin{array}{lcccl} 0.2 & \leq & \beta & \leq & 1.0 \\ 0.2 & \leq & \tau & \leq & 1.0 \\ 8 & \leq & \gamma & \leq & 32 \\ 4 & \leq & \alpha & \leq & 40 \\ 20° & \leq & \theta & \leq & 90° \\ \frac{-0.6\beta}{sin\theta} & \leq & \zeta & \leq & 1.0 \end{array} \tag{2.26}$$

For each brace connected to the chord in a joint, the SCF should be evaluated at both the crown and the saddle. The maximum obtained SCF should then be set as the SCF around the whole brace-chord intersection and used for further stress calculations.

2.2.3 Hot-spot stress in tubular joints

It is the hot-spot stresses that are used in conjunction with the appropriate SN-curves to calculate the fatigue damage. Similarly to the calculations of SCFs, hot-spot stresses are usually determined through FEA. However, DNV-RP-C203 proposes an alternative procedure, where the stresses are calculated at the crown and saddle points by parametric equations. The corresponding hot-spot stress is found by direct summation of the single stress components from axial-, in-plane- and out-of-plane action. Subsequently, the hot-spot stress should be evaluated at eight points around the circumference of the intersection, to take into account that the hot-spot stresses may be larger at intermediate points. The hot-spot stresses at the individual locations seen in Figure 2.5 are then derived by linear interpolation of the contributions from the axial stress at the saddle and chord and a sinusoidal variation of the bending stresses, given by the formulas in Equation 2.27 (DNV, 2014).

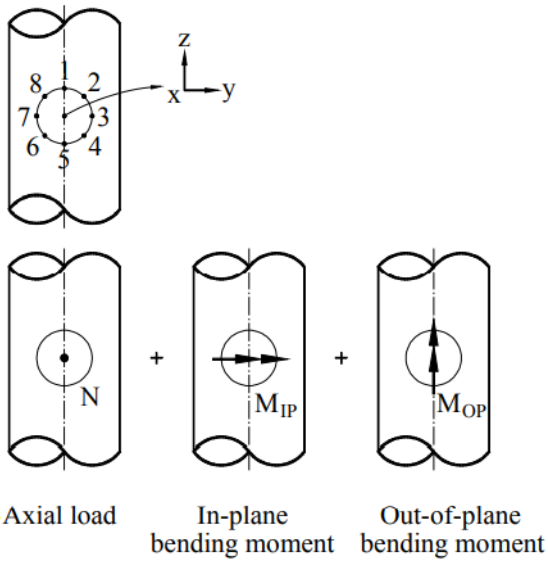


Figure 2.5: The hot-spots around the periphery of the weld connection where the stresses are superimposed (DNV, 2014).

$$
\begin{aligned}
\sigma_1 &= SCF_{AC}\sigma_x + SCF_{MIP}\sigma_{my} \\
\sigma_2 &= \frac{1}{2}(SCF_{AC} + SCF_{AS})\sigma_x + \frac{1}{2}\sqrt{2}SCF_{MIP}\sigma_{my} - \frac{1}{2}\sqrt{2}SCF_{MOP}\sigma_{mz} \\
\sigma_3 &= SCF_{AS}\sigma_x - SCF_{MOP}\sigma_{mz} \\
\sigma_4 &= \frac{1}{2}(SCF_{AC} + SCF_{AS})\sigma_x - \frac{1}{2}\sqrt{2}SCF_{MIP}\sigma_{my} - \frac{1}{2}\sqrt{2}SCF_{MOP}\sigma_{mz} \\
\sigma_5 &= SCF_{AC}\sigma_x - SCF_{MIP}\sigma_{my} \\
\sigma_6 &= \frac{1}{2}(SCF_{AC} + SCF_{AS})\sigma_x - \frac{1}{2}\sqrt{2}SCF_{MIP}\sigma_{my} + \frac{1}{2}\sqrt{2}SCF_{MOP}\sigma_{mz} \\
\sigma_7 &= SCF_{AS}\sigma_x + SCF_{MOP}\sigma_{mz} \\
\sigma_8 &= \frac{1}{2}(SCF_{AC} + SCF_{AS})\sigma_x + \frac{1}{2}\sqrt{2}SCF_{MIP}\sigma_{my} + \frac{1}{2}\sqrt{2}SCF_{MOP}\sigma_{mz}
\end{aligned}
\tag{2.27}
$$

Here, σ_x, σ_{my} and σ_{mz} are the maximum nominal stresses due to axial load, in-plane bending and out-of-plane bending, respectively. The SCFs are given according to Subsection 2.2.2, where SCF_{AS} and SCF_{AC} are the SCFs at the saddle for axial load in the saddle and crown, respectively. SCF_{MIP} and SCF_{MOP} are the SCFs due to in-plane and out-of-plane moment. The hot-spot stress is calculated for all of the eight points, where the highest computed stress is considered governing for fatigue failure, and should conservatively be set as representative for the whole brace (Shabakhty and Khansari, 2019).

2.2.4 Miner-Palmgren summation

Since fatigue loads are based on constant amplitude, while the stresses varies randomly, a method treating an irregular series as a sum of constant amplitude loads is needed (Mendes et al., 2021). The Miner's rule is often used to calculate the damage contribution from a given stress range. It is very simple, but has proven to be no worse than other, more complex methods. The damage contribution from an arbitrary single stress cycle, i, is given from the Miner sum as:

$$D_i = \frac{1}{N_i} = \frac{1}{\bar{a}}(\Delta\sigma_i)^m \tag{2.28}$$

When utilizing the assumption of linear cumulative damage, the equations becomes:

$$D_i = \sum_{i=1}^{j} \frac{n_i}{N_i} = \frac{1}{\bar{a}} \sum_{i=1}^{j} n_i(\Delta\sigma_i)^m \tag{2.29}$$

$\Delta\sigma_i$ is the constant stress range for cycle i where n_i is the corresponding number of cycles. N_i is the number of cycles to failure at $\Delta\sigma_i$, j is the total number of cycles, m is the negative inverse slope of the SN-curve and $\bar{a}$ is the intercept between the design SN-curve and $logN$ axis.

2.2.5 Rainflow counting method

The rainflow counting method is a cycle-counting method which transforms physical measurements, which are governed by irregularities, to a suitable form for fatigue analysis (Amzallag et al., 1994). Practically this means that it allows Miner's rule to be applied. Several other cycle-counting methods also exists, but rainflow counting generally seems to be the best suited for wide-band loading. For cracked specimens the physical significance is lost, as crack closure may happen when subjected to compressive loading (Berge and Ås, 2017).

The rainflow algorithm is based on the principle of stress-strain response of the material. It defines a cycle as a closed hysteresis loop by combining load reversals without affecting the remaining history. Each closed hysteresis loop has an associated stress range and a mean stress that can be compared with the constant amplitude. Half-cycles that lack an equivalent counterpart cannot be used for damage assessment. As can be seen in Figure 2.6, peaks and valleys which are not local minima or maxima are swiftly neglected through elastic deformation to prior load history and the cycle continues along the prior path (Musallam and Johnson, 2012).

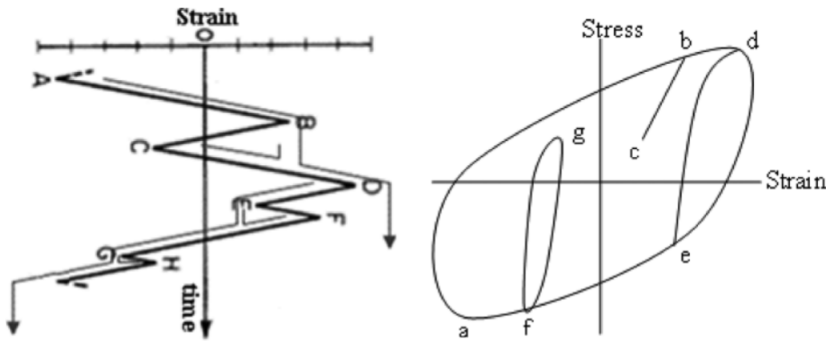


Figure 2.6: Closed hysteresis loop formed by rainflow counting of stress-strain cycles (Musallam and Johnson, 2012).

2.3 Fundamental machine learning theory

ML is a subcategory of artificial intelligence (AI). Goodfellow et al. (2016) describes ML as the machines ability to acquire knowledge by creating patterns from raw data. This differs from conventional programming where a set of rules is defined manually prior to execution. Deep learning (DL) is a subclass of ML which, inspired by the human brain, utilizes a multi-layered ANN architecture (Guo et al., 2016).

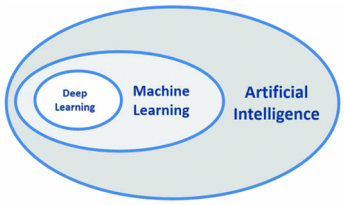


Figure 2.7: Visual representation of class hierarchy: AI, ML and DL (Holzinger et al., 2018).

2.3.1 Supervised- and unsupervised learning

Supervised learning is the most applicable in many situations, as it often is easy to determine the mathematical accuracy of the model because the true output is known in advance. During training, the model is given input and produces an output, which is compared with the true output. An objective function measures the deviation, and adjusts the weights to reduce the error. Mathematically, the reduction takes place as the learning algorithm computes a gradient vector that, for each weight, investigates if the error increases or decreases when the weight is modified (LeCun et al., 2015).

In unsupervised learning the model trains by its own, on unlabeled data, to discover information and trends that may not easily be detected by humans. Unsupervised learning is mostly applied on classification problems, where it connects the data attributes, as acceleration measurements from sensors, to class attributes (Liu, 2011). For structural dynamic behaviour, supervised learning can be used to determine which of the environmental load component that is the dominant factor for the observed load.

2.3.2 Deep feedforward network

A FFNN is characterised by that information only flows in one direction - it propagates forward. A deep FFNN is essentially just some extended version, where multiple layers are connected. Figure 2.8 shows a fully connected FFNN consisting of four layers, implying that all the neurons in one layer are connected to every neuron in the previous layer (Vieira et al., 2017).

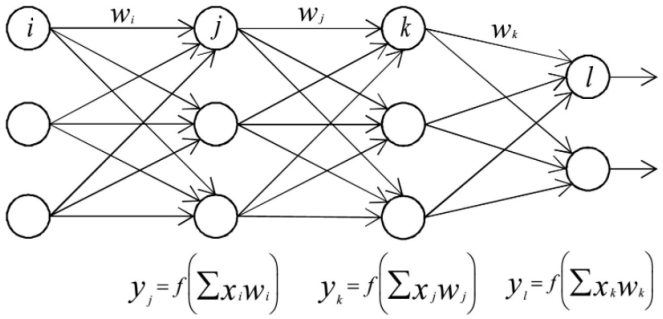


Figure 2.8: Fully connected FFNN (Vieira et al., 2017).

The first layer is the input layer and the last layer is the output layer. The in-between layers are so called hidden layers. As can be seen in Figure 2.8, the output depends on the input.

2.3.3 Neurons

Neurons are the building blocks of the ANN, and can be seen in Figure 2.9.

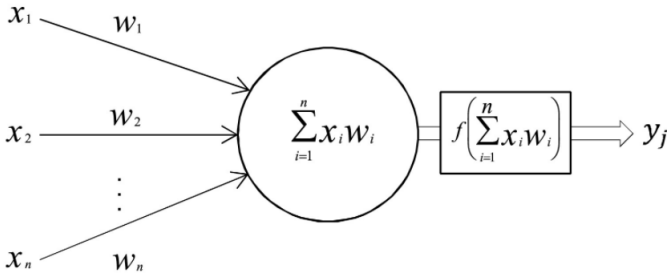


Figure 2.9: Connection of a single neuron in a NN (Vieira et al., 2017).

The neurons gets inputs $x_1, x_2, ..., x_n$ with corresponding weights $w_1, w_2, ..., w_n$, quantifying the relative importance of the of each contribution to the neuron output. In addition, the neuron has a bias, w_0, connected to it, introducing a slight shift to the output. The sum of all weighted inputs and bias then produces a scalar, z:

$$z = \sum_{i=1}^{n} x_i w_i + w_0 = \boldsymbol{w}^T \boldsymbol{x} + w_0 \tag{2.30}$$

Subsequently, the scalar, z, is passed through an activation function chosen by the user, f, where the output, y_j, acts as an input for the next layer. Mathematically, this becomes:

$$y_j = f(z) \tag{2.31}$$

2.3.4 Activation functions

As shown in Equation 2.31, activation functions are used to transform the signal from one neuron to be input for the next layer. Activation functions are needed in order to

assess nonlinear relationships among the data. If activation functions are omitted, the NN acts as a linear regression model (Sharma et al., 2017). An important trait for activation functions is that they need to be differentiable for back propagation methods to be applied for optimization of weights used to minimize errors. Some of the most popular activation functions are the rectifier function, where neurons utilizing it are named rectified linear unit (ReLU), the softmax function, the hyperbolic tangent (tanh) function and the sigmoid function (Vieira et al., 2017).

ReLU

ReLU is one of the simplest activation functions, and maps the input values to the range $[0, \infty]$. This means that any negative values maps to zero, while positive values remain unchanged. Mathematically, the function can be seen in Equation 2.32.

$$f(z) = \begin{cases} z & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases} \tag{2.32}$$

Models that use ReLU are often fast and accurate. However, some of the ReLU neurons may be put into states where they remain inactive, i.e. returns zero, for virtually all inputs. This leads to zero-gradients, and the neuron cannot be activated again.

Tanh

The tanh activation function produces a zero-centered output and maps the input values to the range $[-1, 1]$. This means that negative inputs will remain negative, and vice versa, making the activation function suited for training (De Marchi and Mitchell, 2019). Mathematically, the function can be seen in Equation 2.33.

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.33}$$

2.3.5 Loss functions and optimizers

While training the model, the error needs to be quantified. This is performed by the loss function, $\mathcal{L}$. For regression problems where model predicts numerical values, as fatigue estimation, the mean squared error (MSE) and mean absolute error (MAE) are common choices:

$$\text{MSE:} \quad \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2 \tag{2.34}$$

$$\text{MAE:} \quad \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{y}_i - \hat{\mathbf{y}}_i \tag{2.35}$$

where N is the number of samples, $\hat{\mathbf{y}}_i$ is the value predicted by the model and $\mathbf{y}_i$ is the true value.

Optimization is the process of minimizing or maximizing a function, $f(z)$, by altering z. When the loss is quantified during the training process, the neurons' attributes need to be updated accordingly. The ML-model utilizes an optimizer for this purpose, which assists in minimizing the loss function by adjusting the weights and biases.

Adam

Adaptive moment estimation (Adam) is an efficient stochastic optimization method that only requires fist order gradients, and is well suited for problems that are large in terms of data and/or parameters (Kingma and Ba, 2014). For an objective function, $f(\theta)$, that is differentiable with respect to its parameters, θ, the goal is to minimize the expected value of this function, $E[f(\theta)]$. With a time/iteration step, t, the gradient, g_t, is defined as in Equation 2.36.

$$g_t = \nabla_\theta f_t(\theta) \tag{2.36}$$

Adam updates exponential moving averages of the gradient, m_t, and the squared gradient, v_t, where hyperparameters γ_1 and γ_2 controls the exponential decay rates. Mathematically, this becomes as in Equation 2.37 and Equation 2.38, respectively.

$$m_t = \gamma_1 m_{t-1} + (1 - \gamma_1) \cdot g_t \tag{2.37}$$

$$v_t = \gamma_2 v_{t-1} + (1 - \gamma_2) \cdot g_t^2 \tag{2.38}$$

Subsequently, the bias for the first- and second moment is corrected for:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \gamma_1^t} \\ \hat{v}_t &= \frac{v_t}{1 - \gamma_2^t} \end{aligned} \tag{2.39}$$

Lastly, the update parameters are calculated according to Equation 2.40:

$$\theta_t = \theta_{t-1} - \frac{\varepsilon\, \hat{m}_t}{\sqrt{\hat{v}_t} + \hat{\epsilon}} \tag{2.40}$$

where ε is the global learning rate and $\hat{\epsilon}$ is a small number to avoid any division by zero.

RMSProp

RMSProp is popular among DL practitioners, and has proven to be quite effective and practical in relation to deep NNs. The algorithm keeps a moving average of the squared gradient for each weight, and subsequently dividing the gradient by the square root of this value. Mathematically, the algorithm can be expressed as the equation set seen below in Equation 2.41.

$$
\begin{aligned}
r_t &= (1-\gamma) f'(\theta_t)^2 + \gamma r_{t-1} \\
\theta_{t+1} &= \theta_t - \frac{\varepsilon}{\sqrt{r_t}} f'(\theta_t)
\end{aligned}
\tag{2.41}
$$

Where r is the adjustment parameter, γ is the decay rate, θ is the update parameter and $f'(\theta)$ is the derivative of the error with respect to the weight (Roy et al., 2017).

2.3.6 Backpropagation

Backpropagation refers to a training method in ML, where the gradients obtained from the optimizer and loss from the loss function are used to update the weights and biases. The gradient of the loss function, $\mathcal{L}$, is found by differentiation with respect to the weights, $\boldsymbol{w}$. The updated weights are found by moving a small step, ε, in the direction of steepest descent, as seen in Equation 2.42. As previously, ε is the global learning rate (Purkait, 2019). Note the similarity to the RMSProp optimizer in Equation 2.41.

$$
\boldsymbol{w} = \boldsymbol{w} - \varepsilon \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} \tag{2.42}
$$

2.3.7 Epochs and model fitting

One epoch is simply one stream of the entire training dataset. In other words, the number of epochs denotes how many times the model shall see the same training data. By changing the order of the input data for each run, the ML-model may achieve higher accuracy given no change in the dataset. An appropriate number of epochs is highly important in order to accurately predict unseen data. Too few epochs will result in an underfitted model, implying underachievement in terms of accuracy for both the training- and test data. Too many epochs will overfit the model, implying that it will become overly confident on the training data, while the accuracy will decrease as new, unseen data is used as input. A visual representation can be seen in Figure 2.10.

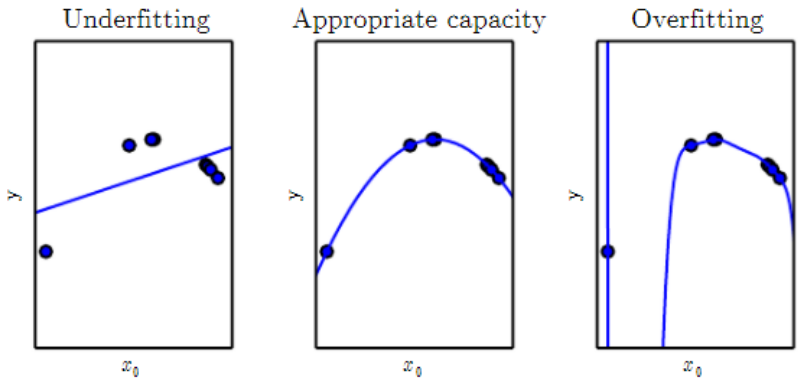


Figure 2.10: Visual representation of model fitting. The leftmost model shows underfitting, which is unable to capture the curvature of the data. The middle figure shows a good generalization of the data, able to accurately predict unseen data. The rightmost figure shows overfitting, where the known data are accurately predicted while the underlying trend is unperceived (Goodfellow et al., 2016).

2.3.8 Data preprocessing

Processing of data before it can be utilized by the ML-model is commonly regarded as one of the most important aspects in ML, as incomplete or duplicate data will lead to a biased model. The most important modifications can briefly be summarized as such:

- Normalize the input data.
- Convert attributes consisting of text-strings to numerical values, commonly denoted categorical values.
- Remove non-satisfactory data.

As mentioned in Subsection 2.3.3, the activation function is determined as the sum of the products between the individual weights and the neuron values. If there is a large difference between the weights and the neuron values, the product will be governed by the neuron value and the model may have a hard time acquiring proper weights. Because the weights initially are given a value in the range [0,1], it is beneficial to normalize the neuron values such that they are in the same range as the weights. Accounting for negative values, the range for neuron values will be [-1,1].

Removal of non-satisfactory data is a wide concept, but the most important aspects will be accounted for. Duplicate data can easily lead to overfitting, as the model gets to comfortable with the known data. Another important point is that if identical samples occurs in both the training- and test set, the model's predictive abilities becomes artificially high. Incomplete data are given as samples where one or more parameter is missing. It is equally important because it introduces room for interpretative variations when feeding the model an incomplete trend. In other words, the ML-model is unable to recognize the mathematical pattern from the data because there is none. Further, as explained in Subsection 2.3.5, the ML-model utilizes some sort of regression to minimize the error. It is therefore recommended to filter for outliers, as their contribution to the mean error will dominate. Practically this means that if any of the variables is disproportional compared to the others, it may be clever to remove the sample.

Chapter 3

Method

This chapter describes the method applied in this thesis, where the methodological structure can be seen in Figure 3.1.

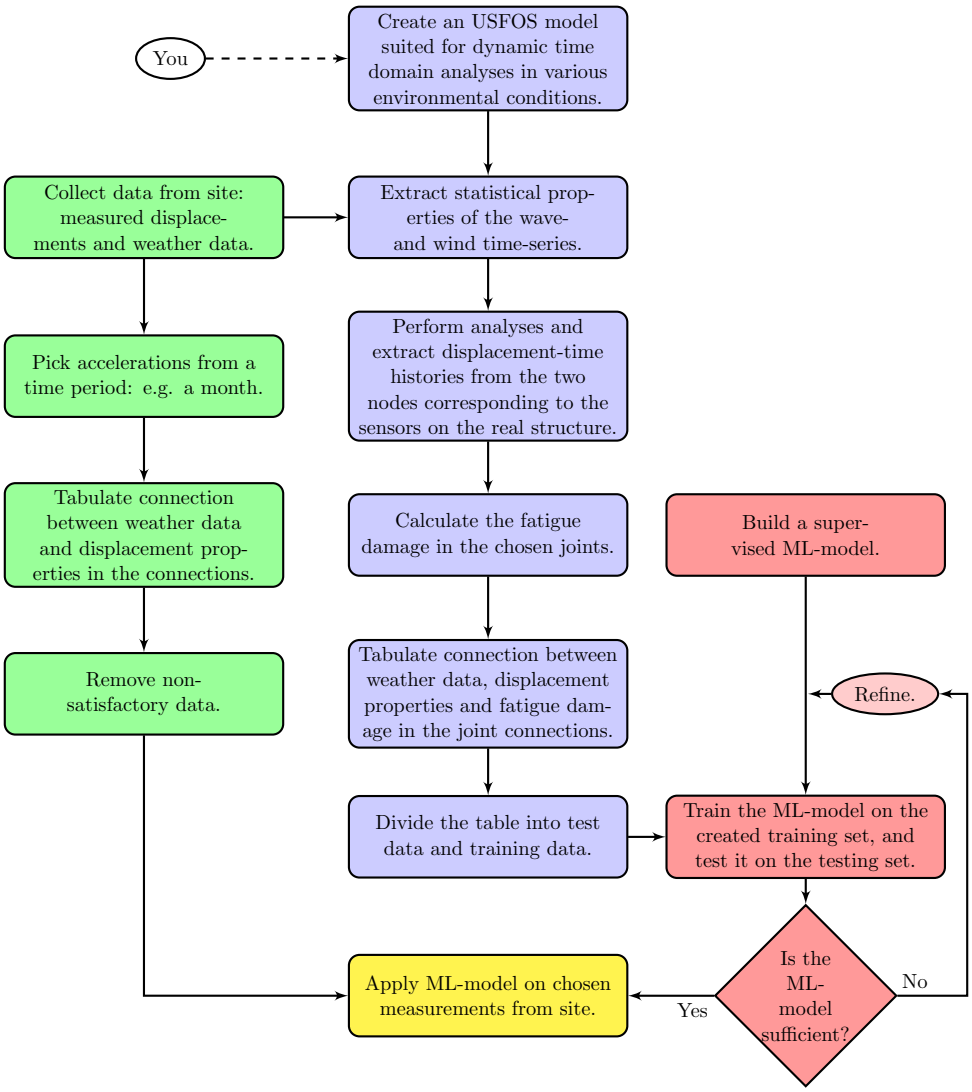
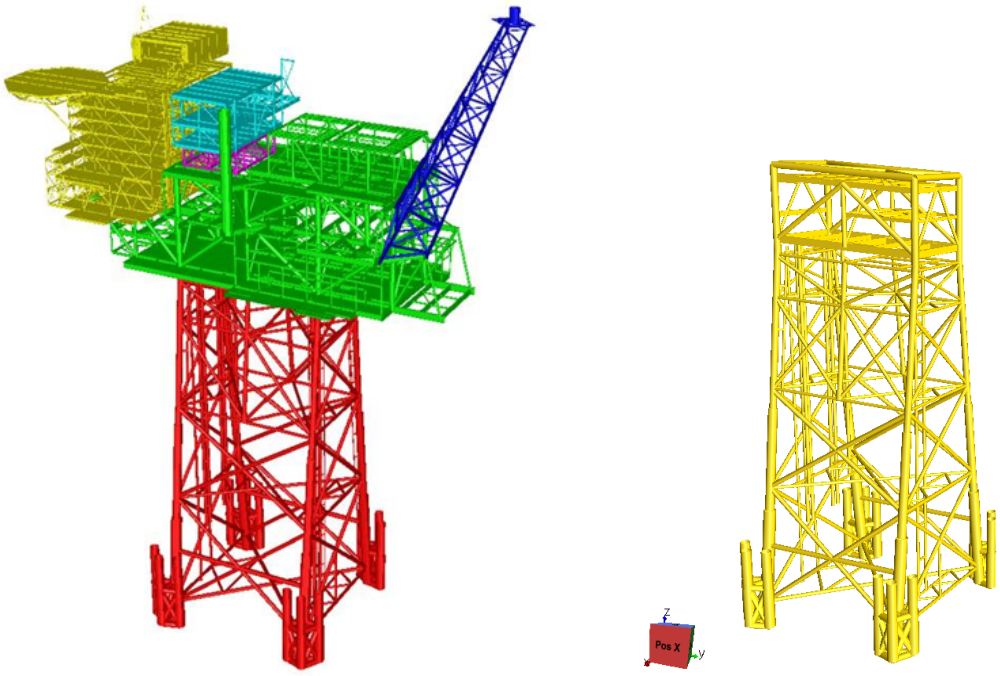


Figure 3.1: Flow chart summarizing the applied method.

As can be seen in Figure 3.1, the whole procedure can be divided into three main categories; data collection, application of ML and testing of a ML-model on monitored data, whereas the last category partly is presented in the results chapter. Firstly, the procedure used to develop a USFOS-model of the jacket structure suited for fatigue analyses in the time domain is described, followed by the simulation- and data collection procedures. Subsequently, a description regarding the process of developing the neural network architecture will be provided, including the training- and testing of the ML-model. Lastly, the procedure of testing the final ML-model on the monitored data provided by Aker BP will be explained.

3.1 USFOS-model description and dataset development

The model used in this thesis is based on the PH-platform located on the Valhall A field in the North Sea. It was made by converting an existing GeniE-model, provided by Aker BP, into a FEM suited for USFOS. Initially, what separated the model from the actual structure is that the actual structure is connected to two nearby platforms by bridges. The bridge connections introduces stiffness from the piping and friction force from the sliding supports. The influence from the bridges was not within the scope of this thesis, and was neglected throughout the simulations.



(a) Original GeniE model provided by Aker BP.

(b) Modified USFOS model used for dynamic time domain simulations.

Figure 3.2: Visual representation of the original model and the modified model. The origin of the coordinate system used in USFOS is located at the center of the platform in the horizontal xy-plane, and at the far bottom of the jacket in vertical z-direction. The reference axes looks skewed, but x-direction is normal to the broad side of the platform, and y-direction is normal to the narrow side.

The two sensors monitoring accelerations are located on the platform topside was represented by two nodes. The location of the sensors was determined by thorough measurements on drawings provided by Aker BP, and can be visualized in Figure 3.3. The actual sensors are capable of measuring accelerations in the platform north-south (NS)-direction and platform east-west (EW)-direction, which then are integrated twice and filtered for noise to get the corresponding displacements. In the USFOS coordinate system, the monitored directions are equivalent to negative x-direction and negative y-direction, respectively. The foundations of which the sensors stands on are tripods highly dominated by axial stiffness. The E-modulus was set 10000 times higher than normal steel, while both the bending stiffness and mass was kept to a minimum, by using thin beam elements and defining a small density different from zero. The coordinates of the nodes can be seen in Table 3.1

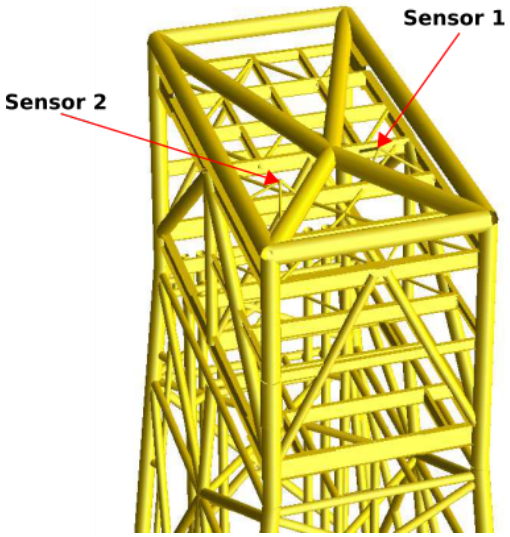


Figure 3.3: Nodal representation of the actual sensors located on the PH-platform. Both sensor 1 and sensor 2 are highlighted. The slightly skewed crossings is the mass tent used to distribute the topside load onto the platform legs.

Table 3.1: Table showing the sensor coordinates in the model coordinate system.

	x [m]	y [m]	z [m]
Sensor 1	-5.28	4.55	125.925
Sensor 2	5.28	7.58	125.295

3.1.1 Model simplifications

To create a sufficient dataset to be used for training, validation and testing of the ML-model, an appropriate amount of dynamic time domain simulations subjected to different environmental conditions had to be performed. As computational efficiency was of the essence, the model had to be stripped down to be as simple as possible without compromising with the structural properties. The simplifications are summarized below:

- Small beams considered irrelevant for the dynamics of the jacket platform, as ladders and hanging tubes, was removed. This was mainly done to reduce the computational

effort needed for each simulation, but small beam elements can also cause numerical instability during the simulations.

- The topside was removed and replaced by a single mass point located at the topside CoG, found in the FLS-report conducted by Aker Solutions and provided by Aker BP.

- Initially, quite a few beam elements along the platform legs, in proximity of joints, had been reinforced with plates, implying that they were thicker than the adjacent beams. Due to the geometrical differences, these beams became very short. A small length-to-diameter ratio results in extremely high terms in the stiffness matrix due to the $\frac{12EI}{L^3}$-term, and can cause numerical instability. As it was important to ensure that the bracing was fully connected to the platform legs through nodes, it was not arbitrary which elements that could be merged. Geometrical adjustments of the beams had to be made, and it was assumed that the reinforcement mainly was meant to increase the platform ultimate limit state (ULS) capacity, and has minor influence on the FLS capacity. Therefore, to ensure conservatism, the thinnest adjacent tubes was extended to replace the ill-conditioned beam elements.

3.1.2 Pile stiffness and topside mass

The pile stiffness was applied according to the FLS-report provided by Aker BP, which was based on a "fatigue-wave" of 6m. As the pile stiffness is of linear nature, it was divided equally onto four springs and applied to the bottom of each platform leg. The total stiffness matrix can bee seen in Table 3.2.

Table 3.2: Table showing pile stiffness matrix for fatigue condition, with units [MN/rad], and [MN/m]

Kx	Ky	Kz	Kxx	Kyy	Kzz
280	0	0	0	-1950	0
0	280	0	1950	0	0
0	0	3300	0	0	0
0	1950	0	21500	0	0
-1950	0	0	0	21500	0
0	0	0	0	0	4200

To properly distribute the topside mass equally onto the legs, as well as deal with the vertical CoG-coordinate, a mass tent was created. The mass tent is characterized by being almost massless and very stiff, such that redistribution of forces due to plastic deformation does not occur. The E-modulus was set 10000 times higher than normal steel, but unlike for the tripods, the bending stiffness was substantial. The total applied topside mass can be seen in Table 3.3.

Table 3.3: Table showing topside mass for fatigue condition, with units [tonne] and [m]. The reference coordinate system is the center of the jacket in the xy-plane and a mean water level of 74.63m in z-direction.

MX	MY	MZ	CoG-X	CoG-Y	CoG-Z
21535	21535	21535	-0.01	5.90	52.03

3.1.3 Structural damping

The structural damping matrix may be defined through Rayleigh damping as a linear combination of the system mass and system stiffness, where $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{K}$ is the stiffness matrix and α_1 and α_2 are the mass-proportional- and the stiffness-proportional coefficient, respectively.

$$\boldsymbol{C} = \alpha_1 \boldsymbol{M} + \alpha_2 \boldsymbol{K} \tag{3.1}$$

Generally, the mass-proportional term damps out the high frequency modes, while the stiffness-proportional term damps out the lower frequency modes (Langen, 1999). In order to apply Equation 3.1, the mass- and stiffness matrices needs to be orthogonal to the eigenvectors, $\boldsymbol{\Phi}$. The modal damping parameter, for mode i, can then be found from Equation 3.2.

$$\bar{c}_i = \boldsymbol{\Phi}^T \boldsymbol{C} \boldsymbol{\Phi} = \alpha_1 \bar{m}_i + \alpha_2 \bar{k}_i \tag{3.2}$$

Subsequently, the damping ratio is defined as seen in Equation 3.3, where c is the modal damping, c_{cr} is the critical damping, $\bar{m}_i$ is the modal mass, $\bar{k}_i$ is modal stiffness and ω is the eigenfrequency for mode i.

$$\lambda_i = \frac{c}{c_{cr}} = \frac{\bar{c}_i}{2\bar{m}_i \omega_i} \tag{3.3}$$

By substituting Equation 3.2 into Equation 3.3, the damping ratio can be rewritten according to Equation 3.4.

$$\lambda_i = \frac{1}{2\bar{m}_i \omega_i} \left(\alpha_1 \bar{m}_i + \alpha_2 \bar{k}_i \right) = \frac{1}{2} \left(\frac{\alpha_1}{\omega_i} + \alpha_2 \omega_i \right) \tag{3.4}$$

λ_1 and λ_2 are typically determined by performing two experiments at two different frequencies, ω_1 and ω_2. When the two damping ratios are known, α_1 and α_2 can be determined according to Equation 3.5 and Equation 3.6, respectively.

$$\alpha_1 = \frac{2\omega_1 \omega_2}{\omega_2^2 - \omega_1^2} (\lambda_1 \omega_2 - \lambda_2 \omega_1) \tag{3.5}$$

$$\alpha_2 = \frac{2(\omega_2 \lambda_2 - \omega_1 \lambda_1)}{\omega_2^2 - \omega_1^2} \tag{3.6}$$

To ensure conservatism, it is important to use frequencies such that the damping is not overestimated at the region of large response. A general example of the behaviour of the damping ratio can be see in Figure 3.4.

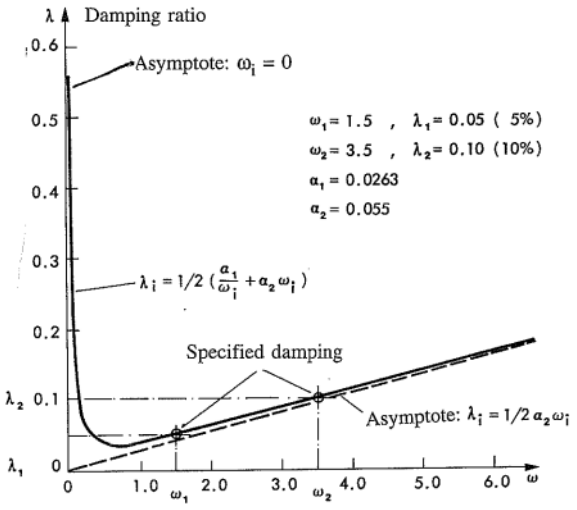


Figure 3.4: General example of damping ratio as a function of eigenfrequency (Langen, 1999).

The three first eigenmodes was available in the FLS-report, equal to 2.77 s, 2.73 s and 2.50 s for zero subsidence, where the damping ratio was given as 1.5 % for all modes. To not overestimate the damping at important frequencies, the mass-proportional coefficient, α_1, was set to zero, such that the damping ratio follows the asymptote:

$$\lambda = \frac{1}{2}\alpha_2\omega_{0,max} = \frac{1}{2}\alpha_2\frac{2\pi}{2.50} \tag{3.7}$$

With $\lambda = 0.015$, the resulting stiffness-proportional coefficient became: $\alpha_2 = 0.012$.

3.1.4 Wave load parameters

The mass- and drag coefficients in Morison's equation explained in Subsection 2.1.6 was determined through the NORSOK standard N-003 seen in Table 3.4 (NORSOK, 2017).

Table 3.4: Chosen values of drag- and inertia coefficients for the main jacket structure and conductors (NORSOK, 2017).

	Jacket structure	Conductor	condition
C_d	1.15	1.15	
C_m	1.6	1.6	above elevation +2 m
C_m	1.2	1.2	below elevation +2 m

3.1.5 Selecting appropriate environmental actions

In this thesis, the appropriate sea states was found by evaluating the hourly wave time-series provided by Aker BP, and wind data extracted from Meteorologisk Institutt. A manual evaluation of the environmental conditions was necessary since the date and time of the critical sea states was essential. The provided wave data does not include direction, and it was therefore assumed that the wave direction followed the wind direction. In order for the USFOS model to replicate the dynamic behaviour of the original model in realistic conditions, statistical evaluations regarding the wave elevation- and displacement data, as well as wind data, had to be made.

Extracting spectral properties of the wave time-series

By assuming that the wave elevation, ζ, is a Gaussian process with zero mean, the autocorrelation function of the wave elevation, $R_{\zeta\zeta}$, is defined as in Equation 3.8 (Newland, 2005).

$$R_{\zeta\zeta}(\tau) = E[\zeta(t)\zeta(t+\tau)] \tag{3.8}$$

The autocorrelation function is related to the wave spectrum through the Fourier transform and inverse Fourier transform, respectively:

$$\begin{aligned} S_{\zeta\zeta}(\omega) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} R_{\zeta\zeta}(\tau)e^{-i\omega\tau}d\tau \\ R_{\zeta\zeta}(\tau) &= \int_{-\infty}^{\infty} S_{\zeta\zeta}(\omega)e^{i\omega\tau}d\omega \end{aligned} \tag{3.9}$$

By utilizing the relation in Equation 3.9, the following properties regarding the autocorrelation function can be defined:

$$R_{\zeta\zeta}(0) = \int_{-\infty}^{\infty} S_{\zeta\zeta}(\omega)d\omega \tag{3.10}$$

$$\begin{aligned} \frac{d^2}{d\tau^2}R_{\zeta\zeta}(0) &= \frac{d^2}{d\tau^2}\left\{\int_{-\infty}^{\infty} S_{\zeta\zeta}(\omega)e^{i\omega\tau}d\omega\right\}\Bigg|_{\tau=0} \\ &= -\int_{-\infty}^{\infty}\omega^2 S_{\zeta\zeta}(\omega)d\omega \end{aligned} \tag{3.11}$$

Numerically, $\frac{d}{d^2\tau}R_{\zeta\zeta}(\tau)$ was calculated by use of the second order central difference formulation, seen in Equation 3.12

$$\frac{d^2}{d\tau^2}R_{\zeta\zeta}(\tau) = \frac{R_{\zeta\zeta}(\tau+\Delta t) - 2R_{\zeta\zeta}(\tau) + R_{\zeta\zeta}(\tau-\Delta t)}{(\Delta t)^2} \tag{3.12}$$

Recognizing the definition of the spectral moments, $m^{(n)}$:

$$m^{(n)}(\zeta,t) = \int_{-\infty}^{\infty} \omega^n S_{\zeta\zeta}(\omega) d\omega, \quad n = 1,2,3... \tag{3.13}$$

Since neither $S_{\zeta\zeta}(\omega)$ or ω is negative for a one-sided spectrum, the following relation between the autocorrelation function and the spectral moments applies:

$$m^{(0)} = R_{\zeta\zeta}(0) \tag{3.14}$$

$$m^{(2)} = \left| \frac{d^2}{d\tau^2} R_{\zeta\zeta}(0) \right| \tag{3.15}$$

By using the result obtained from Equation 3.14, an estimate of the significant wave height can be calculated (Myrhaug, 2019):

$$H_s \approx H_{m0} = 4\sqrt{m^{(0)}} \tag{3.16}$$

Subsequently, the peak period, T_p, can be approximated by Equation 3.17.

$$T_z \approx 1.41 T_{m02} = 1.41 \cdot 2\pi \sqrt{\frac{m^{(0)}}{m^{(2)}}} \tag{3.17}$$

The resulting scatter diagram can be seen in Figure 3.5, and the corresponding scatter table, showing the number of individual sea states, can be seen in Appendix C. The Python code wave_analysis_site.py was used to perform the calculations, and can be found in Appendix E.3.

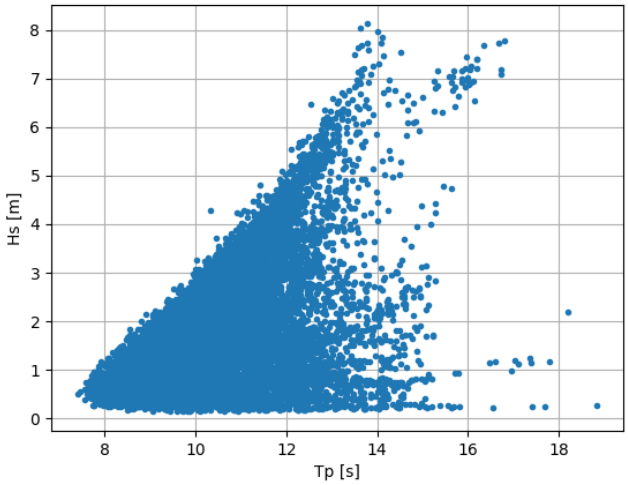


Figure 3.5: Scatter diagram created statistically based on all obtained continuous, hourly wave data between January 2020 and December 2021, provided by Aker BP. Note that some of the points where T_p is disproportionately large relative to H_s might be due to drift-off or discontinuities in the measurements. The largest errors has been filtered away.

Extracting wind data

The wind was extracted from Meteorologisk institutt by use of scripting techniques. The Python code used to extract the data can be found in Google Colab[2]. From the raw data, it was possible to extract the mean wind speed and mean wind direction given in 10-minute intervals. Once extracted, the wind speed and -direction was averaged out to fit the 1-hour intervals used for the waves. An example of the extracted data, averaged to hourly intervals, can be seen in Figure 3.6, where it can be seen that the average wind direction corresponds quite well with the Coriolis effect.

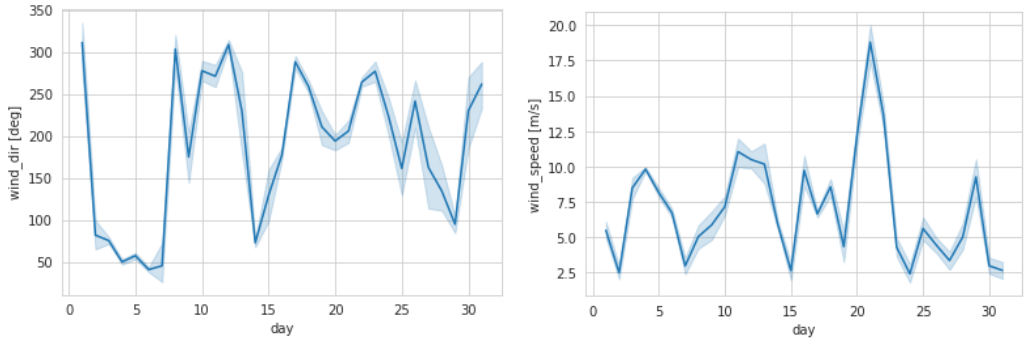


Figure 3.6: Hourly wind direction and -speed for January 2021

[2]https://colab.research.google.com/drive/1PhnctcUIGQchg8-pbFZUKdVV8G4ipPXI?usp=sharing

3.1.6 Simulation procedure

The fatigue simulations were performed through time domain simulations by applying combinations of site relevant environmental loads within a specified time interval. By defining an irregular sea though a spectrum, the time interval needs to be long enough to ensure stationary conditions. In other words, the simulation needs to proceed until the standard deviation, σ, becomes stationary. Jia et al. (2008) used 60 minutes for their time domain simulations of a jacket structure. Due to limited computational resources, a time interval of 15 minutes was consistently used in this thesis, and was assumed to sufficiently represent the statistics of the applied wave spectra. The time increment was set to 0.1 s. The loading due to waves was considered to be governing. Therefore, the wind speed- and direction occurring at the same date and time of the individual sea states was selected. However, the wave direction was considered to be equal the direction of the wind. A total of 132 simulations was performed to create the final dataset. Therefore, the Python script RunUsfos.py found in Appendix E.1, was created to automate the process. The script was optimized to run one simulation per core in the processor, which unfortunately was limited to two on the applied computer. The procedure is summarized in algorithm 1.

Algorithm 1: Algorithm for automating USFOS simulations

```
N ← number of sea states
Create N folders
for i ← 1 to N do
    Run Folder ← run[i]
    move control file → Run Folder
    move structure file → Run Folder
    move load file → Run Folder
    move fatal file → Run Folder
    if Run Folder is folder then
        sea state[i] → control file
        wind load[i] → control file
        if No error in Run Folder then
            run USFOS
            displacements from sensors ← run Dynres
            fatigue damage in hot-spots ← run Fatal
            Delete USFOS results file
        end
    end
end
```

Applying irregular waves through spectrum

The JONSWAP wave spectrum is commonly used in the North Sea, and is considered to be a quite good model for wind generated sea (Myrhaug, 2019). The wave spectrum, $S_{\zeta\zeta}(\omega)$, is generally described by the significant wave height, H_s, and the spectral peak period, T_p (USFOS, 2010):

$$S_{\zeta\zeta}(\omega) = \delta g^2 \omega^{-5} exp\left[-1.25\left(\frac{\omega}{\omega_p}\right)^{-4}\right]\gamma^{exp\left(-\frac{(\omega/\omega_p-1)^2}{2\sigma^2}\right)} \tag{3.18}$$

where ω_p is the peak period defined according to Equation 3.19.

$$\omega_p = \frac{2\pi}{T_p} \tag{3.19}$$

γ is the peak enhancement factor given in Equation 3.20.

$$\begin{aligned} \delta &= 0.036 - \frac{0.0056 T_p}{\sqrt{H_s}} \\ \gamma &= exp\left[3.483\left(1 - \frac{0.1975\delta T_p^4}{H_s^2}\right)\right] \end{aligned} \tag{3.20}$$

The amplitude of each wave component is determined according to Equation 3.21;

$$\zeta_{a,j} = \sqrt{2\int_{\omega_{l,j}}^{\omega_{u,j}} S_{\zeta\zeta}(\omega) d\omega} \tag{3.21}$$

where $\omega_{l,j}$ and $\omega_{u,j}$ represents the lower- and upper angular frequency limit for wave component j, respectively. In this thesis, the frequency range was divided into $N = 120$ intervals of equal length, such that $\Delta\omega = \frac{\omega_u - \omega_l}{N}$, where:

- $\omega_u = \frac{2\pi}{T_{max}}$ where T_{max} was chosen to be $(T_p + 4)$.
- $\omega_l = \frac{2\pi}{T_{min}}$, where T_{min} was chosen to be 4 for all sea states.

The JONSWAP spectrum is only valid for the combinations of H_s and T_p that satisfies the requirement in Equation 3.22.

$$3.6\sqrt{H_s} < T_p < 5\sqrt{H_s} \tag{3.22}$$

Due to the limited validity area of combinations of H_s and T_p, the JONSWAP spectrum is not valid for all the sea states obtained in Figure 3.5. However, only some of the most dramatic sea states was used throughout the simulations, where the JONSWAP spectrum is valid.

Applying wind field to model

To apply wind in the USFOS simulations, the wind direction was modified to comply with the USFOS coordinate system. From the mean water level and above, the wind field was applied as a z-varying curve given in Equation 3.23

$$U(z) = U_{10}\left(\frac{z}{10}\right)^{p} \tag{3.23}$$

where U_{10} is a constant denoting the mean wind 10 m above mean surface elevation from the extracted data, and p is a factor to describe the wind profile. Since the topside was removed, some care had to be shown when replicating the wind load as accurately as possible. The static wind load is calculated as shown in Equation 3.24

$$F_{wind} = \frac{1}{2}\rho_w C_d v_w^2 A \tag{3.24}$$

where ρ_w is the wind density, C_d is the drag coefficient, v_w is the wind velocity and A is the area. Since the platform is asymmetric in x-direction and y-direction, unique scaling factors has to be multiplied with the wind velocity in both x- and y-direction. The scaling factors was found by requiring that the wind load before and after topside removal was approximately equal, resulting in an iterative procedure. With assistance from my supervisors and Atle Nøsen from Aker BP, and evaluations of drawings and the original model, an Excel spreadsheet calculating the base shear load and overturning moment due to topside wind only was developed, and used as reference for the wind velocity scaling. The spreadsheet follows the procedure found in NORSOK N-003 (NORSOK, 2017). Since both the base shear load and the overturning moment had to be equal the prior values, Equation 3.24 cannot just be rearranged with respect to the velocity. The appropriate scaling factors was found through simulations in USFOS where a wind field was the only applied environmental action. The obtained results can be seen in Table 3.5.

Table 3.5: Wind speed scaling factors for wind with 0 degree heading and 90 degree heading. BSL denotes the base shear load, OM denotes the overturning moment, $\hat{x}$ denotes x-direction, $\hat{y}$ denotes y-direction and p is the wind profile power in Equation 3.24.

Heading	Goal BSL	Sim. BSL	Goal OM	Sim. OM	Scale, $\hat{x}$	Scale, $\hat{y}$	p
0 deg	0.52 MN	0.53 MN	65.8 MNm	63.5 MNm	3.5	3.556	1.5
90 deg	0.39 MN	0.40 MN	48.3 MNm	47.5 MNm	3.5	3.556	1.5

Combinations of environmental actions

The choice of environmental conditions used to train- and test the ML-model are based on some of the most critical sea states during the two-year period when monitored data has been available. This remedy was done to not compromise with the JONSWAP spectrum applicability, and force the ML-model to provide somewhat conservative estimates. However, some values of the mean wind speed has been altered slightly for each simulation, together with the wind- and wave directions. This was done in order to encapsulate all possible directions and avoid highly correlated input features, which will be elaborated in Subsection 3.2.2. The environmental conditions used for the simulations are summarized in Table 3.6:

Table 3.6: All 132 environmental conditions used for the simulations. The wind/wave heading annotation implies that a total of eleven simulations was performed for each combination of H_s and T_p.

H_s [m]	T_p [s]	mean wind speed [m/s]	wind/wave heading [deg]
6	12	± 19.1	first:15, step:30, last:345
7	13	± 19.6	first:15, step:30, last:345
8	14	± 20.9	first:15, step:30, last:345
7	14	± 20.8	first:15, step:30, last:345
6	13	± 18.7	first:15, step:30, last:345
5	12	± 18.4	first:15, step:30, last:345
5	11	± 17.5	first:15, step:30, last:345
8	13	± 22.1	first:15, step:30, last:345
4	10	± 12.8	first:15, step:30, last:345
6	9	± 12.8	first:15, step:30, last:345
5	8	± 12.8	first:15, step:30, last:345

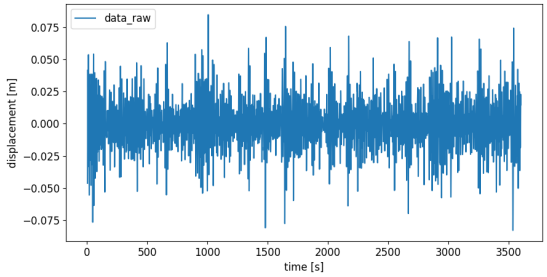
3.1.7 Comparison of displacement

In order to verify that the displacements obtained from USFOS reasonably matches the measured displacements, it is advantageous to compare the two through time-series. Obviously, phase is impossible to compare given that the platform is subjected irregular wave load through a spectrum and the inherent randomness in real-life wave elevation. But the magnitude is comparable. To do so, it is important to choose identical environmental conditions, preferably quite dramatic. The wind influence on platform displacement was considered to be of secondary importance, and follows the time and date of the sea state. Hence, the tabulated sea state in Table 3.7, selected from the scatter table in Appendix C, was applied.

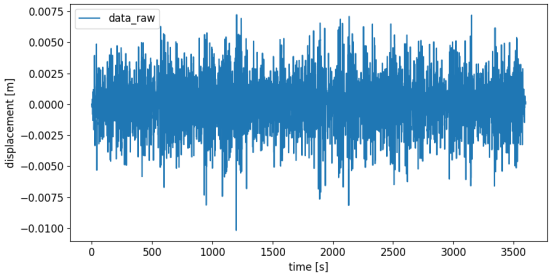
Table 3.7: Environmental conditions used for the simulation comparing measured and simulated displacements. To comply with the measurement intervals, the duration is one hour. The wind direction is given as (compass angle / USFOS angle).

H_s	8 m
T_p	13 s
Mean wind speed	22.1 $\frac{\text{m}}{\text{s}}$
Wind dir	(214.5 deg / -27.5 deg)
Date	21.01.2021
Hour	13:00-14:00

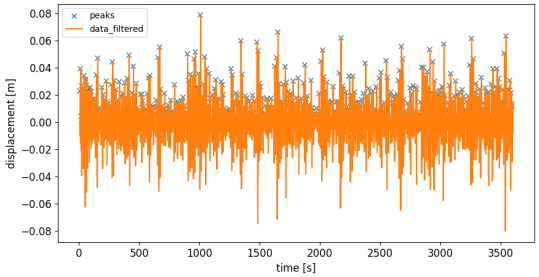
The obtained results can be seen in the figures below, where the static displacement is subtracted for all time-series. To get smoother functions and more distinct peaks for the distribution fitting described in Section 3.1.9, the displacement time-series was slightly filtered by applying a low pass filter. Since subsidence and the influence from the connecting bridges was neglected, the simulated displacement should be slightly higher. The bridges especially restrains x-directional motion, which entails that the y-directional displacement is more alike. The most dramatic environmental conditions during January 2021, shown in Appendix D, occurred on the 21st. However, the most extreme monitored jacket response, equal to about 0.03 m, was missed by few hours and resembles the y-directional displacement quite a lot. The following hours had almost identical environmental parameters, but caused greater translational motions of the real structure.



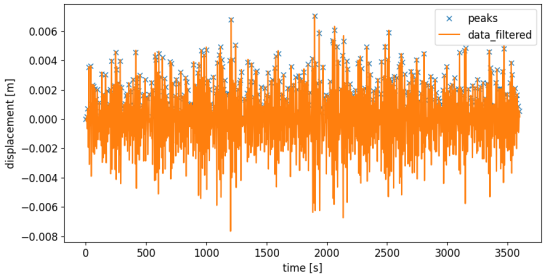
(a) Raw displacement from USFOS



(b) Raw monitored displacement

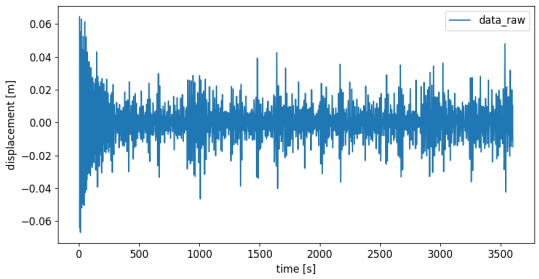


(c) Filtered displacement from USFOS

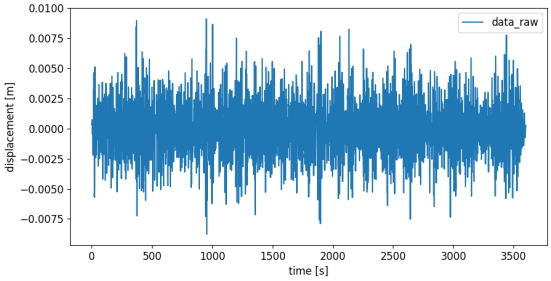


(d) Filtered monitored displacement

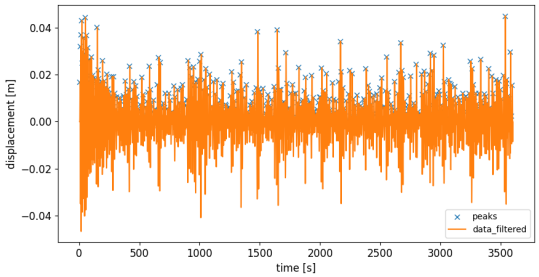
Figure 3.7: Displacement comparison in x-direction for sensor 1.



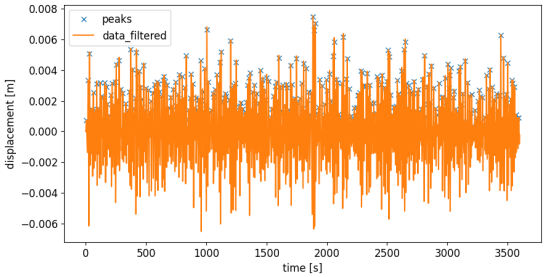
(a) Raw displacement from USFOS



(b) Raw monitored displacement

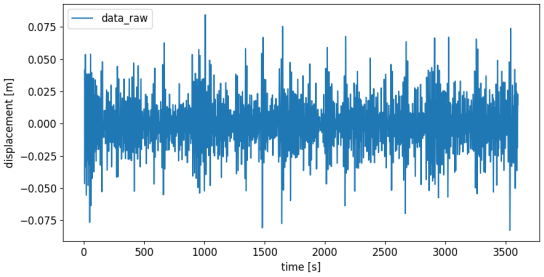


(c) Filtered displacement from USFOS

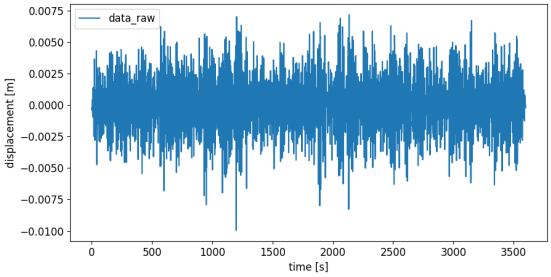


(d) Filtered monitored displacement

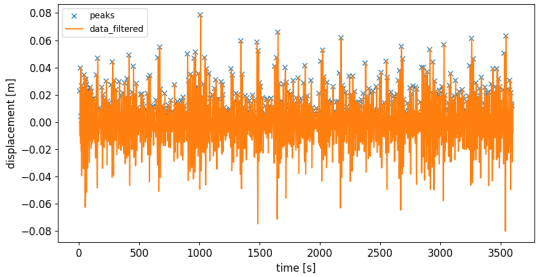
Figure 3.8: Displacement comparison in y-direction for sensor 1.



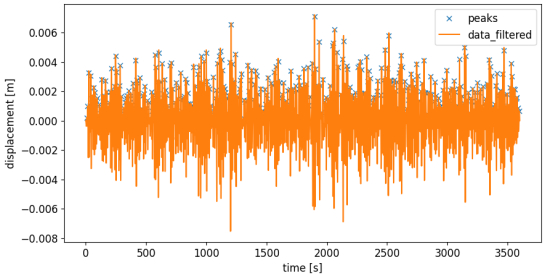
(a) Raw displacement from USFOS



(b) Raw monitored displacement

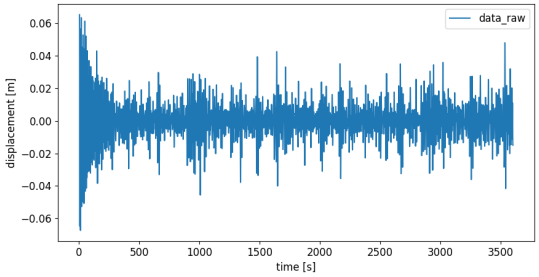


(c) Filtered displacement from USFOS

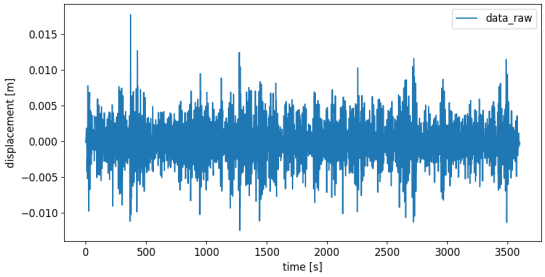


(d) Filtered monitored displacement

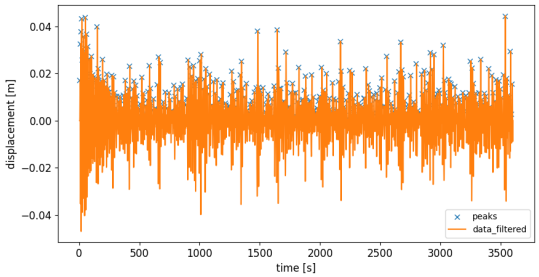
Figure 3.9: Displacement comparison in x-direction for sensor 2



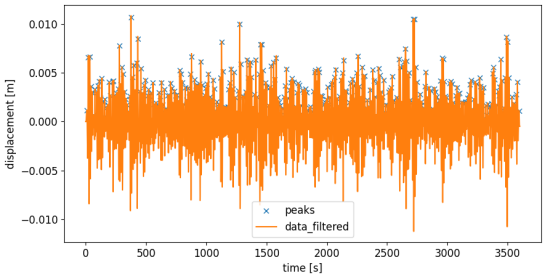
(a) Raw displacement from USFOS



(b) Raw monitored displacement



(c) Filtered displacement from USFOS



(d) Filtered monitored displacement

Figure 3.10: Displacement comparison in y-direction for sensor 2

3.1.8 Fatigue damage calculations

The USFOS-module Fatal, which allows the user to directly perform fatigue damage calculations on specified joints, was used to calculate the fatigue damage. The procedure follows the theory covered in Section 2.2, with parametric equations and built-in or user-defined SN-curves. Fatal was set to skip 60 time-steps, equal to six seconds, to prevent that the initialization of gravity and loads introduced extreme half stress cycles once rainflow counting was applied. The importance can be visualized by looking at the large displacements occurring at the start of the simulations in Subsection 3.1.7.

Critical joints

The joints assumed critical in this thesis, was based on the FLS-report conducted by Aker Solutions. The joints with the lowest estimated fatigue life was selected, and will be compared in the testing of final ML-model- in the result chapter. The connection between joint indexation and the node number in the USFOS model can be seen in Table 3.8.

Table 3.8: Connection between joint indexation for data tabulation and nodes in the USFOS model. The order of which the braces are written corresponds to the subindexation in Subsection 3.1.10.

Joint index	Node number	Connecting braces
1	22	91, 108, 269, 281, 294
2	34	89, 95, 271, 275, 319, 322
3	44	63, 100, 279, 283, 285
4	56	97, 102, 273, 277, 314, 320
5	137	270, 274, 413, 425, 495, 507, 509
6	181	272, 284, 415, 440, 468, 482, 490
7	188	278, 282, 431, 442, 465, 471, 488
8	191	276, 280, 423, 433, 492, 505, 506
Total number of connections:		50

Selecting appropriate SN-curve

As covered in Subsection 2.2.2, the peak stress from the weld geometry is of minor significance for the fatigue life of tubular joints. Therefore, a single curve, the T-curve, also used by Aker Solutions in their FLS-report, is used. The T-curve was therefore applied throughout the simulations in this thesis, and is tabulated in Table 3.9.

Table 3.9: T-curve for tubular joints in seawater with cathodic protection (DNV, 2014).

Weld class/ SN-curve	$N \leq 10^6$ cycles		$N > 10^6$ cycles	Fatigue limit at 10^7 cycles[3]	Thickness exponent	SCF in the weld detail[4]
	m_1	$log\bar{a}_1$	$m_2 = 5.0$ $log\bar{a}_2$		k	
T	3.0	11.764	15.606	52.63	0.25[5]	1.00

[3] Largest local stress range corresponding to $N = 10^7$ cycles. The reader is referred to Section 2.11 in DNV-RP-C203 (DNV, 2014).

[4] As derived by the hot-spot method.

[5] $k = 0.30$ for SCF > 10

3.1.9 Data collection procedure

A key part of this thesis was the procedure of selecting- and collecting data considered optimal for the ML-model, and simultaneously was easily accessible through monitored data. For each simulation, key data had to be extracted and processed in some way to be utilized by the ML-model. The applied method was performed with the Python script post_process_Weibull.py in Appendix E.2, summarized in algorithm 2. The data table was exported to an Excel-spreadsheet in order to be accessed at a later time by the ML-model.

Algorithm 2: Algorithm for data collection and -tabulation

```
Data: Displacement time-series, fatigue damage in critical joints, significant wave
      height, peak period, wind direction, mean wind speed
Result: Excel-table for ML
Table ← empty
N ← number of simulations/environmental conditions
for i ← 1 to N do
    Run Folder ← simulation[i]
    Store Hs[i], Tp[i], wind speed[i] and wind direction[i] in Table
    for All displacement time-series in Run Folder do
        Apply low pass filter
        peaks ← find peaks[displacement]
        k, lambda ← fit Weibull distribution to peaks
        Store k and lambda in Table
    end
    for All critical connections defined in Fatal do
        Classify corresponding joint
        val ← maximum fatigue damage in connection
        pos ← hot-spot location of maximum damage
        Store val and pos in Table
    end
end
Table → Excel
```

Selecting appropriate fatigue damage in hot-spots

For each connection specified in the Fatal control file, the fatigue damage among eight hot-spots along the circumference of the tubes, four along the brace side and four along the chord side, is tabulated in a text-file. The positions along the circumference of the tubes are 0 °, 90°, 180° and 270° for both the chord- and the brace side. The maximum fatigue damage along the eight positions was, in compliance with the procedure of Jia et al. (2008), set at the one-year fatigue damage, and was together with the position tabulated in the dataset utilized by the ML-model.

Fitting distribution to displacements

Given the way fatigue calculations are performed in USFOS through Fatal, described in Section 2.2, only the stress ranges and number of cycles are considered by application of Rainflow counting. Recognizing the relationship between stress and displacement, described in Subsection 2.1.2, it was considered important to capture the underlying statistics

of the displacement time-series. As most of the simulated- and monitored displacement time-series was somewhat symmetric about the x-axis, it was considered sufficient to only fit a distribution to the positive peaks.

The largest maximum between two adjacent zero-up-crossings can generally be described reasonably well by the 2-parameter Weibull distribution, within a stationary period (Haver, 2019). The cumulative distribution function (CDF) is defined according to Equation 3.25:

$$F_X(x) = 1 - exp\left\{-\left(\frac{x}{\lambda_d}\right)^{k_d}\right\} \tag{3.25}$$

where λ_d is the scale parameter and k_d is the shape parameter. The process of fitting the peaks was done by using the method of moments. For a sample of size N, the method of moments is defined by requiring that the sample moments, in Equation 3.26, are equal to the population moments in Equation 3.27.

$$\begin{aligned} \text{Sample mean:} \quad & E[\hat{X}] = \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \\ \text{Sample variance:} \quad & Var[\hat{X}] = S_X^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \end{aligned} \tag{3.26}$$

$$\begin{aligned} \text{Population mean:} \quad & E[X] = \lambda_d \Gamma\left(1 + \frac{1}{k_d}\right) \\ \text{Population variance:} \quad & Var[X] = \lambda_d^2\left[\Gamma\left(1 + \frac{2}{k_d}\right) - \Gamma^2\left(1 + \frac{1}{k_d}\right)\right] \end{aligned} \tag{3.27}$$

Here, $\Gamma()$ is the Gamma function. The distribution parameters, λ_d and k_d, are found by inserting the values obtained from Equation 3.26 into Equation 3.27. By rearranging the population mean with respect to λ_d, the equation for population variance can be solved iteratively for k_d.

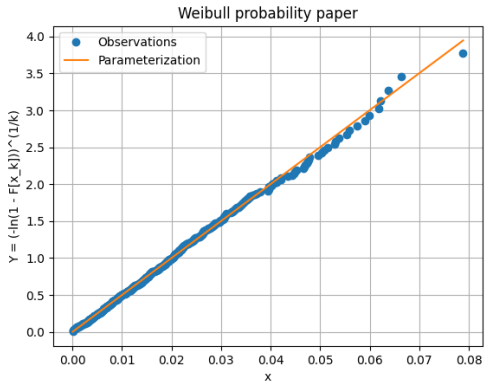
To visualize whether or not the Weibull distribution fits the displacement peaks in a sufficient manner, it is advantageous to plot them in a probability paper. A probability paper is created by rearranging the CDF to get it on the form $Y(x) = mx + b$, where m is the slope and b is the intercept. Applying this to the 2-parameter Weibull CFD yields:

$$Y = [-ln(1 - \hat{\hat{F}}_X(x))]^{\left(\frac{1}{k_d}\right)}, \quad m = \frac{1}{\lambda_d}, \quad b = 0 \tag{3.28}$$

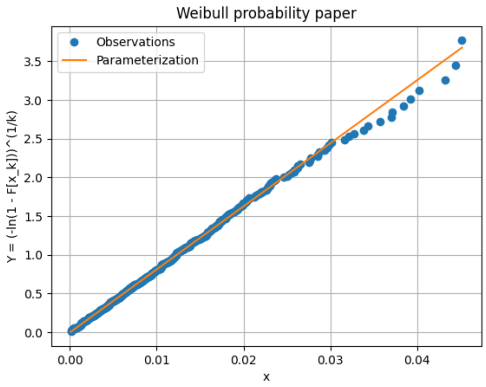
where $\hat{\hat{F}}_X(x)$ is the sample distribution of the sorted sample, defined according to Equation 3.29.

$$\hat{\hat{F}}_X(x_i) = \frac{i}{N+1}, \quad i = 1, 2, ..., N \tag{3.29}$$

If the distribution fits the data in a proper manner, the data trend should resemble a straight line in the probability paper. The resulting probability papers from the same simulation as used for the displacement comparison in Subsection 3.1.7 are shown below:

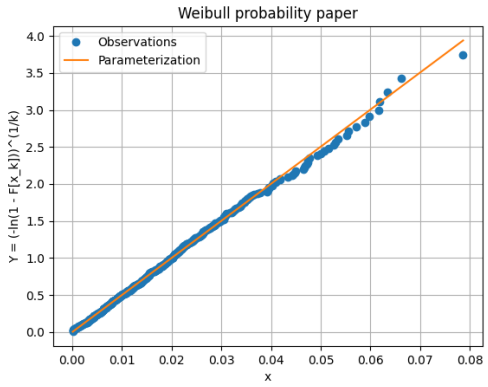


(a) x-directional displacement.

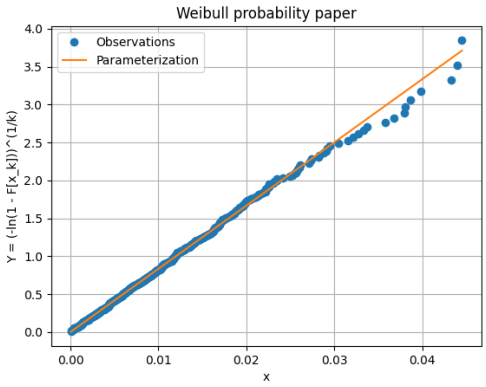


(b) y-directional displacement.

Figure 3.11: Probability paper of 2-parameter Weibull distribution fitted to filtered displacement in USFOS x- and y-direction for sensor node 1.



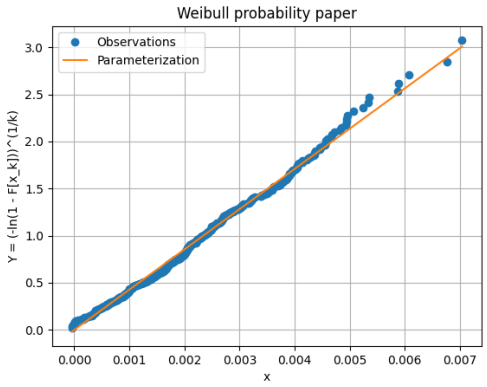
(a) x-directional displacement.



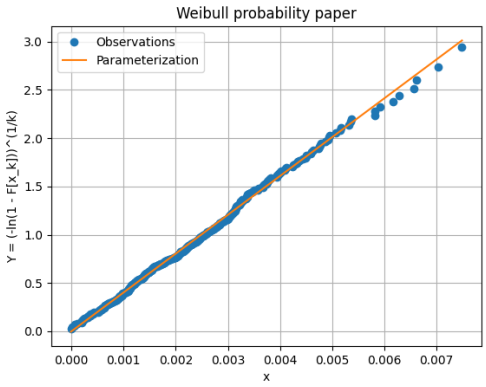
(b) y-directional displacement.

Figure 3.12: Probability paper of 2-parameter Weibull distribution fitted to filtered displacement in USFOS x- and y-direction for sensor node 2.

To verify that the 2-parameter Weibull distribution also is fit to describe the monitored displacements, the probability papers from the same displacement comparison can be seen below:

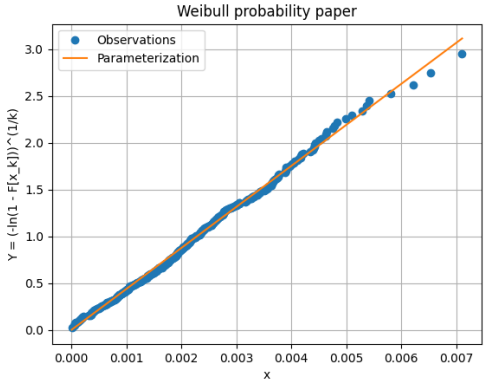


(a) x-directional displacement.

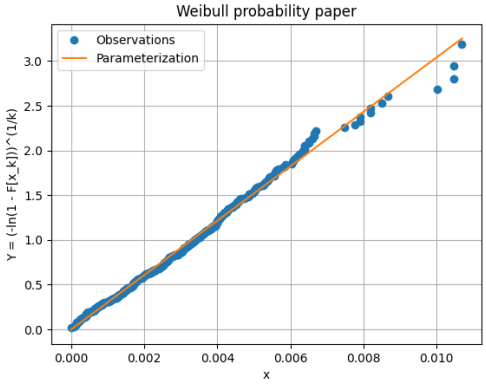


(b) y-directional displacement.

Figure 3.13: Probability paper of 2-parameter Weibull distribution fitted to filtered displacement in USFOS x- and y-direction for sensor 1.



(a) x-directional displacement.



(b) y-directional displacement.

Figure 3.14: Probability paper of 2-parameter Weibull distribution fitted to filtered displacement in USFOS x- and y-direction for sensor 2.

3.1.10 Data tabulation

A total of 132 dynamic time domain simulations was performed to extract a sufficient amount of data to train and test the ML-model. When established, the dataset was organized in the following manner. The top row in the dataset are the headers of the columns, which describes the attributes of the columns. The individual columns represents features of the dataset. One single row denotes one complete observation, often called sample, and can contain either numerical- or categorical quantities. Since the model should predict the damage among 50 connections, each simulation contributed with 50 samples. This implies that duplicate data was unavoidable, which, as covered in Subsection 2.3.8, can be regarded as questionable. The data considered relevant for each sea state needs to provide relevant, unique information and be easily accessible without performing any dynamic time domain simulations. The parameters included for each simulation are listed below:

- Index column denoting the critical joints, with unit [-].

- Subindex column denoting all the braces connected in the respective joints, with unit [-].
- Wave data: H_s and T_p describing the characteristics of the 1-hour sea state, with units [m] and [s].
- Wind data: mean wind speed and -direction during 1 hour, with units $[\frac{\mathrm{m}}{\mathrm{s}}]$ and [deg].
- Shape- and scale parameter of the 2-parameter Weibull distribution fitted to the displacement peaks, with units [-].
- Yearly fatigue damage for all hot-spots, given stationary environmental conditions, with unit [-].
- Position around the circumference of the tube where maximum fatigue damage occurred, with unit [-]. This parameter was only included for visualization purposes, as it is a simulation based property.

The tabulated data used for training, validation and testing can be found in Google Drive[6].

[6] https://docs.google.com/spreadsheets/d/1hJBdczmNNAVhqVi3PRBdpXuD5ek32cME/edit?usp=sharing&ouid=110407881319023833523&rtpof=true&sd=true

3.2 Machine learning

As mentioned in the introduction, the ML-model is limited to quantification of fatigue damage in connections in critical joints, and is therefore exempt from compound problems which is present in for instance image classification. This allows for a single, quite simple ML-model, where an ordinary FFNN is sufficient. The obvious advantages of a simple NN is the time consumption during training and testing, as well as replicability in similar studies.

An important thing to note when trying to predict such small numbers as fatigue damage, with a vast difference in relative magnitude and very specific, is that accuracy is difficult to assess directly. In other words, if one is to estimate the accuracy from a binary perspective, the model will most certainly perform poorly. Therefore, the model performance was evaluated through the following points:

- Through MSE and/or MAE.
- Accurate estimations of the highest fatigue damage values.
- Generally conservative estimates of fatigue damage.
- Through loss.

3.2.1 Environment setup

The programming language Python was used to establish the ML-model. Python is known for its ease of use, and offers a vast amount of libraries. TensorFlow is an open-source library within ML developed and maintained by Google, and made public in November 2015. In addition Keras was utilized. Keras is a library used as an extension of TensorFlow, providing a more abstract interface for the user, easing the process of building the NN (De Marchi and Mitchell, 2019).

Moreover, the ML code was written using Google Colaboratory, Colab for short. Colab allows the user to execute Python program directly in the browser without having to download the needed libraries. In addition, the code can be executed piecewise, simplifying the post processing as the most computationally demanding parts does not have to be re-executed. Using Colab for ML tasks is also beneficial since it allows utilization of graphics processing units to increase computational power. Lastly, established work can be shared across computers by saving it in Google Drive.

3.2.2 Processing tabulated data

The data obtained from the simulations consist of 6600, with 15 attributes. Each sample contains 14 numerical attributes and one categorical. To understand how the preprocessing of the data is done, it is advantageous to see a description of the attributes at an early stage. This is shown in Table 3.10, where the position around the circumference is neglected since it only was used for data visualization.

Table 3.10: Description of dataset. As previously, $\hat{x}$ denotes x-direction and $\hat{y}$ denotes y-direction.

Jacket fatigue damage dataset		
No.	Attributes	Values
1	Connection	1 - 50
2	Joint	1 - 8
3	λ_d sensor 1, $\hat{x}$	0.002290 - 0.023552
4	λ_d sensor 1, $\hat{y}$	0.006905 - 0.055723
5	λ_d sensor 2, $\hat{x}$	0.002317 - 0.023562
6	λ_d sensor 2, $\hat{y}$	0.006936 - 0.055723
7	k_d sensor 1, $\hat{x}$	0.378170 - 3.211260
8	k_d sensor 1, $\hat{y}$	1.049538 - 5.486857
9	k_d sensor 2, $\hat{x}$	1.341414 - 3.248437
10	k_d sensor 2, $\hat{y}$	1.049772 - 5.520304
11	H_s	4 - 8
12	T_p	8 - 14
13	Wind/wave direction	15 - 345
14	Mean wind speed	8.3 - 23.1
Output values		
1	Damage	2.7543e-10 - 7.185e-04

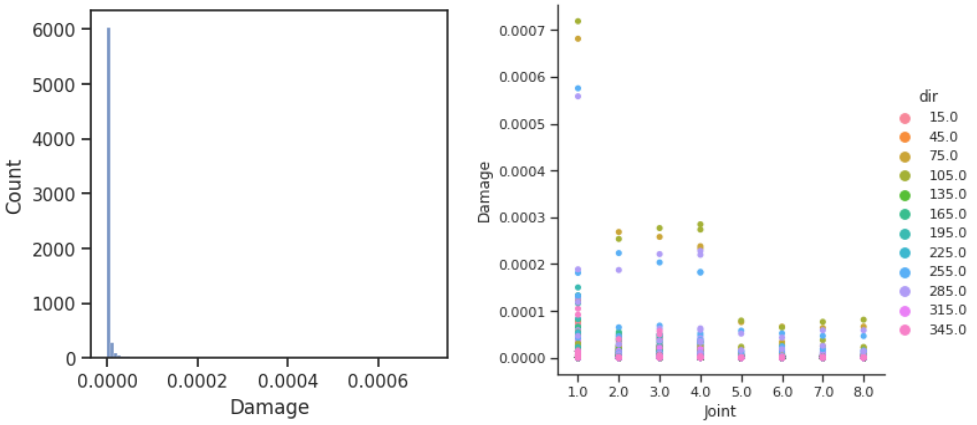
Data scaling

The MinMaxScaler is a common function to scale the input parameters to be in the range [0,1] for ANNs, and was used for all the input features in this thesis. Mathematically, the scaling is given by Equation 3.30.

$$X_{i,new} = \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{3.30}$$

where X_{min} is the minimum value in the feature column, X_{max} is the maximum value in the feature column and X_i denotes feature number i for feature column X.

It is not common practice to scale the output feature. However, as can be seen in both in Table 3.10 and Figure 3.15, the ML-model will have a hard time distinguishing the individual samples. Additionally, the regression is likely to be heavily influenced by the vast amount of extremely small damage values.



(a) Histogram (b) Categorical plot

Figure 3.15: Unscaled fatigue damage illustrating the problematic distribution of the tabulated fatigue damage.

Therefore, a few remedies was performed. These are summarized below:

1. Remove the 500 samples with the smallest fatigue damage. Updated sample size: 6100.
2. Remove samples where the fatigue damages above 0.0002, since they can be considered outliers and strictly restrict the range by being disproportionately large. Updated sample size: 6083.
3. Scale the fatigue damage to get a greater dispersion within the range [0,1] by dividing on the new maximum value equal to 0.00018884.
4. Remove all scaled samples where the fatigue damage is below 0.0002. Updated sample size: 4721.

It is important to mention that by scaling the output feature, the model gets trained to predict the output wrongly, and both the true damage and the predicted damage needs to be multiplied with the scaling factor accordingly. The reason behind this measure is explained in Subsection 3.2.7.

Attribute vectorization

In ML it is common practice to vectorize directional attributes. This is because it is difficult for the NN to grasp that 0 deg represents the same as 360 deg. Hence, the direction was vectorized and multiplied with H_s to create an x-component, $H_{s,x}$, and a y-component, $H_{s,y}$.

Data correlation

A goal in regression analysis is to ensure that the independent variables provides unique information, and only relates to the dependent variable, which in this case is the fatigue damage. Multicollinearity occurs when independent variables associates with changes in

another. Multicollinearity in increasing order introduces problems cumulatively during model fitting, as a small change in an independent variable can cause drastic changes in the model output. Uncorrelated input features can contribute to reduce the bias and decrease the computational time of the learning algorithm. A heat map showing the correlation matrix between the features of the dataset is shown in the Figure 3.16.

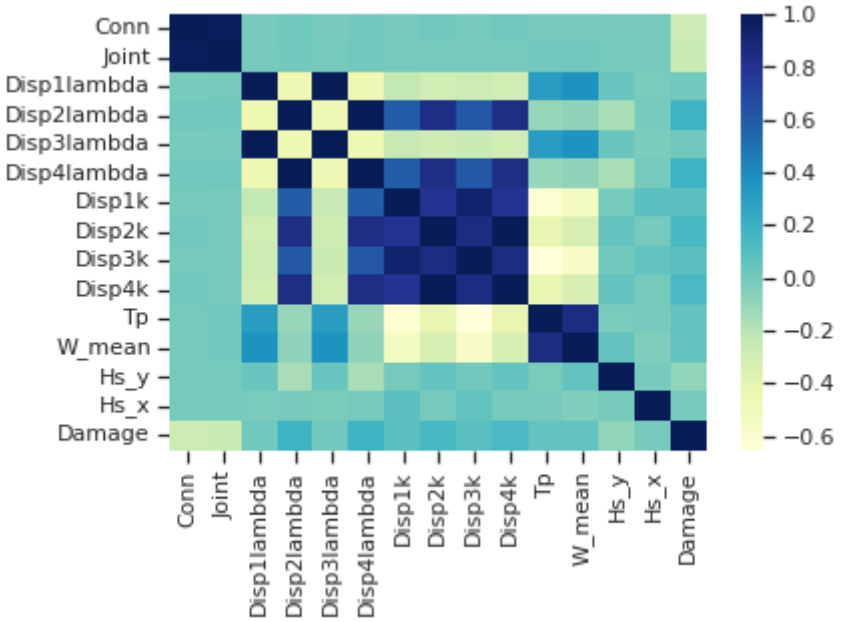


Figure 3.16: Correlation matrix of the jacket fatigue damage dataset.

From Figure 3.16 it can be seen that quite a few of the input features are extremely correlated, especially the scale- and shape parameters of the fitted Weibull distributions. The consequences of removing correlated attributes are investigated further in Subsection 3.2.6.

Training-, validation- and test set

The training- and validation set contains 4248 samples, analogous to 90 % of the total dataset. 90 % of the samples was used to fit the model. The remaining 10 % of the set was defined to be used for validation, to get continuous unbiased evaluations of the model fitting while tuning the hyperparameters.

The test set contains 473 samples, analogous to 10 % of the total dataset. The test set was used to evaluate the fitted model on unseen data.

3.2.3 Training the model

The machine learning algorithm used to train the model is explained in algorithm 3, whereas the whole script, including the validation and testing, can be found in Google Colab[7].

Algorithm 3: Algorithm for machine learning model training

```
Dataset ← Read Excel
Dataset ← Preprocess Dataset
Training set, test set ← Split Dataset 90%-10%
Initialize weights
for epoch ← 1 to end do
    for sample ← 1 to sample size(training set) do
        Predicted damage ← Predict(sample input)
        Loss ← Calculate loss(True damage, Predicted damage)
        if modulus(sample) == batch size then
            Calculate loss gradients
            Calculate optimizer gradients
            Update model parameters
        end
    end
end
```

3.2.4 Experiments with model architecture and epoch sensitivity

The FFNN-architecture seen in Table 3.11 was used as a foundation for the experiments. All layers are applied variance scaling as the kernel initializer, which determines the initial weights. For regression problems, the initialized weights are usually normally distributed, but it was found rather quickly that it led to overfitting already after a few epochs, and thus no reason to investigate further. For all the experiments, a batch size of 32 was used, denoting the amount of samples the model processes before updating the model weights. A small batch size is memory efficient, but may increase the total computational time used to fit the model. The optimizer was chosen to be Adam with default settings, and the loss function was MAE.

Table 3.11: Layer characteristics of model.

Layer	Layer type	Shape	Activation function	Kernel initializer
1	Dense	(,14)	ReLU	Variance scaling
2	Dense	(,32)	ReLU	Variance scaling
3	Dense	(,32)	ReLU	Variance scaling
4	Dense	(,32)	ReLU	Variance scaling
5	Dense	(,1)	-	-

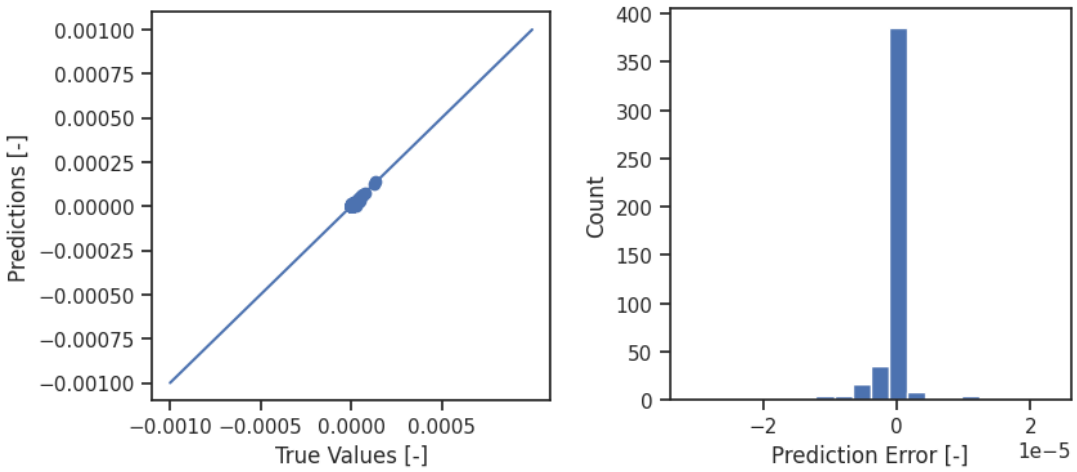
Note that the output layer in the model lacks an activation function. This is an important difference to NNs applied to linear regression problems. The initial part of the optimization of the ML-model consisted of figuring out the optimal number of epochs. The obtained results are shown in Table 3.12.

[7] https://colab.research.google.com/drive/1QWduwe9wJYdkeLpzqOWKTO9RPDfhePCA?usp=sharing

Table 3.12: Results from experimenting with the optimal number of epochs needed by the ML-model. MSE and MAE are calculated with respect to the predicted- and true yearly fatigue damage values.

Epochs	Training loss	Validation loss	MAE	MSE	Model fitting
50	0.0178	0.0155	0.0190	0.00400	Under fit
150	0.0149	0.0146	0.0190	0.00402	Under fit
250	0.0118	0.0112	0.0123	0.00160	Good fit
350	0.0082	0.0071	0.0079	0.00059	Good fit
450	0.0090	0.0095	0.0117	0.00119	Good fit
550	0.0061	0.0076	0.0072	0.00055	Good fit
650	0.0075	0.0079	0.0088	0.00048	Good fit
750	0.0047	0.0053	0.0052	0.00018	Good fit
850	0.0061	0.0061	0.0068	0.00025	Good fit
950	0.0051	0.0058	0.0055	0.00021	Good fit
1050	0.0062	0.0073	0.0097	0.00083	Good fit

To get an understanding of how accurate the predictions in Table 3.12 are, it is beneficial to visualize the predicted results and corresponding error. The most accurate result during the experiment was obtained with 750 epochs, and can be seen in Figure 3.17.



(a) Predictions vs true damage

(b) Predicted values minus true values

Figure 3.17: ML-model results for 750 epochs.

3.2.5 Experimenting with optimizer and activation function

In order to investigate the robustness of the ML-model, investigations regarding the choice of optimizer and activation function was performed. For these experiments, the amount of epochs was limited to 750, 850 and 950.

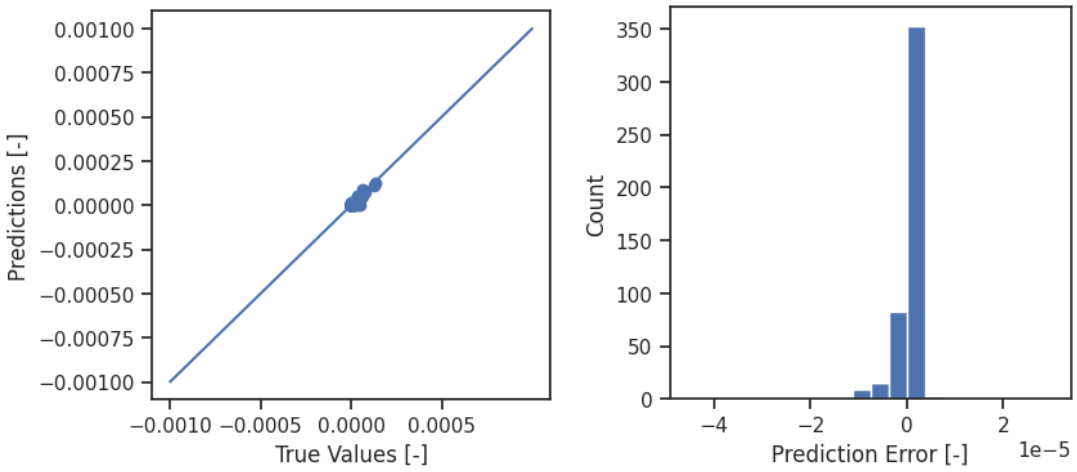
tanh as activation function

For these experiments, the same architecture as described in Table 3.11 was used with Adam as optimizer. The batch size was set to 32.

Table 3.13: Experimenting with tanh as activation function. MSE and MAE are calculated with respect to the predicted- and true yearly fatigue damage values.

Epochs	Training loss	Validation loss	MAE	MSE	Model fitting
750	0.0124	0.0115	0.0143	0.00222	Good fit
850	0.0080	0.0078	0.0103	0.00083	Good fit
950	0.0082	0.0085	0.0100	0.00061	Good fit

The most accurate result during the experiment was obtained with 950 epochs, and can be seen in Figure 3.18.



(a) Predictions vs true damage

(b) Predicted values minus true values

Figure 3.18: ML-model results for 950 epochs.

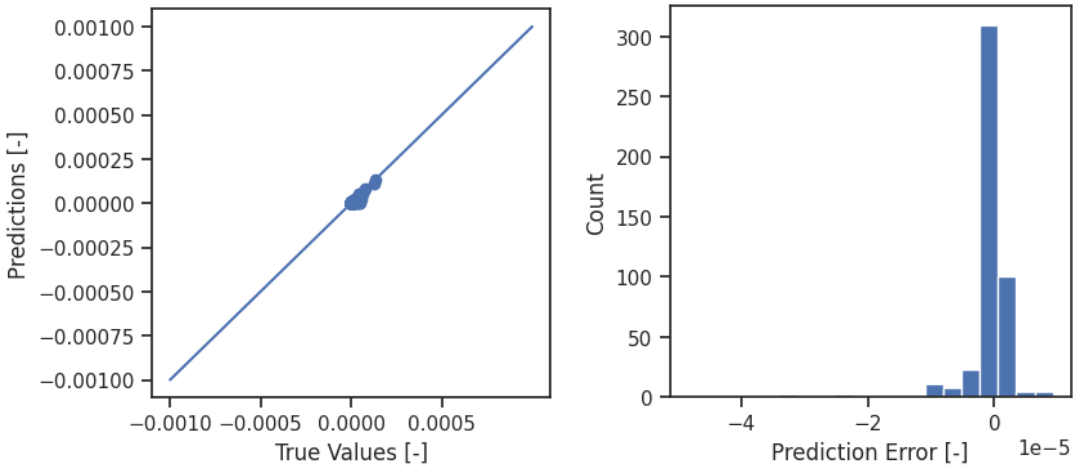
RMSProp as optimizer

For these experiments, the same architecture as described in Table 3.11 was used with ReLU as activation function. The batch size was set to 32.

Table 3.14: Experimenting with RMSProp as optimizer. MSE and MAE are calculated with respect to the predicted- and true yearly fatigue damage.

Epochs	Training loss	Validation loss	MAE	MSE	Model fitting
750	0.0065	0.0071	0.0074	0.00039	Good fit
850	0.0081	0.011	0.0101	0.00117	Good fit
950	0.0068	0.0086	0.0095	0.00069	Good fit

The most accurate result during the experiment was obtained with 950 epochs, and can be seen in Figure 3.19.



(a) Predictions vs true damage

(b) Predicted values minus true values

Figure 3.19: ML-model results for 950 epochs.

3.2.6 Experiments with input features

It was of interest to investigate how the removal of certain attributes affected the ML-model, considering both a redundant- and a minimalist perspective. The ML-model as described in Subsection 3.2.4 was used, with 950 epochs, but the shape of the input layer adapts according to the amount of removed input features. The obtained results are presented in Chapter 4.

Removal of Weibull distribution parameters

Over the course of a year it is likely that at least one of the sensors will be disconnected for some time. Therefore, it was advantageous to investigate whether or not the removal of one sensor decreased the accuracy. By considering the correlation matrix in Figure 3.16, it can be seen that the distribution parameters across the sensors are highly correlated, which indicates minor consequences.

Removal of wave direction

The on site wave direction has in this thesis been assumed to follow the wind direction obtained from Meteorologisk institutt. This is not necessarily true, especially since the exact measurement locations differs slightly. Hence, an investigation regarding the influence of the wave direction for the ML-model was made. Previously, the wind- and wave direction was vectorized together with H_s. As this no longer is a viable option, the wind direction was vectorized with the mean wind speed in order to provide the model some directional information.

3.2.7 Verification of fatigue damage scaling importance

To prove the importance of damage scaling it is meaningful to illustrate the performance of the ML-model when less preprocessing was done. The scaling of input features does not need more than the mathematical support provided in Subsection 2.3.8, but the scaling of the output feature does. Output scaling is uncommon and forces the model to predict wrong results. The main issue encountered when not scaling the fatigue damage was that the model, even with dropout layers, quickly got overfit before it was able to capture the desired trends. From a standpoint where conservatism is wanted, an overfitted model produced the worst outcome given an otherwise accurate model. Reason being that all the predictions fell close to the "CoG" of the simulated damages, similar to linear regression models. In other words, the model underestimated all substantial damages, and overestimated the non-critical values. As illustrated in Figure 3.15, the damage values calculated through Fatal was heavily skewed towards small values, relatively speaking, confirming that "CoG"-predictions provides non-conservative cumulative estimations over a sufficient time period.

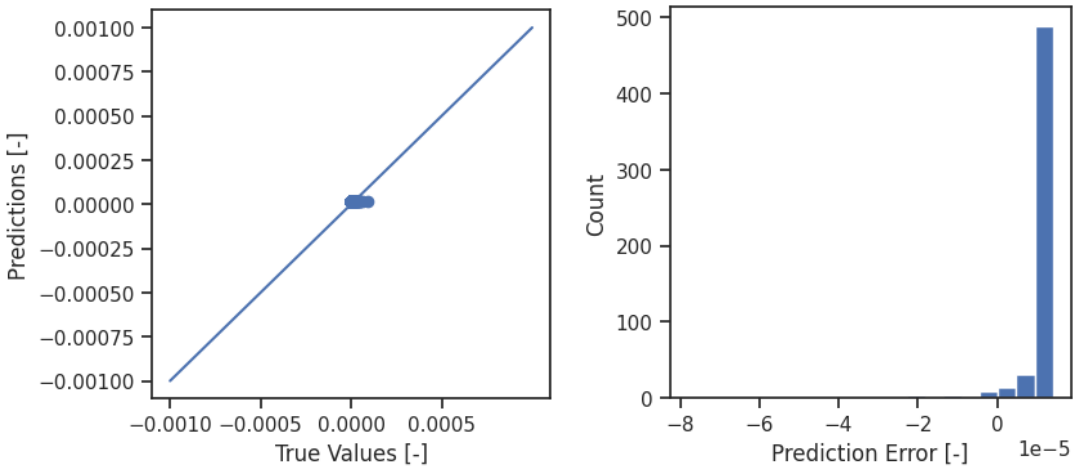
By increasing the amount of hidden layers, the model gets more prone to overfitting. Without scaling the fatigue damage, the initial model, described in Subsection 3.2.4, was quite sensitive to the number of epochs; less than 50 epochs always provided poor results. Hence, precautions had to be made. Primarily two remedies was possible, decreasing the amount of hidden layers or introduce dropout layers in between the existing layers. However, to be able to obtain low enough output, the amount of hidden layers had to be increased. Thus, dropout layers was a necessity.

Misman et al. (2019) received good results in their attempt to classify the age of Abalones (snails) based on features quantifying their size, among others. Their results improved by incorporating dropout layers in between the hidden layers to prevent overfitting. Generally, implementation of one or more dropout layers contributes with three main things: reduce overfitting, decrease computational time due to smaller architectures and improve generalization in NNs. The dropout layer may sound counter-intuitive at first, but is actually well reasoned. By temporarily removing some neurons, randomly for each epoch, the active neurons gets forced to balance their weaknesses and strengths to better encapsulate the desired trends. However, one downside of dropout layers is that the cost function, quantifying the average loss over the training set, no longer makes sense, since neurons gets randomly deactivated.

A representative result when fatigue damage scaling was not performed is illustrated in Figure 3.20. To counteract overfitting, dropout layers was implemented between all the hidden layers. The ML-model architecture able to provide "CoG"-predictions is seen in Table 3.15, with Adam as optimizer. Due to epoch sensitivity, the batch size was one.

Table 3.15: Layer characteristics of ML-model applied on unscaled fatigue damage.

Layer	Layer type	Shape	Activation function	Kernel initializer	Dropout rate
1	Dense	(,14)	ReLU	Variance scaling	-
2	Dense	(,32)	ReLU	Variance scaling	-
3	Dropout	(,32)	-	-	0.2
4	Dense	(,32)	ReLU	Variance scaling	-
5	Dropout	(,32)	-	-	0.2
6	Dense	(,32)	ReLU	Variance scaling	-
7	Dropout	(,32)	-	-	0.2
8	Dense	(,32)	ReLU	Variance scaling	-
9	Dense	(,1)	-	-	-



(a) Predictions vs true damage

(b) Predicted values minus true values

Figure 3.20: The best obtained predictions with unscaled fatigue damage. The number of epochs was set to 50 with MSE as loss function. The batch size was set to one.

3.2.8 Applying the model on real-time monitored data

As a final challenge, the ML-model was used to estimate the accumulated fatigue damage during January 2021. To do so, the monitored data had to be tabulated on the same format as the simulated data. The Python code January2021_disp.py in Appendix E.4 was used to generate the dataset, where the tabulated data can be found in Google Drive[8]. A total of 744 hourly combinations of environmental conditions equals to 37200 samples. Subsequently, samples with missing input was removed. For January 2021, only three sea states had to be removed, corresponding to 150 samples. Furthermore, to comply with the scaling of the training data, the input features needed to be scaled to the range [0,1] with the MinMaxScaler. The predicted output values needed to be corrected by multiplying with the maximum fatigue damage from the dataset used for model fitting and -testing. As the predicted output is yearly fatigue damage, each prediction had to be scaled down to the experienced one-hour duration. Subsequently, the accumulated damage during January 2021 for each individual connection was found by direct summation of the contributions for each hour. The Python code used to predict the fatigue damage through the ML-model is the same script in Google Colab as linked in Subsection 3.2.3.

[8]https://docs.google.com/spreadsheets/d/1hJBdczmNNAVhqVi3PRBdpXuD5ek32cME/edit?usp=sharing&ouid=110407881319023833523&rtpof=true&sd=true

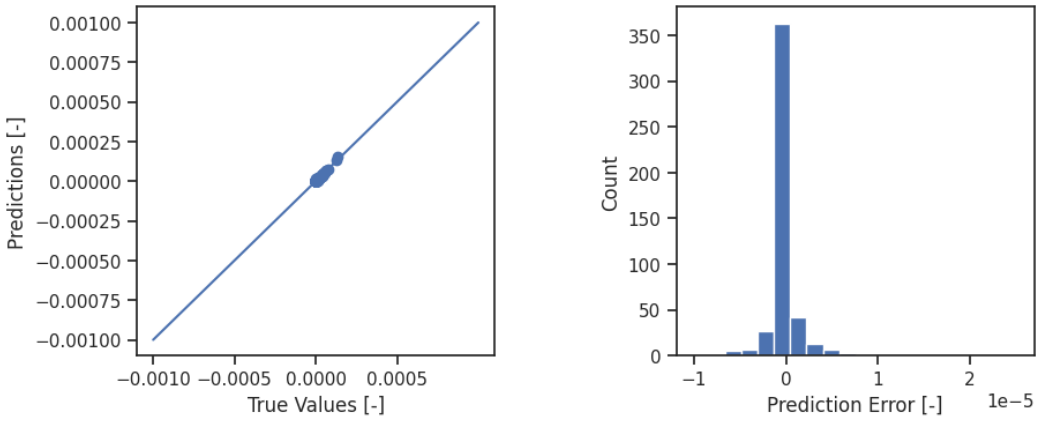
Chapter 4

Results and discussion

Although intermediate results and experiments with the ML-model are described throughout Chapter 3, a more thorough review of the final results will be presented in this chapter, along with a discussion of the results. The ML-model architecture shown in Table 3.11, trained over 950 epochs with Adam as optimizer and MAE as loss function was, among other highly qualified candidates, able to accurately replicate the yearly fatigue damage on the jacket structure calculated by use of USFOS. The results presented in this chapter are based on this model, where its simplicity is beneficial in terms of computational demand.

4.1 Predicted values from test set

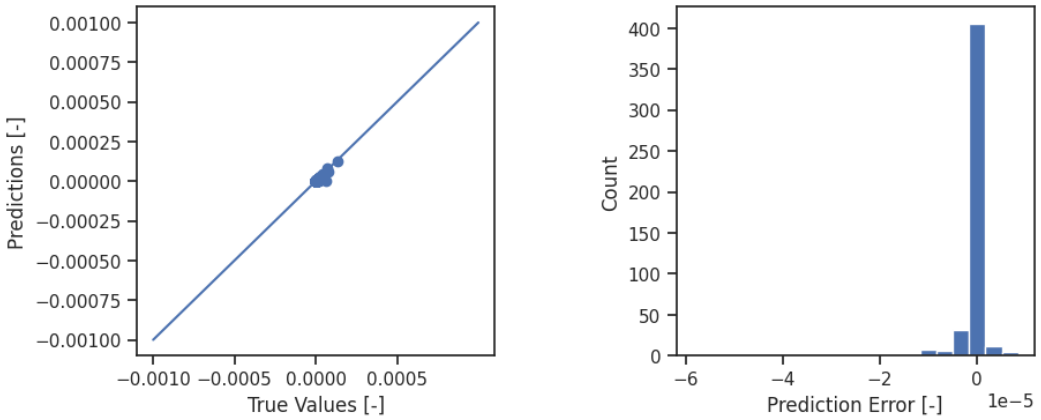
The results obtained from the test set are all presented with plots showing predicted yearly fatigue damage against the true USFOS-generated damage and corresponding errors. The line, $y = x$, illustrates identical predicted- and true values. The error was calculated as the predicted values minus the true values, which means that a positive error corresponds to overestimation, i.e. conservative estimations.



(a) Predictions vs true damage

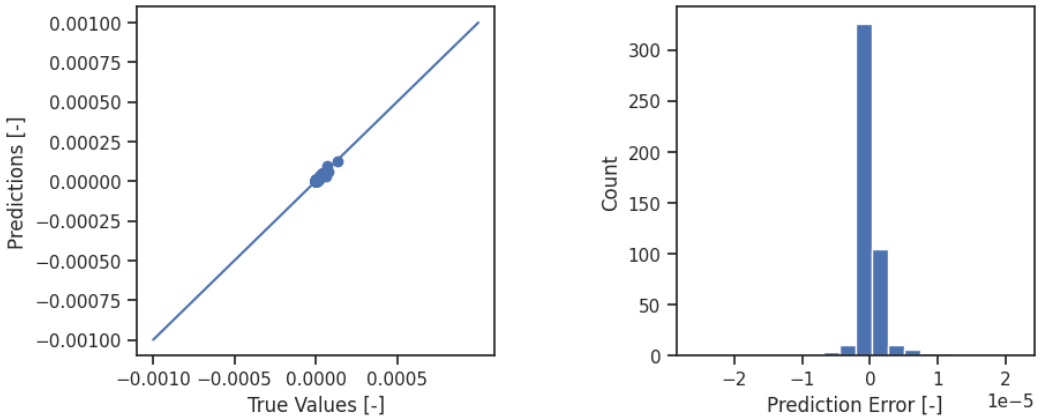
(b) Predicted values minus true values

Figure 4.1: ML-model performance with all input features.



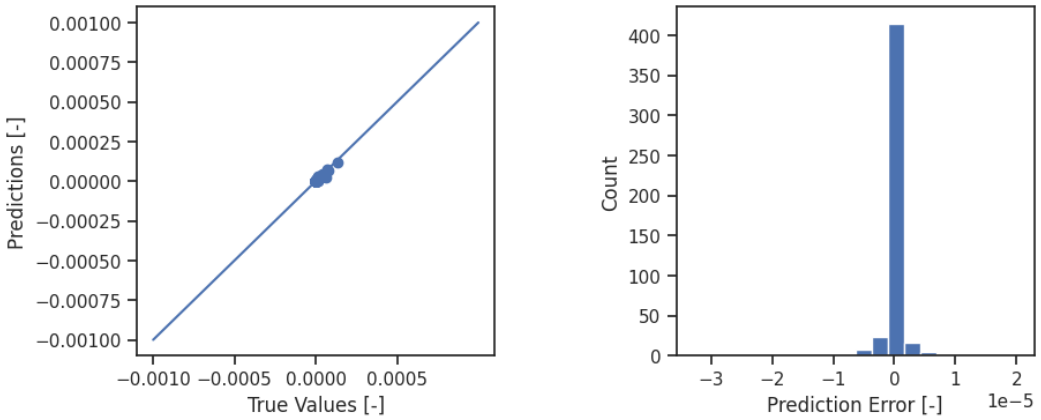
(a) Predictions vs true damage (b) Predicted values minus true values

Figure 4.2: ML-Model performance without Weibull parameters from sensor node 2.



(a) Predictions vs true damage (b) Predicted values minus true values

Figure 4.3: ML-Model performance without wave direction.



(a) Predictions vs true damage (b) Predicted values minus true values

Figure 4.4: ML-model performance without wave direction and Weibull parameters from sensor node 2.

The figures above shows that the predicted results from a FFNN corresponded very well with the simulated results from USFOS. The results were only slightly negatively affected by the removal of input features, which implies a redundancy. The redundancy of tabulated input features complies well with the correlation matrix in Figure 3.16, which shows that the Weibull parameters are highly correlated with each other. In addition, the vectorized wave direction is quite uncorrelated with the fatigue damage, which explains the accurate predictions when wave direction was excluded. The low correlation between the two latter features is probably a result of the 50 different connections. As the connections was spread across the entire jacket platform, a directional change increased the damage in some connections, but decreased the damage in others.

4.1.1 Predicted values for each sample

To get a better understanding of how the prediction error was distributed across the samples, Figure 4.5 illustrates the predicted values and true values for the case where wave direction was excluded. The reason behind highlighting this particular composition of input features is that it was used for the prediction of accumulated fatigue damage during January 2021 presented in Section 4.3. The equivalent plots for the other compositions of input features are presented in Appendix B.

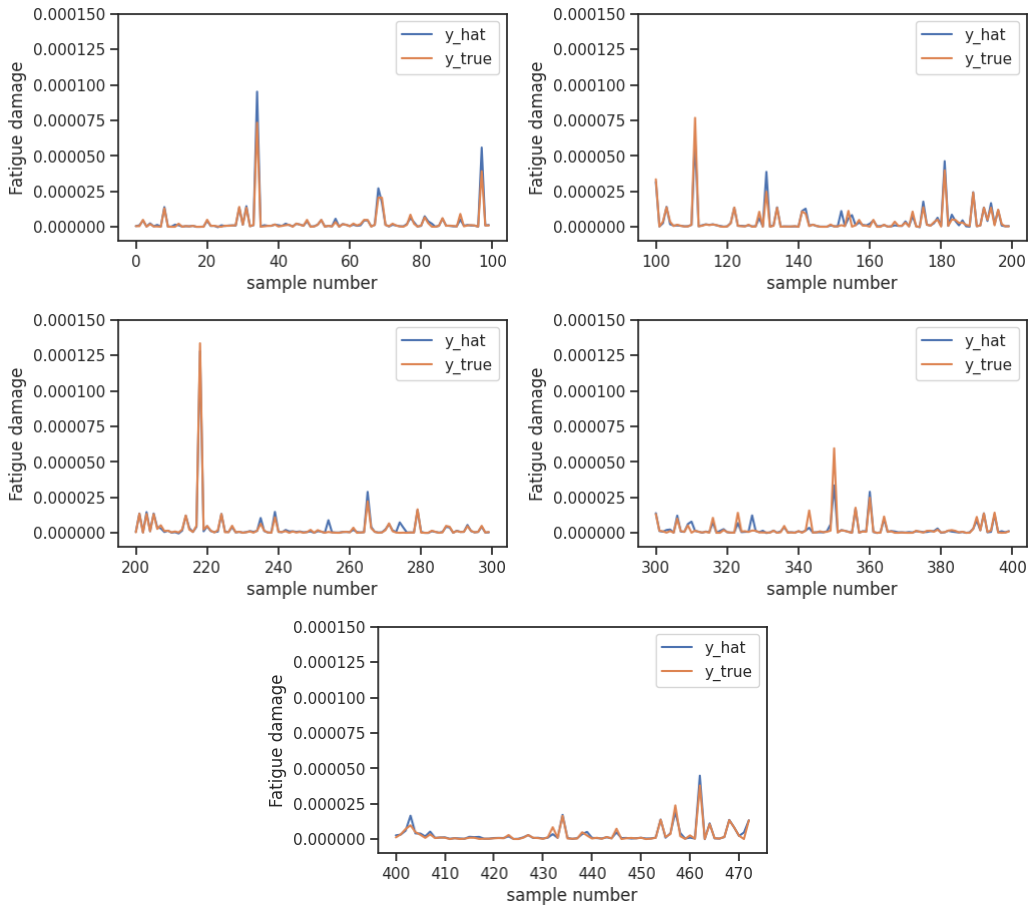


Figure 4.5: Predicted- and simulated yearly fatigue damage versus sample number on the test set, with wave direction excluded from the input features. y_true is the fatigue damage calculated through Fatal and y_hat is the predicted results from the ML-model.

It is important to keep in mind that some of the connections used in the model training and -fitting are more or less irrelevant from a fatigue failure perspective. In that sense, the model's ability to accurately predict the fatigue damage peaks is more critical, while still maintaining good accuracy on the moderate and low values. Figure 4.5 shows that the model slightly overestimated the extreme values, and that the error mainly was due to overprediction of small values. This result complies well with the initial desire of conservatism.

4.2 Loss from the validation data

The ML-model performance on the validation set given different input features can be quantified through the loss, which in was set to be MAE. The final losses are presented in Table 4.1, where it is important to note that the MAE was calculated with respect to the scaled fatigue damage, implying a higher loss than otherwise would have been.

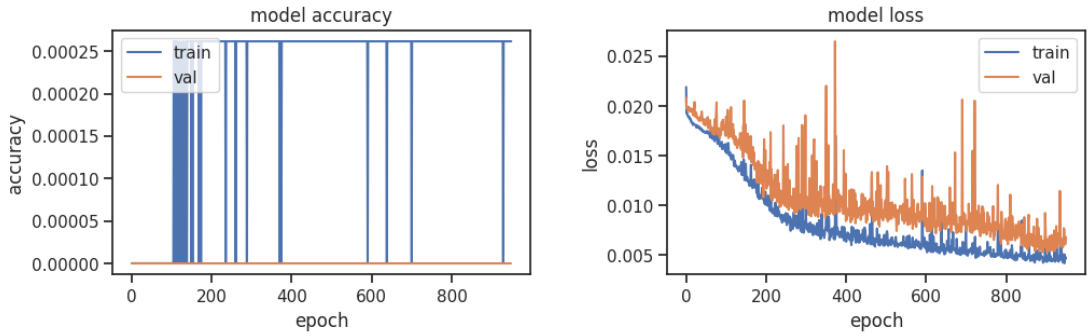
Table 4.1: Final ML-model loss with different input features from the validation data. The loss is calculated with respect to the scaled fatigue damage.

Description	MAE
All input features	0.0050
Excluding Weibull parameters	0.0077
Excluding wave direction	0.0055
Excluding Weibull parameters and wave direction	0.0081

The similarities in MAE across the predictions with different input features supports the statement about a redundant amount of input features discussed above.

4.2.1 Loss from training- and validation data

To illustrate the training stability, the model development related to accuracy and loss as a function of epochs, for the case where wave direction was excluded, is presented in Figure 4.6. The reason behind highlighting this particular composition of input features is that it was used for the prediction of accumulated fatigue damage during January 2021 presented in Section 4.3. In Figure 4.6b, a lot of noise in the training loss corresponds to an unstable learning rate, while a smooth graph indicates a stable learning rate. It is important to note that the MAE was calculated with respect to the scaled fatigue damage, implying a higher loss than otherwise would have been. As mentioned in Section 3.2, a binary evaluation of the accuracy of the model is of little use, and the exact accuracy values are therefore not so important. However, the accuracy should increase towards a stable value as the number of epochs increases. The equivalent plots for the other compositions of input features are shown in Appendix A.



(a) Model accuracy

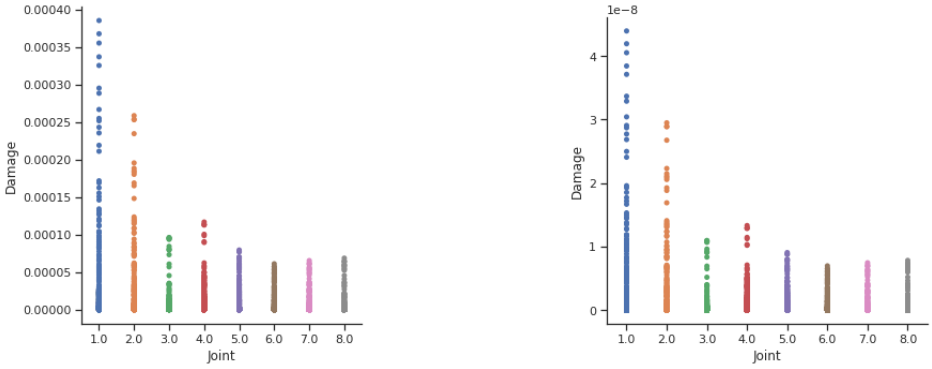
(b) Model loss

Figure 4.6: ML-model accuracy and -loss as a function of epochs, with wave direction excluded from the input features. The loss function is MAE, calculated with respect to the scaled yearly fatigue damage.

In Figure 4.6b, both the training- and validation loss decreases continuously and stabilizes somewhat around 900 epochs, indicating a good fit. The loss from the validation set is slightly higher than the loss from the training data. However, the deviation between the two is minimal, and parts of the difference may be due to a more complex validation- than training set. Further, it can be seen that the training loss is quite smooth, while the validation loss carries some more noise. This varied slightly for each performed fit, but is likely a consequence of that the validation loss is measured after each epoch, while the training loss is measured after each batch. By removing some input features, the loss generally increased slightly, with more noise, but was able to stabilize after a sufficient amount of epochs. This intuitively makes sense, since by removing input features, one also reduces the amount of available relationships for the ML-model. However, it is important to mention that the loss values are small, and the noise therefore seems more dramatic than it is.

4.3 Accumulated fatigue damage during January 2021

As a final result, the accumulated fatigue damage during January 2021 was estimated. The predicted damages for each sample can be seen in Figure 4.7 and the accumulated damage for each connection can be seen in Table 4.2.



(a) Yearly fatigue damage

(b) Hourly fatigue damage

Figure 4.7: Predicted fatigue damage for each joint during January 2021.

Table 4.2: Predicted accumulated fatigue damage for each connection during January 2021.

Connection	Fatigue damage	Connection	Fatigue damage
1	1.68e-6	26	9.13e-8
2	7.40e-7	27	4.55e-8
3	1.59e-7	28	2.51e-8
4	6.96e-8	29	1.38e-8
5	1.21e-7	30	1.38e-7
6	6.00e-7	31	1.33e-7
7	3.39e-7	32	1.34e-7
8	3.01e-7	33	1.16e-7
9	2.28e-7	34	4.65e-8
10	2.38e-7	35	2.00e-8
11	2.24e-7	36	6.26e-9
12	2.05e-7	37	1.54e-7
13	1.37e-7	38	1.37e-7
14	1.99e-7	39	1.25e-7
15	2.74e-7	40	8.80e-8
16	2.84e-7	41	3.79e-8
17	2.75e-7	42	2.21e-8
18	2.36e-7	43	1.59e-8
19	1.96e-7	44	1.21e-7
20	1.35e-7	45	7.72e-8
21	1.44e-7	46	5.36e-8
22	1.27e-7	47	3.08e-8
23	1.33e-7	48	1.75e-8
24	1.26e-7	49	4.41e-8
25	1.04e-7	50	1.32e-7

For the predicted accumulated fatigue damage during January 2021, the accuracy of the results are difficult to verify exactly, but some evaluations can be done. Several predictions were made, whereas slight variations in the predicted values occurred each time, while most trends remained intact. As can be seen in Figure 4.7, the predicted damage has a descending curve from joint 1 to joint 8. Even though the directional influence from the environmental actions is not directly comparable, it complies quite well with the tabulated, unscaled fatigue damage shown in Figure 3.15b.

Although the accumulated fatigue damage during January normally is non-representative for the yearly damage, comparisons can be made. The maximum predicted accumulated damage occurred in connection 1, and corresponds to a lifespan of 49603 years, given identical environmental conditions without any applied design safety factor. The corresponding minimum total lifetime of connection 1, calculated by Aker Solutions in their FLS-report for two different combinations of subsidence, was equal to 413 years with applied design safety factor. The predicted accumulated fatigue damage during January 2021 therefore seems unrealistically low. Since the predictions complies well with the results generated through USFOS, it indicates that the problems lie within the performed fatigue simulations. By recognizing that the simulated displacements was substantially larger than the monitored displacements, this result is quite surprising.

4.4 Uncertainties in the USFOS fatigue simulations

Its important to emphasize that the accuracy of the ML model in terms of assessing a real structure was inherently limited by the quality of the performed time domain fatigue analyses. Simplifications of the provided FEM had to be made, where it was important not to lose information which affected the dynamic behaviour of the structure. Nevertheless, the reader should be aware of some factors limiting the accuracy of the method applied to generate a dataset for ML in terms of real-life application:

- The bridge connections, and their influence in terms of stiffness, was never known.
- No information about current or marine growth affecting the dynamics of the structure was available.
- The topside was replaced by a point mass. The influence on the wave loads is negligible, while the influence from the wind loses some of its purpose when the topside is not physically present. In addition, wind gusts could not reasonably be applied in USFOS.
- Subsidence was neglected throughout the simulations. It is likely that this increased the deviation between the simulated and monitored displacements.
- Some of the smaller structural configurations in the jacket FEM was removed. They did presumably not affect the structural properties, but the load contribution due to waves could not be represented.
- 15 minute simulations may not have been sufficient to obtain a stable standard deviation of the applied spectra.
- As stated in Subsection 3.1.8, Fatal was set to skip the initial six seconds of the simulations to avoid extreme half-cycles. This was perhaps a bit short of what was actually needed.
- The tripods supporting the nodes representing the sensors on the actual structure was defined to have a substantial axial stiffness, ideally to limit the nodal translation to comply with the remainder of the topside. Some unwanted translational- and bending motion may have occurred, but the influence is considered to be of minor importance, as the magnitude is not included in addition to the tabulated Weibull distribution parameters. The potential unwanted motions could have been investigated by comparing the translations of the sensor nodes with the tripod clamps.
- Fatal is not a publicly available USFOS-module, and the accuracy of the fatigue calculations used as ground truth should therefore be shown some scepticism. Inaccuracy in the fatigue calculations is especially plausible in joints where the parametric equations in Subsection 2.2.2, used to calculate the SCFs, not necessarily are correct.

4.5 Dataset evaluation

Since a substantial part of this thesis consisted of creating a dataset suited for ML, it is important to question whether or not it was structured in the best possible way. For cases where the waves came from a pure x- or y-direction, the displacement pattern normal to the wave direction was poorly described by the 2-parameter Weibull distribution. An intricate dilemma appears, since the displacement pattern in the same direction as the applied waves was so incredibly well described by the Weibull distribution. This discovery was omitted throughout the simulations, but may influence the results when evaluating on-site monitored data. Such samples may be important to remove.

The dataset was customized to facilitate that the machine learning model should predict the governing fatigue damage in 50 connections distributed over eight joints. This implies that each simulation corresponded to 50 samples, where each sample, except for the fatigue damage and indexation features, had to be repeated 50 times. Even though the training- and test set was split in such a way that the influence of quite similar data should be minimal, it is possible that the results became somewhat biased by that the model became disproportionately confident on "known" data. This indicates that a model for each joint or connection could have been advantageous.

Even though a lot of simulations was performed, only a limited amount of sea states was selected. More diversity could perhaps have improved the results. Simultaneously as it was important to limit the sea states to be within the validity range of the JONSWAP spectrum, a wider range of sea states could have provided a better foundation for the data scaling discussed in the subsection below.

4.5.1 Data scaling

The input features in the training- and validation set was scaled to the range [0,1], which indeed limits the ML-model capabilities to this range. The same scaling procedure was applied on the test set and the set used to predict the accumulated fatigue damage during January 2021. Although the results on the test set were close to ground truth, an alternative scaling procedure could have been to scale the other sets by utilizing the minimum- and maximum values from the training set. This way, the data gets scaled relative to already known values, and the ML-model may become more adapted to what can be considered as abnormal input values.

Chapter 5

Conclusion and recommendations for further work

5.1 Conclusion

The objectives of this thesis was to develop a dataset suited for ML, and subsequently build- and train a NN to replicate fatigue damage in critical joints on a jacket structure, calculated through time domain FEA. By combining monitored data with machine learning, a trained NN may predict the real-time fatigue damage, and contribute to reduce the long-term usage of FEA.

The dataset used for training an testing was constructed though a multi-step process. Initially, a GeniE FEM was converted to a USFOS FEM and simplified in order to be suited for dynamic time domain fatigue simulations. Subsequently, appropriate combinations of environmental loads was calculated through statistical evaluations of monitored data and systematically applied the FEM. The resulting fatigue damage was tabulated with properties of the environmental actions and 2-parameter Weibull distribution parameters fitted to the displacement peaks for the two nodes representing the sensors on the real platform.

A quite simple FFNN was developed, and provided promising indications that an ordinary FFNN indeed is capable of quite accurate fatigue damage predictions of critical joints in an offshore jacket structure, when trained and tested on a dataset constructed by use of a FEM subjected to irregular waves and wind from different headings. The robustness was confirmed by evaluating predictions made with different types of activation functions and optimizers, in addition to removal of certain input features. Furthermore, slightly conservative predictions of the highest damage peaks was possible to achieve, which complies well with a desire of conservative estimations. For the applied FFNN, data scaling and removal of outliers was a prerequisite for obtaining accurate results. This included scaling of the output feature to increase the dispersion.

The limitations seem to lie within the relationship between the performed FEA and the real structure which is to be assessed. An accurate ML-model does not necessarily imply accurate predictions of the fatigue damage on the real structure, and is essential for achieving reliable results in real-life application.

5.2 Recommendations for further work

Considering the quite large field of study in this thesis, only one possible solution procedure has been investigated. Therefore, several recommendations for further work among all branches explained in the method is proposed. In relation to the modelling and simulations, the following steps are recommended:

- Expand the system to two or three jackets connected by bridges.
- Establish a generally more exact representation of the platform(s).
- Perform quality checks on the calculated fatigue damage, to ensure that the ML-model encapsulates the desired real-life trends.

For the tabulation of data to be applied on the ML-model, the following suggestions are made:

- Establish other combinations of input features, possibly with varying degrees of statistical evaluations.
- Tabulate unique data for each joint to avoid replicated input features across the samples.
- Expand the generated dataset used for training, validation and testing to be more diverse. This is essential if several joint-specific ML-models are created.

And lastly, the following steps are proposed for ML:

- Experiment with different types of scaling processes. It is generally not desirable to scale the output feature.
- Experiment with different types of model architectures, which becomes especially relevant in combination with different types of tabulation methods.
- A FFNN could possibly be used in combination with a recursive NN, where the recursive NN uses input from the FFNN to predict fatigue damage based on previous history. Recursive NNs are commonly applied in stock predictions.

Bibliography

Ahmadi, Hamid and Lotfollahi-Yaghin, Mohammad Ali (2013) Effect of SCFs on S–N based fatigue reliability of multi-planar tubular DKT-joints of offshore jacket-type structures, *Ships and Offshore Structures*, 8(1), pp. 55–72. DOI: 10.1080/17445302.2011.627750. eprint: https://doi.org/10.1080/17445302.2011.627750. Available at: https://doi.org/10.1080/17445302.2011.627750.

Amdahl, Jørgen (2009) *Marin Teknikk 2 - Del 2: Knekking*. NTNU - Department of Marine Technology.

Amzallag, C., Gerey, J.P., Robert, J.L. and Bahuaud, J. (1994) Standardization of the rainflow counting method for fatigue analysis, *International Journal of Fatigue*, 16(4), pp. 287–293. ISSN: 0142-1123. DOI: https://doi.org/10.1016/0142-1123(94)90343-3. Available at: https://www.sciencedirect.com/science/article/pii/0142112394903433.

Berge, Stig and Ås, Sigmund Kyrre (2017) *Fatigue and Fracture Design of Marine Structures*. 3rd ed. NTNU.

Chaudhury, G.K. and Dover, W.D. (1985) Fatigue analysis of offshore platforms subject to sea wave loadings, *International Journal of Fatigue*, 7(1), pp. 13–19. ISSN: 0142-1123. DOI: https://doi.org/10.1016/0142-1123(85)90003-9. Available at: https://www.sciencedirect.com/science/article/pii/0142112385900039.

community, Tensorflow (2021) *The sequential model*. Available at: https://www.tensorflow.org/guide/keras/sequential_model (Accessed: 7th Dec. 2021).

De Marchi, Leonardo and Mitchell, Laura (2019) *Hands-On Neural Networks: Learn how to build and train your first neural network model using Python*. Packt Publishing Ltd.

DNV (2014) *DNV-RP-C203: Fatigue Design of Offshore Steel Structures*.

Dong, Wenbin, Moan, Torgeir and Gao, Zhen (2012) Fatigue reliability analysis of the jacket support structure for offshore wind turbine considering the effect of corrosion and inspection, *Reliability Engineering & System Safety*, 106, pp. 11–27. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2012.06.011. Available at: https://www.sciencedirect.com/science/article/pii/S0951832012001160.

Figueiredo, Eloi, Park, Gyuhae, Farrar, Charles R, Worden, Keith and Figueiras, Joaquim (2011) Machine learning algorithms for damage detection under operational and envir-

onmental variability, *Structural Health Monitoring*, 10(6), pp. 559–572. DOI: 10.1177/1475921710388971. eprint: https://doi.org/10.1177/1475921710388971. Available at: https://doi.org/10.1177/1475921710388971.

Goodfellow, Ian, Bengio, Yoshua and Courville, Aaron (2016) *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Guo, Yanming, Liu, Yu, Oerlemans, Ard, Lao, Songyang, Wu, Song and Lew, Michael S. (2016) Deep learning for visual understanding: A review, *Neurocomputing*, 187. Recent Developments on Deep Big Vision, pp. 27–48. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2015.09.116. Available at: https://www.sciencedirect.com/science/article/pii/S0925231215017634.

Hansen, Jannick Balleby, Brincker, Rune, Mathiasen, Søren, Jørgensen, Anders, Knudsen, Mads Bøgevad and Tygesen, UT (2011) Combining GPS and integrated sensor signals, *Proc. International Operational Modal Analysis Conference (IOMAC 2011)*. Vol. 1.

Haver, Sverre (2019) *Metocean modelling and prediction of extremes*. Haver & havet, University in Stavanger, ntnu.

Holzinger, Andreas, Kieseberg, Peter, Weippl, Edgar and Tjoa, A. Min (2018) Current Advances, Trends and Challenges of Machine Learning and Knowledge Extraction: From Machine Learning to Explainable AI, *Machine Learning and Knowledge Extraction*. Ed. by Andreas Holzinger, Peter Kieseberg, A Min Tjoa and Edgar Weippl. Cham: Springer International Publishing, pp. 1–8. ISBN: 978-3-319-99740-7.

Jia, Junbo, Ellefsen, R. and Holmas, Tore (Jan. 2008) An efficient nonlinear dynamic approach for calculating wave induced fatigue damage of offshore structures and its industrial applications for lifetime extension, *Proceedings of the International Offshore and Polar Engineering Conference*, pp. 310–317.

Kingma, Diederik P. and Ba, Jimmy (2014) *Adam: A Method for Stochastic Optimization*. DOI: 10.48550/ARXIV.1412.6980. Available at: https://arxiv.org/abs/1412.6980.

Langen, Ivar (1999) *Dynamisk analyse av konstruksjoner*. nob. S.l.

LeCun, Yann, Bengio, Yoshua and Hinton, Jeffrey (2015) Deep learning, Nature 521, 436-444. DOI: 10.1038/nature14539. Available at: https://doi.org/10.1038/nature14539.

Liu, Bing (2011) 'Unsupervised Learning'. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 133–169. ISBN: 978-3-642-19460-3. DOI: 10.1007/978-3-642-19460-3_4. Available at: https://doi.org/10.1007/978-3-642-19460-3_4.

Luna, Julio, Falkenberg, Ole, Gros, Sébastien and Schild, Axel (2020) Wind turbine fatigue reduction based on economic-tracking NMPC with direct ANN fatigue estimation, *Renewable Energy*, 147, pp. 1632–1641. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2019.09.092. Available at: https://www.sciencedirect.com/science/article/pii/S0960148119314235.

Mendes, Paulo, Correia, José A. F. O., Mourão, António, Pereira, Rafael, Fantuzzi, Nicholas, Jesus, Abílio De and Calçada, Rui (2021) Fatigue Assessments of a Jacket-Type

Offshore Structure Based on Static and Dynamic Analyses, *Practice Periodical on Structural Design and Construction*, 26(1), p. 04020054. DOI: 10.1061/(ASCE)SC.1943-5576.0000533. eprint: https://ascelibrary.org/doi/pdf/10.1061/\%28ASCE\%29SC.1943-5576.0000533. Available at: %7Bhttps://ascelibrary.org/doi/abs/10.1061/%5C%28ASCE%5C%29SC.1943-5576.0000533%7D.

Meteorologisk institutt (2022). Available at: https://frost.met.no/index.html (Accessed: 5th Feb. 2022).

Misman, Muhammad, A Samah, Azurah, Aziz, Nur, Majid, Hairudin, Ali Shah, Zuraini, Hashim, Haslina and Harun, Muhamad Farhin (Sept. 2019) Prediction of Abalone Age Using Regression-Based Neural Network, pp. 23–28. DOI: 10.1109/AiDAS47888.2019.8970983.

Moan, Torgeir (2003) *TMR 4190 - Finite Element Modelling and Analyses of Marine Structures*. NTNU - Department of Marine Technology.

— (2005) Reliability-based management of inspection, maintenance and repair of offshore structures, *Structure and Infrastructure Engineering*, 1(1), pp. 33–62. DOI: 10.1080/15732470412331289314. eprint: https://doi.org/10.1080/15732470412331289314. Available at: https://doi.org/10.1080/15732470412331289314.

Musallam, Mahera and Johnson, C. Mark (2012) An Efficient Implementation of the Rainflow Counting Algorithm for Life Consumption Estimation, *IEEE Transactions on Reliability*, 61(4), pp. 978–986. DOI: 10.1109/TR.2012.2221040.

Myrhaug, Dag (2019) *TMR 4182 Marine dynamics*. Department of Marine Technology, Faculty of Engineering Science and Technology, NTNU.

N'Diaye, A., Hariri, S., Pluvinage, G. and Azari, Z. (2007) Stress concentration factor analysis for notched welded tubular T-joints, *International Journal of Fatigue*, 29(8), pp. 1554–1570. ISSN: 0142-1123. DOI: https://doi.org/10.1016/j.ijfatigue.2006.10.030. Available at: https://www.sciencedirect.com/science/article/pii/S0142112306003161.

Newland, D.E (2005) *An introduction to Random Vibrations, Spectral & Wavelet Analysis*. 3rd ed. Dover Publications, Inc, Mineola, New York.

NORSOK, Standard N-003 (2017) Actions and action effects, *Standards Norway*.

Pettersen, B. (2007) *Marin Teknikk 3 - Hydrodynamikk*. Akademika Forlag.

Potvin, A.B., Kuang, J.G., Leick, R.D. and Kahlich, J.L. (Aug. 1977) Stress Concentration in Tubular Joints, *Society of Petroleum Engineers Journal*, 17(04), pp. 287–299. ISSN: 0197-7520. DOI: 10.2118/5472-PA. eprint: https://onepetro.org/spejournal/article-pdf/17/04/287/2158740/spe-5472-pa.pdf. Available at: https://doi.org/10.2118/5472-PA.

Purkait, Niloy (2019) *Hands-On Neural Networks with Keras: Design and create neural networks using deep learning and artificial intelligence principles*. Packt Publishing Ltd.

Roy, Saikat, Das, Nibaran, Kundu, Mahantapas and Nasipuri, Mita (2017) Handwritten isolated Bangla compound character recognition: A new benchmark using a novel deep learning approach, eng. *Pattern recognition letters*, 90, pp. 15–21. ISSN: 0167-8655.

Shabakhty, Naser and Khansari, Arash (Mar. 2019) Fatigue Analysis of a Jacket Structure to Linear and Weakly Nonlinear Random Waves, *Journal of Offshore Mechanics and Arctic Engineering*, 141(6). 061602. ISSN: 0892-7219. DOI: 10.1115/1.4042946. eprint: https://asmedigitalcollection.asme.org/offshoremechanics/article-pdf/141/6/061602/6403725/omae_141_6_061602.pdf. Available at: https://doi.org/10.1115/1.4042946.

Sharma, Sagar, Sharma, Simone and Athaiya, Anidhya (2017) Activation functions in neural networks, *towards data science*, 6(12), pp. 310–316.

Søreide, Tore H., Amdahl, Jørgen, Eberg, Ernst, Holmås, Tore and Hellan, Øyvind (1994) *USFOS - A Computer Program for Progressive Collapse Analysis of Steel Offshore Structures. Theory Manual.* Available at: https://www.usfos.no/manuals/usfos/theory/documents/Usfos_Theory_Manual.pdf (Accessed: 15th Nov. 2021).

Stacey, A and Sharp, J V (Dec. 1997) *Fatigue damage in offshore structures - causes, detection and repair.*

Tang, Yougang, Qing, Zhaoxi, Zhu, Longhuan and Zhang, Ruoyu (2015) Study on the structural monitoring and early warning conditions of aging jacket platforms, *Ocean Engineering*, 101, pp. 152–160. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2015.04.011. Available at: https://www.sciencedirect.com/science/article/pii/S0029801815000864.

USFOS (2010) *Hydrodynamics - Theory Description of use Verification.* Available at: https://www.usfos.no/manuals/usfos/theory/documents/Usfos_Hydrodynamics.pdf (Accessed: 15th Nov. 2021).

Vieira, Sandra, Pinaya, Walter H.L. and Mechelli, Andrea (2017) Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications, *Neuroscience & Biobehavioral Reviews*, 74, pp. 58–75. ISSN: 0149-7634. DOI: https://doi.org/10.1016/j.neubiorev.2017.01.002. Available at: https://www.sciencedirect.com/science/article/pii/S0149763416305176.

Worden, Keith, Farrar, Charles R, Manson, Graeme and Park, Gyuhae (2007) The fundamental axioms of structural health monitoring, eng. *Proceedings of the Royal Society. A, Mathematical, physical, and engineering sciences*, 463(2082), pp. 1639–1664. ISSN: 1364-5021.

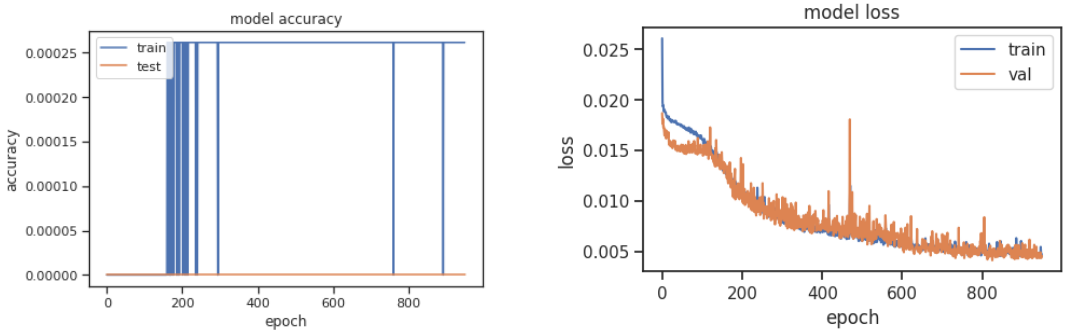
Yan, Wangchen, Deng, Lu, Zhang, Feng, Li, Tiange and Li, Shaofan (2019) Probabilistic machine learning approach to bridge fatigue failure analysis due to vehicular overloading, *Engineering Structures*, 193, pp. 91–99. ISSN: 0141-0296. DOI: https://doi.org/10.1016/j.engstruct.2019.05.028. Available at: https://www.sciencedirect.com/science/article/pii/S0141029618334345.

Mean Stress Effect on Fatigue of Welded Joint in FPSOs (June 2006). Vol. Volume 3: Safety and Reliability; Materials Technology; Douglas Faulkner Symposium on Reliability and Ultimate Strength of Marine Structures. International Conference on Off-

shore Mechanics and Arctic Engineering, pp. 403–412. DOI: 10.1115/OMAE2006-92056. eprint: https://asmedigitalcollection.asme.org/OMAE/proceedings-pdf/OMAE2006/47489/403/4532365/403_1.pdf. Available at: https://doi.org/10.1115/OMAE2006-92056.

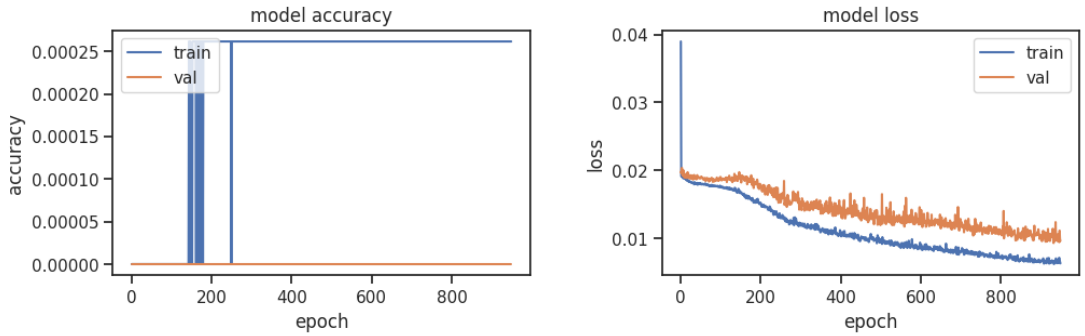
Appendix A

ML-model accuracy and -loss



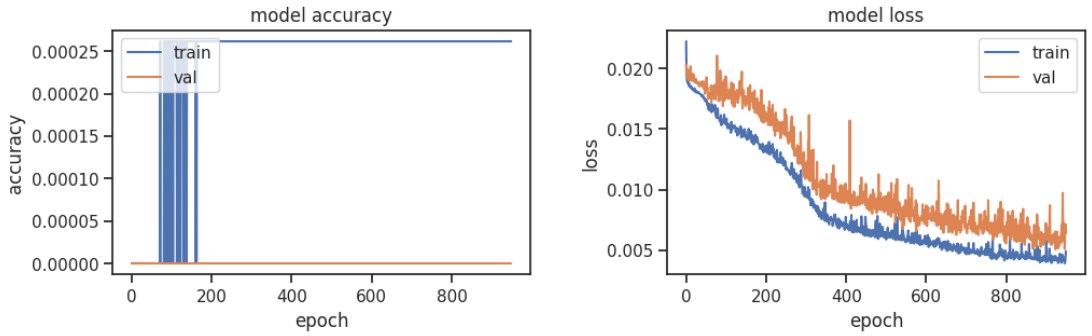
(a) Model accuracy (b) Model loss

Figure A.1: ML-model accuracy and -loss with all input features.



(a) Model accuracy (b) Model loss

Figure A.2: ML-model accuracy and -loss without Weibull parameters from sensor node 2.



(a) Model accuracy

(b) Model loss

Figure A.3: ML-model accuracy and -loss without Weibull parameters from sensor node 2 and wave direction.

Appendix B

ML-model predictions for each sample

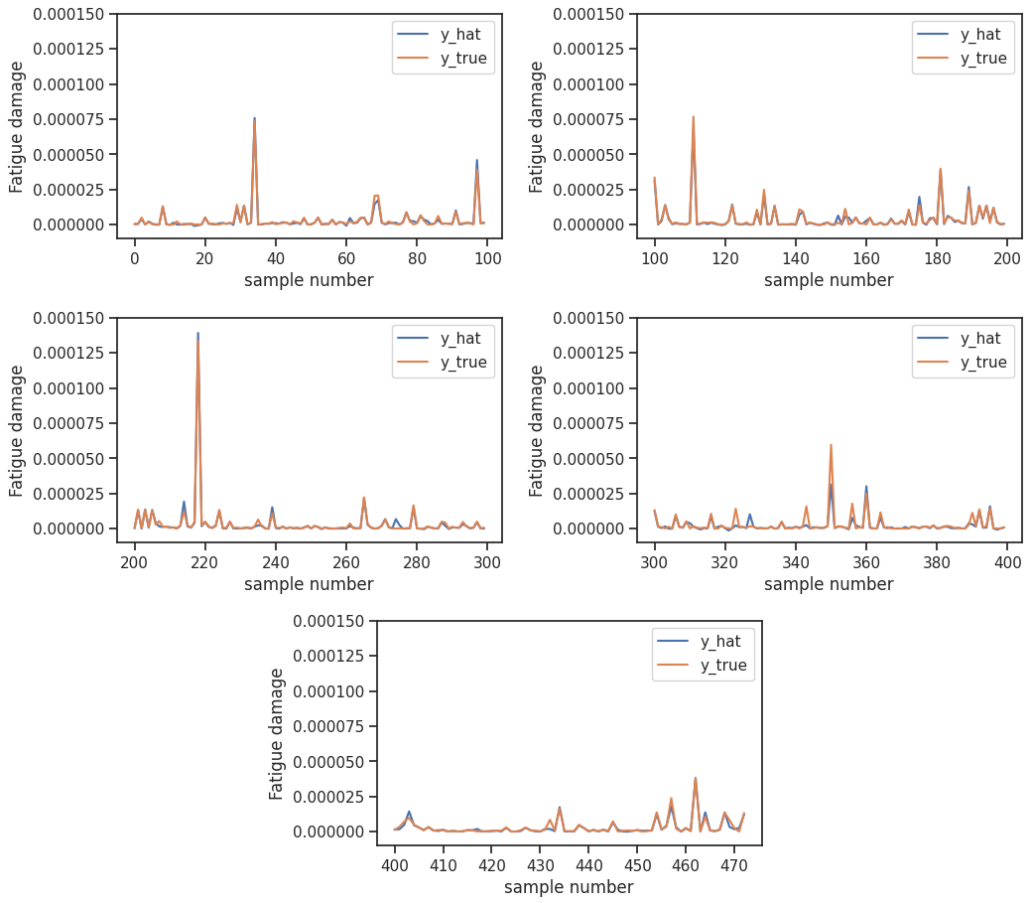


Figure B.1: Predicted fatigue damage values versus sample number on the test set, with all input features. y_true is the fatigue damage calculated in Fatal and y_hat is the predicted results from the ML-model.

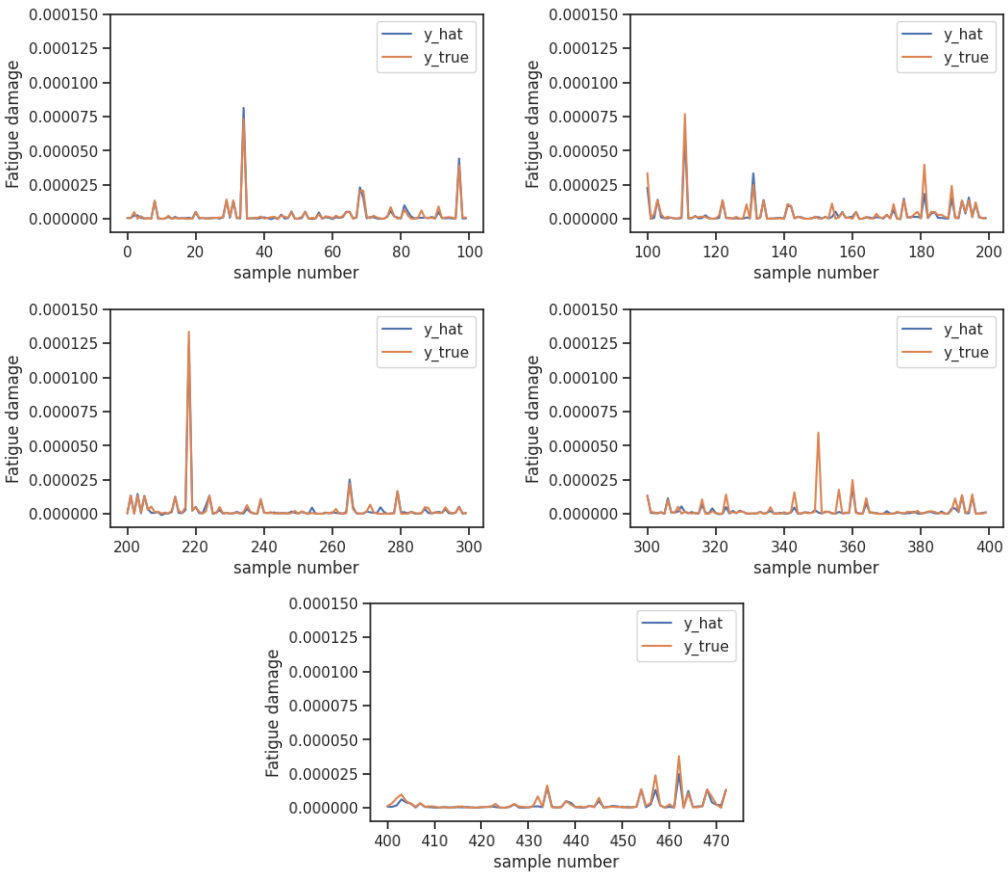


Figure B.2: Predicted fatigue damage values versus sample number on the test set, without Weibull parameters from sensor node 2. y_true is the fatigue damage calculated in Fatal and y_hat is the predicted results from the ML-model.

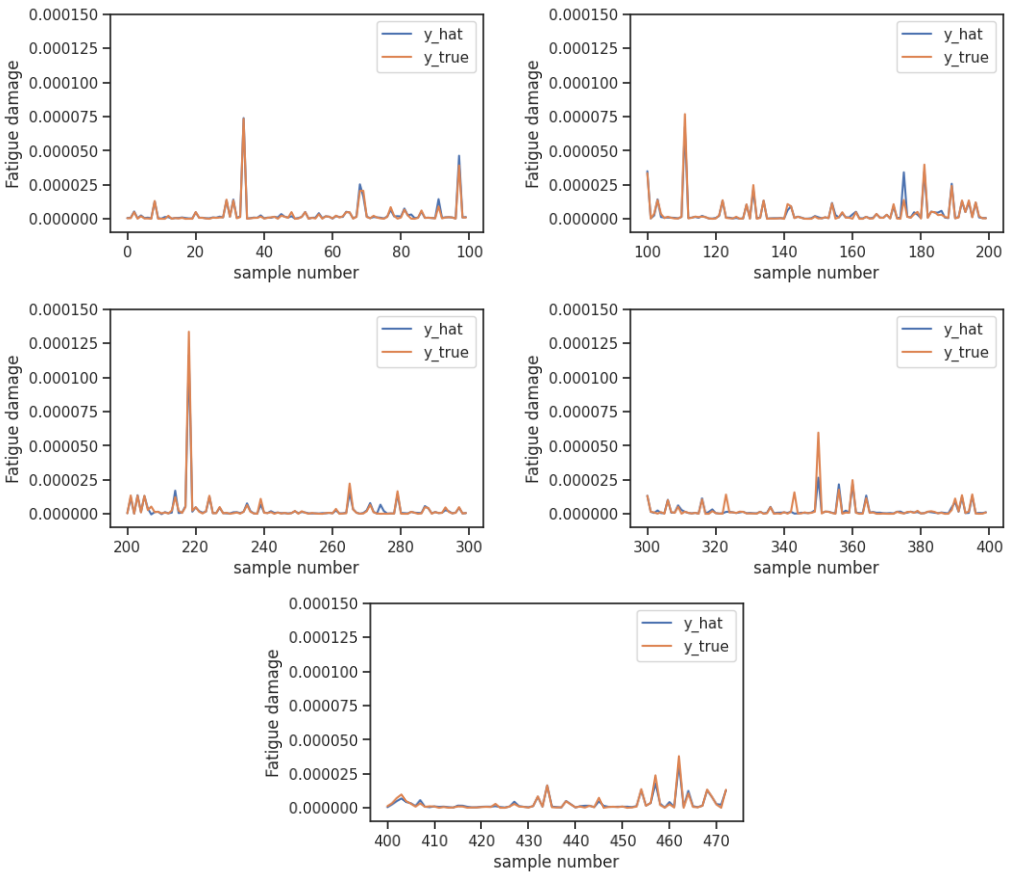


Figure B.3: Predicted fatigue damage values versus sample number on the test set, without Weibull parameters from sensor node 2 or wave direction. y_true is the fatigue damage calculated in Fatal and y_hat is the predicted results from the ML-model.

Appendix C

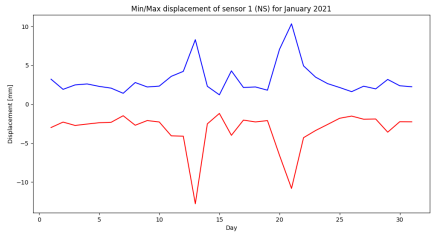
Scatter table

Hs[m] \ Tp[s]	[7-8)	[8-9)	[9-10)	[10-11)	[11-12)	[12-13)	[13-14)	[14-15)	[15-16)	[16-17)	[17-18)	[18-19)
[0-0.5)	0	0	0	0	0	0	0	0	0	0	0	0
[0.5-1)	131	1377	1167	511	297	194	75	39	7	1	0	0
[1-1.5)	4	555	1471	867	417	172	35	24	0	2	5	0
[1.5-2)	0	100	1371	896	336	155	54	19	3	0	0	0
[2-2.5)	0	2	630	1095	297	82	36	25	2	0	0	1
[2.5-3)	0	0	56	918	287	101	33	22	3	0	0	0
[3-3.5)	0	0	0	361	409	96	16	5	1	0	0	0
[3.5-4)	0	0	0	41	352	87	16	3	0	0	0	0
[4-4.5)	0	0	0	3	203	86	22	4	3	0	0	0
[4.5-5)	0	0	0	0	45	85	17	1	2	0	0	0
[5-5.5)	0	0	0	0	2	87	13	5	0	0	0	0
[5.5-6)	0	0	0	0	0	43	24	3	0	0	0	0
[6-6.5)	0	0	0	0	0	11	27	6	3	0	0	0
[6.5-7)	0	0	0	0	0	0	14	6	12	4	0	0
[7-7.5)	0	0	0	0	0	0	7	2	8	7	0	0
[7.5-8)	0	0	0	0	0	0	4	4	0	3	0	0
[8-8.5)	0	0	0	0	0	0	2	0	0	0	0	0

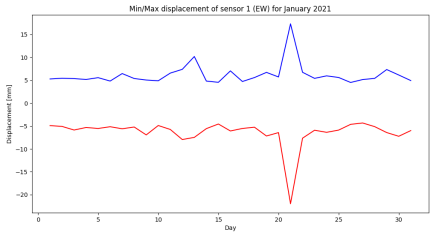
Figure C.1: Scatter table showing the amount of individual hourly sea states, based on the wave data provided by Aker BP.

Appendix D

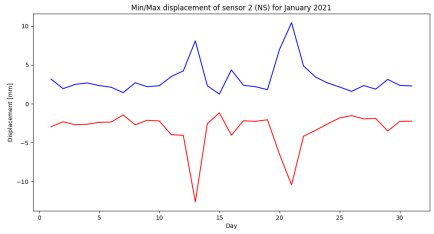
Displacements during January 2021



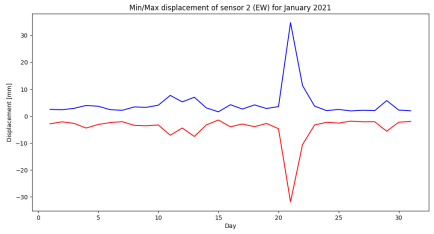
Sensor 1, $\hat{NS} = -\hat{x}$



Sensor 1, $\hat{EW} = -\hat{y}$



Sensor 2, $\hat{NS} = -\hat{x}$



Sensor 2, $\hat{EW} = -\hat{y}$

Figure D.1: Minimum- and maximum monitored displacements for each day during January 2021.

Appendix E

Python codes

E.1 Code to automate USFOS simulations, *RunUsfos.py*

```
"""
Goal: Automate USFOS-simulations
@Author: Gjermund Smedsland
"""
import os
import pandas as pd
from shutil import copyfile
import numpy as np
import multiprocessing as mul
from functools import partial
import rainflow
from statistics import mean, stdev
import xlwings as xw
import time

def run_usfos(run_folder, head_name, model_name, load_name, file_loc):
    cwd = os.getcwd()
    os.chdir(run_folder)

    #creating input file defining input for the usfos run
    inp_usfos = 'parameters.txt'
    f = open(inp_usfos,'w')
    f.write(head_name + '\n')
    f.write(model_name + '\n')
    f.write(load_name + '\n')
    f.write('res\n')
    f.close()

    #creating input files for dynres
    inp_dynres = ['parametersdynres_acc1.txt', 'parametersdynres_acc2.txt',
                  'parametersdynres_acc3.txt', 'parametersdynres_acc4.txt',
                  'parametersdynres_disp1.txt', 'parametersdynres_disp2.txt',
                  'parametersdynres_disp3.txt', 'parametersdynres_disp4.txt',
                  'parametersdynres_waveelev.txt']
    output_files = ['ACC1.txt', 'ACC2.txt', 'ACC3.txt', 'ACC4.txt', ...
        'DISP1.txt',
                    'DISP2.txt', 'DISP3.txt', 'DISP4.txt', 'WaveElev.txt']
    for idx, name in enumerate(inp_dynres):
        f = open(name,'w')
        f.write('res\n')
```

```
40          f.write('2\n')
41          f.write('1\n')
42          f.write('0\n')
43          f.write(str(idx + 175) + '\n')
44          f.write(output_files[idx] + '\n')
45          f.write('0\n')
46          f.write('0\n')
47          f.close()
48
49      # Creating input files for FATAL
50      inp_fatal = 'fatal_parameters.txt'
51      f_fatal = open(inp_fatal, 'w')
52      f_fatal.write('res\n')
53      f_fatal.write('fatal.ctr\n')
54      f_fatal.write('\n')
55      f_fatal.write('fatigue')
56      f_fatal.close()
57
58      #running usfos
59      program = r'c:\"program files"\"USFOS (64 bit) 8.9"\bin\usfos.exe'
60      run_inp = program + ' < ' + inp_usfos
61      os.system(run_inp)
62
63      #running dynres
64      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\dynres.exe" < ...
            parametersdynres_acc1.txt')
65      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\dynres.exe" < ...
            parametersdynres_acc2.txt')
66      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\dynres.exe" < ...
            parametersdynres_acc3.txt')
67      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\dynres.exe" < ...
            parametersdynres_acc4.txt')
68      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\dynres.exe" < ...
            parametersdynres_disp1.txt')
69      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\dynres.exe" < ...
            parametersdynres_disp2.txt')
70      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\dynres.exe" < ...
            parametersdynres_disp3.txt')
71      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\dynres.exe" < ...
            parametersdynres_disp4.txt')
72      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\dynres.exe" < ...
            parametersdynres_waveelev.txt')
73
74      #running fatal
75      os.system('"C:\\Program Files\\USFOS (64 bit) 8.9\\bin\\fatal.exe" < ...
            fatal_parameters.txt')
76
77      #Delete RAF-file to save space
78      os.remove('res.raf')
79
80      #change bac to working directory
81      os.chdir(cwd)
82
83
84  def ...
        create_folders(run_names,run_loc_master,head_name,model_name,df,fatal_name, ...
        load_name):
85      a = 0
86      run_loc = []
87      for run in run_names:
88          run_loc.append(run_loc_master + run)
89          os.mkdir(run_loc[a])
90          #copying input files to runfolder
```

```
91          copyfile(run_loc_master+'\\'+head_name+'.fem',run_loc[a]+'\\'+head_name+'.fem')
92          copyfile(run_loc_master+'\\'+model_name+'.fem',run_loc[a]+'\\'+model_name+'.fem')
93          copyfile(run_loc_master+'\\'+fatal_name+'.ctr',run_loc[a]+'\\'+fatal_name+'.ctr')
94          copyfile(run_loc_master+'\\'+load_name+'.fem',run_loc[a]+'\\'+load_name+'.fem')
95          # Change "run case" parameters
96          change_run_case_param(run_loc[a]+'\\'+head_name+'.fem', df, a)
97          change_run_case_param(run_loc[a]+'\\'+model_name+'.fem', df, a)
98
99          a+=1
100     return run_loc
101
102
103 def change_run_case_param(filename, df, a):
104
105     if "control" in filename:
106         with open(filename, 'r') as file:
107             filedata = file.read()
108
109         # Replace the target string
110         for name in df.columns:
111             filedata = filedata.replace(name, str(df.at[a,name]))
112
113         # Write the file out again
114         with open(filename, 'w') as file:
115             file.write(filedata)
116
117     return
118
119
120 def usfos_mp(head_name, model_name, load_name, run_loc_master):
121     pool = mul.Pool(2)
122     partial_usfos = partial(run_usfos, head_name=head_name, ...
            model_name=model_name,
123                                 load_name=load_name, file_loc=run_loc_master)
124
125     pool.map(partial_usfos, run_loc)
126
127     pool.close()
128     pool.join()
129
130
131 if __name__ == "__main__":
132     run_loc_master = "RunFolder\\Stripped_topside_T2\\Sim_disp_comparison2\\"
133     head_name = 'PH_control'
134     model_name = 'PH_stru'
135     fatal_name = 'fatal'
136     load_name = 'PH_ufo_load'
137     results_name = 'res'
138
139     # Define run_case parameters to be used in head.fem and stru.fem
140     df = pd.read_csv(run_loc_master + 'runCase_param.txt', sep = "\s+", ...
            header=None, skiprows=1)
141     df.columns = ['endT', 'Hs', 'Tp', 'dir', 'dimxy', 'W_mean', 'Wdr']
142
143     # Create folder names
144     run_names = []
145     for i in range(df.shape[0]):
146         run_names.append('run_%i'%i)
147
148     # Create folders and run simulations
149     run_loc = create_folders(run_names, run_loc_master, head_name, ...
            model_name, df, fatal_name, load_name)
150     usfos_mp(head_name, model_name, load_name, run_loc_master)
```

E.2 Code to tabulate dataset for ML, *post_process_Weibull.py*

```
"""
Goal: Use 2-parameter Weibull parameters as input for the ML-model
@Author: Gjermund Smedsland
"""
import numpy as np
import pandas as pd
import math as m
from statistics import mean, stdev, variance
import xlwings as xw
from scipy.stats import weibull_min
from scipy.signal import find_peaks, lfilter, butter, freqz

def collect_displacements(run_loc_master, name, counter):

    FILElist = ['DISP1.txt', 'DISP2.txt', 'DISP3.txt', 'DISP4.txt']

    DF = pd.concat([pd.read_csv(run_loc_master + name + '\\' + path, names ...
        = ['time' + str(idx), path[:-4] + str(counter)], sep='\s+') for ...
        idx, path in enumerate(FILElist)], axis=1)
    DF.rename(columns={'time0':'time'}, inplace=True)
    for i in range(1,2):
        DF.drop(columns=['time' + str(i)], inplace=True)

    return DF

def butter_lowpass(cutoff, fs, order=5):
    return butter(order, cutoff, fs=fs, btype='low', analog=False)

def butter_lowpass_filter(data, cutoff, fs, order=5):
    b, a = butter_lowpass(cutoff, fs, order=order)
    y = lfilter(b, a, data)
    return y

def EX_and_STD(DF):

    # Use the packages from "statistics". Numpy provides different STD(X)
    list1 = [] # Lambda for nodal disp in x-dir for node 1
    list2 = [] # Lambda for nodal disp in y-dir for node 1
    list3 = [] # Lambda for nodal disp in x-dir for node 2
    list4 = [] # Lambda for nodal disp in y-dir for node 2
    list5 = [] # k for nodal disp in x-dir for node 1
    list6 = [] # k for nodal disp in y-dir for node 1
    list7 = [] # k for nodal disp in x-dir for node 2
    list8 = [] # k for nodal disp in y-dir for node 2

    for name in DF.columns:
        if name != 'time':

            # Filter the dislpacements to create cleaner datasets
            sample_interval = 0.1 # [seconds]
            sample_freq = 1 / sample_interval # [Hz]
            order = 6
            cutoff = 0.2 # [Hz] - cutoff frequency
            b, a = butter_lowpass(cutoff, sample_freq, order) # Filter ...
                coefficients
            DF[name] = butter_lowpass_filter(DF[name], cutoff, sample_freq, ...
                order)

            DF[name] -= mean(DF[name])
```

```
56
57              # Find the local maxima of each displacement sinusoidal
58              peaks, _ = find_peaks(DF[name], height=mean(DF[name]))
59
60              df_new = pd.DataFrame()
61              df_new[name] = DF[name][peaks]
62              df_new['disp_sorted'] = np.sort(df_new[name])
63              df_new.reset_index(inplace=True)
64              df_new.drop(columns=['index'], inplace=True)
65
66              # Fit the 2-parameter Weibull distribution to the peaks
67              k, dummy123, lama = weibull_min.fit(df_new[name], floc=0)
68
69              for _ in range(50):
70                  if name[4] == '1':
71                      list1.append(lama)
72                      list5.append(k)
73                  elif name[4] == '2':
74                      list2.append(lama)
75                      list6.append(k)
76                  elif name[4] == '3':
77                      list3.append(lama)
78                      list7.append(k)
79                  elif name[4] == '4':
80                      list4.append(lama)
81                      list8.append(k)
82
83      return list1, list2, list3, list4, list5, list6, list7, list8
84
85
86  def collect_damage(run_loc_master, run_names):
87      df = pd.read_csv(run_loc_master + run_names + '\\fatigue.dam', ...
            sep='\s+', skiprows=15)
88
89      df.drop('BPos4', axis=1, inplace=True)
90      df.drop('#', axis=1, inplace=True)
91      df.dropna(inplace=True)
92
93      return df
94
95
96  def find_max2(df_dmg):
97      df_upd = pd.DataFrame()
98      for name in df_dmg.columns:
99          df_upd[name] = pd.to_numeric(df_dmg[name])
100     df_upd.columns = ["CPos1", "CPos2", "CPos3", "CPos4", "BPos1", "BPos2", ...
            "BPos3", "BPos4"]
101     df_upd['maxVal'] = df_upd.max(axis=1)
102     df_upd['maxPos'] = df_upd.idxmax(axis="columns")
103     val_list = df_upd['maxVal'].to_numpy()
104     pos_list = df_upd['maxPos'].to_numpy()
105
106     return val_list, pos_list
107
108
109 def which_joint(row):
110     start = [1, 6, 12, 17, 23, 30, 37, 44]
111     stop = [5, 11, 16, 22, 29, 36, 43, 50]
112     for idx, val in enumerate(start):
113         if (row['HotSpot'] ≥ val) and (row['HotSpot'] ≤ stop[idx]):
114             return (idx+1)
115     else:
116         return 0
```

```
117
118
119 if __name__ == "__main__":
120     run_loc_master = "RunFolder\\Stripped_topside_T2\\Sim6\\"
121
122     # Extract run_case parameters
123     df = pd.read_csv(run_loc_master + 'runCase_param.txt', sep = "\s+", ...
        header=None, skiprows=1)
124     df.columns = ['endT', 'Hs', 'Tp', 'dir', 'dimxy', 'W_mean', 'Wdr']
125
126     # Extract folder names
127     run_names = []
128     for i in range(df.shape[0]):
129         run_names.append('run_%i'%i)
130
131     ### Post processing
132     # Collect displacement data
133     dummy_data = []
134     for idx in range(0,len(run_names)):
135         dummy_data.append(collect_displacements(run_loc_master, ...
            run_names[idx], idx))
136         DF = pd.concat(dummy_data, axis = 1)
137
138     # Remove duplicate columns from displacement data
139     DF = DF.loc[:, ¬DF.columns.duplicated()]
140     for name in DF.columns:
141         if 'time' in name and name != 'time':
142             DF.drop(name, axis=1, inplace=True)
143
144     ### Create a new dataframe containing the most relevant diapacement- ...
        and fatigue data
145     DF_excel = pd.DataFrame()
146
147     HotSpot_list = []
148     for _ in range(0,len(run_names)):
149         for j in range(1,51):
150             HotSpot_list.append(j)
151
152      # Extract maximum fatigue damage
153         Max_DMG = []
154         DMG_pos = []
155         for name in run_names:
156             DF_dmg = collect_damage(run_loc_master, name)
157             dummy_val, dummy_pos = find_max2(DF_dmg)
158             Max_DMG.append(dummy_val)
159             DMG_pos.append(dummy_pos)
160
161     DF_excel['HotSpot'] = HotSpot_list
162     DF_excel['Joint'] = DF_excel.apply(lambda row : which_joint(row), axis=1)
163     DF_excel['Disp1lambda'], DF_excel['Disp2lambda'], ...
        DF_excel['Disp3lambda'], DF_excel['Disp4lambda'], ...
        DF_excel['Disp1k'], DF_excel['Disp2k'], DF_excel['Disp3k'], ...
        DF_excel['Disp4k'] = EX_and_STD(DF)
164     DF_append = pd.DataFrame(np.repeat(df.values, 50, axis=0))
165     DF_append.columns = df.columns
166
167     for name in DF_append.columns:
168         DF_excel[name] = DF_append[name]
169     DF_excel['Damage'] = np.reshape(Max_DMG, (DF_excel.shape[0], 1))
170     DF_excel['ClockPos'] = np.reshape(DMG_pos, (DF_excel.shape[0], 1))
171
172     # Export DataFrame to Excel
173     wb = xw.Book('RunFolder\HotSpots.xlsx')
```

```
174     wb.sheets.add("Sim6_Weibull2")
175     ws = wb.sheets["Sim6_Weibull2"]
176     ws['A1'].options(pd.DataFrame, index=False, expand='table').value = ...
            DF_excel
177     wb.save()
178     wb.close()
```

E.3 code to identify relevant sea states, *wave_analysis_site.py*

```
1  """
2  Goal: Extract and evaluate wave elevation from Valhall site.
3  @Author: Gjermund Smedsland
4  """
5  import numpy as np
6  import pandas as pd
7  import matplotlib.pyplot as plt
8  import seaborn as sns
9  import statsmodels.api as sm
10 from scipy.signal import butter, lfilter
11 import os
12 from statistics import mean, variance
13 import math as m
14 import xlwings as xw
15
16 def butter_lowpass(cutoff, fs, order=5):
17     return butter(order, cutoff, fs=fs, btype='low', analog=False)
18
19 def butter_lowpass_filter(signal, cutoff, fs, order=5):
20     b, a = butter_lowpass(cutoff, fs, order=order)
21     return lfilter(b, a, signal)
22
23 def Filter_Signal(signal, sample_freq):
24     order = 6       # [-]
25     cutoff = 0.25    # [Hz] - cutoff frequency
26     return butter_lowpass_filter(signal, cutoff, sample_freq, order)
27
28 def Calculate_Autocorr(signal):
29     signal_corrected = signal - mean(signal) # Correct such that the mean ...
           is subtracted; EX=0 (unbiased)
30     return sm.tsa.acf(signal_corrected)
31
32 def Create_Scatter_Matrix(Hs_list, Tp_list, Hs_bins, Tp_bins):
33     Scatter_matrix = np.zeros((50,40), dtype=int)
34
35     for idx in range(len(Hs_list)):
36         for i in range(len(Hs_bins)):
37             for j in range(len(Tp_bins)):
38                 if (Hs_bins[i-1] ≤ Hs_list[idx]) and (Hs_list[idx] < ...
                        Hs_bins[i]) and (Tp_bins[j-1] ≤ Tp_list[idx]) and ...
                        (Tp_list[idx] < Tp_bins[j]):
39                      Scatter_matrix[i][j] += 1
40
41     return Scatter_matrix
42
43 def plot_problems(DF, df):
44     sns.lineplot(x="time", y="wave_filtered", data=DF)
45     plt.show()
46
47     sns.lineplot(x=df.index, y="Rxx", data=df, label="Rxx")
48     sns.lineplot(x=df.index, y="Rxx_acc", data=df, label="Rxx_ddot")
```

```
49      plt.grid()
50      plt.legend()
51      plt.show()
52      return
53
54  if __name__ == "__main__":
55      sample_interval = 0.5 # [s]
56      sample_frequency = 2 # [Hz]
57      Hs_list = []
58      Tp_list = []
59      time_list = []
60
61      ## Time series handling
62      main_path = "RunFolder\\wave\\"
63      path_list = os.listdir(main_path)
64
65      for path in path_list:
66          DF = pd.read_csv(main_path + path, sep=",", delimiter=",", ...
                index_col=None, skiprows=3, names=["wave"])
67
68          DF['wave'] = DF['wave'] - mean(DF['wave'])
69          DF['wave_filtered'] = Filter_Signal(DF['wave'], sample_frequency)
70          DF['time'] = DF.index * sample_interval
71
72          sns.lineplot(x='time', y='wave', data=DF)
73          plt.show()
74
75          ## Characteristic properties
76          EndT = DF['time'].iloc[-1] # Should be 3h=10800s
77          df = pd.DataFrame()
78          df['Rxx'] = variance(DF['wave_filtered']) * ...
                Calculate_Autocorr(DF['wave_filtered'].values)
79          df['Rxx_acc'] = (df['Rxx'].shift(1)  - 2*df['Rxx'] + ...
                df['Rxx'].shift(-1)) / sample_interval**2
80
81          m0 = df.at[0, 'Rxx']
82          m2 = abs(df.at[1, 'Rxx_acc'])
83          Hs = 4*m.sqrt(m0)
84          Tp = 1.41 * 2 * np.pi * m.sqrt(m0 / m2)
85          print("m0 = {0} | m2 = {1} | Hs = {2} | Tp = {3}".format(m0, m2, ...
                Hs, Tp))
86
87          if m.isnan(m0) or m.isnan(m2):
88              plot_problems(DF, df)
89              continue
90          elif Hs > 10:
91              plot_problems(DF, df)
92              continue
93          elif Tp > 19:
94              plot_problems(DF, df)
95              continue
96          else:
97              Hs_list.append(Hs)
98              Tp_list.append(Tp)
99              time_list.append(path[10:-10])
100
101     ## Plot scatter diagram
102     plt.figure()
103     plt.plot(Tp_list, Hs_list, '.')
104     plt.xlabel('Tp [s]')
105     plt.ylabel('Hs [m]')
106     plt.grid()
107     plt.show()
```

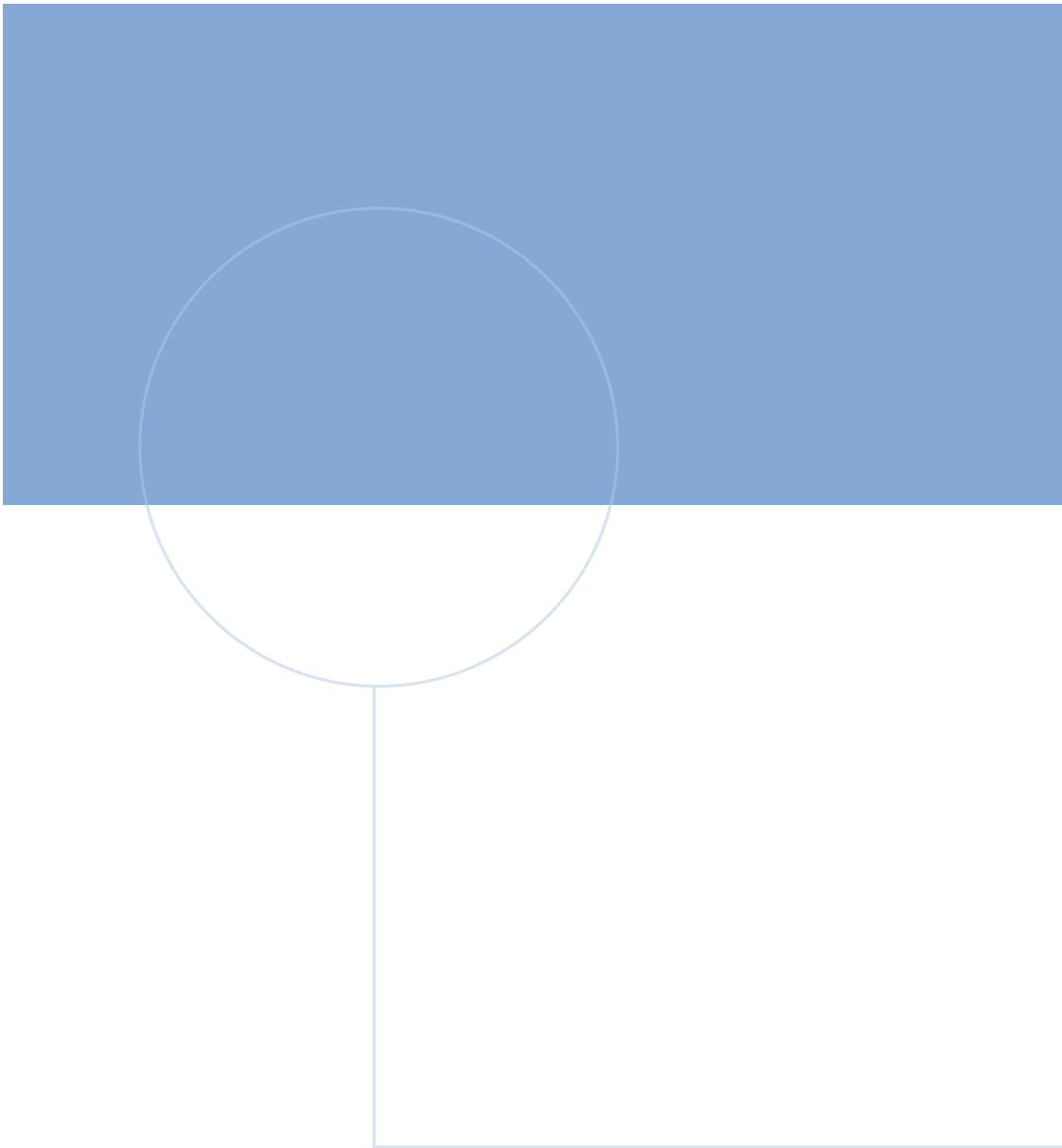
```
108
109     Hs_bins = np.linspace(0.5,25,50)
110     Tp_bins = np.linspace(1,40,40)
111     scatter_matrix = Create_Scatter_Matrix(Hs_list, Tp_list, Hs_bins, Tp_bins)
112     DF_excel = pd.DataFrame(scatter_matrix, index=Hs_bins)
113     DF_excel.columns = Tp_bins
114
115     # Export to Excel
116     wb = xw.Book('RunFolder\HotSpots.xlsx')
117     wb.sheets.add('Scatter_Diagram_hour')
118     ws = wb.sheets['Scatter_Diagram_hour']
119     ws['A1'].options(pd.DataFrame, index=True, expand='table').value = DF_excel
120     wb.save()
121     wb.close()
122
123     # Just to verify the time of the sea states
124     DF_excel2 = pd.DataFrame()
125     DF_excel2['Hs'] = Hs_list
126     DF_excel2['Tp'] = Tp_list
127     DF_excel2['DateTime'] = time_list
128
129     wb2 = xw.Book('RunFolder\HotSpots.xlsx')
130     wb2.sheets.add('Sea_state_time_hour')
131     ws2 = wb2.sheets['Sea_state_time_hour']
132     ws2['A1'].options(pd.DataFrame, index=True, expand='table').value = ...
            DF_excel2
133     wb2.save()
134     wb2.close()
```

E.4 Code to tabulate monitored data, *January2021_disp.py*

```
1  """
2  Goal: Use 2-parameter Weibull parameters as input for the ML-model
3  @Author: Gjermund Smedsland
4  """
5  import os
6  import pandas as pd
7  import numpy as np
8  import math as m
9  from statistics import mean, stdev, variance
10 import xlwings as xw
11 import matplotlib.pyplot as plt
12 import seaborn as sns
13 from scipy.signal import find_peaks, lfilter, butter, freqz
14 from scipy.stats import weibull_min
15
16 def butter_lowpass(cutoff, fs, order=5):
17     return butter(order, cutoff, fs=fs, btype='low', analog=False)
18
19 def butter_lowpass_filter(data, cutoff, fs, order=5):
20     b, a = butter_lowpass(cutoff, fs, order=order)
21     y = lfilter(b, a, data)
22     return y
23
24 def EX_and_STD():
25
26     list1 = [] # Lambda for nodal disp in x-dir for node 1
27     list2 = [] # Lambda for nodal disp in y-dir for node 1
28     list3 = [] # Lambda for nodal disp in x-dir for node 2
29     list4 = [] # Lambda for nodal disp in y-dir for node 2
```

```
    list5 = [] # k for nodal disp in x-dir for node 1
    list6 = [] # k for nodal disp in y-dir for node 1
    list7 = [] # k for nodal disp in x-dir for node 2
    list8 = [] # k for nodal disp in y-dir for node 2

    main_path = "RunFolder\\valph_motion\\"
    path_list = os.listdir(main_path)

    for path in path_list:
        if "2101" in path:  # If January 2021
            print("path is: {0}".format(path))
            DF = pd.read_csv(main_path + path, sep=",", delimiter=",", ...
                index_col=None, skiprows=1)
            DF.drop([0], inplace=True)
            DF.drop(columns=["Loc1_EW_AC", "Loc1_NS_AC", "Loc1_Res_Disp", ...
                "Loc2_EW_AC", "Loc2_NS_AC", "Loc2_Res_Disp"], inplace=True)
            DF.columns=["DISP1", "DISP2", "DISP3", "DISP4"]

            DF.columns = DF.columns.str.strip()
            for name in DF.columns:
                DF[name] = (DF[name].apply(lambda x : float(x)))
                DF[name] = -DF[name]/1000 # unit is [mm], transfer to USFOS ...
                    coordinate system

            for name in DF.columns:
                if name != 'time':

                    # Filter the dislpacements to create cleaner datasets
                    sample_interval = 0.292969 # [seconds]
                    sample_freq = 1 / sample_interval # [Hz]
                    order = 6
                    cutoff = 0.2 # [Hz] - cutoff frequency
                    b, a = butter_lowpass(cutoff, sample_freq, order) # ...
                        Filter coefficients
                    DF[name] = butter_lowpass_filter(DF[name], cutoff, ...
                        sample_freq, order)

                    DF[name] -= mean(DF[name]) # Subtract the mean drift off

                    # Find the local maxima of each displacement sinusoidal
                    peaks, _ = find_peaks(DF[name], height=mean(DF[name]))

                    df_new = pd.DataFrame()
                    df_new[name] = DF[name][peaks]
                    df_new['disp_sorted'] = np.sort(df_new[name])
                    df_new.reset_index(inplace=True)
                    df_new.drop(columns=['index'], inplace=True)

                    # Fit the 2-parameter Weibull distribution to the peaks
                    k, dummy123, lama = weibull_min.fit(df_new[name], floc=0)

                    for _ in range(50):
                        if name[4] == '1':
                            list1.append(lama)
                            list5.append(k)
                        elif name[4] == '2':
                            list2.append(lama)
                            list6.append(k)
                        elif name[4] == '3':
                            list3.append(lama)
                            list7.append(k)
                        elif name[4] == '4':
                            list4.append(lama)
```

```
88                              list8.append(k)
89
90      return list1, list2, list3, list4, list5, list6, list7, list8
91
92  def which_joint(row):
93      start = [1, 6, 12, 17, 23, 30, 37, 44]
94      stop = [5, 11, 16, 22, 29, 36, 43, 50]
95      for idx, val in enumerate(start):
96          if (row['Conn'] ≥ val) and (row['Conn'] ≤ stop[idx]):
97              return (idx+1)
98      else:
99          return 0
100
101 if __name__ == "__main__":
102
103     # Import environmental conditions
104     DF_env_cond = pd.read_excel("RunFolder\\HotSpots.xlsx", ...
            sheet_name="Env_cond_jan_2021", index_col=None, na_values=0)
105     DF_env_cond = DF_env_cond.dropna()
106     DF_env_cond2 = pd.DataFrame(np.repeat(DF_env_cond.values, 50, axis=0))
107     DF_env_cond2.columns = DF_env_cond.columns
108
109     ### Create a new dataframe containing the most relevant dispacement- ...
            and environmental data
110     DF_excel = pd.DataFrame()
111
112     # Append joint and connection indexation
113     HotSpot_list = []
114     for _ in range(DF_env_cond.shape[0]):
115         for j in range(1,51):
116             HotSpot_list.append(j)
117
118     DF_excel['Conn'] = HotSpot_list
119     DF_excel['Joint'] = DF_excel.apply(lambda row : which_joint(row), axis=1)
120
121     # Append Weibull distribution parameters fitted to displacement peaks
122     DF_excel['Disp1lambda'], DF_excel['Disp2lambda'], ...
            DF_excel['Disp3lambda'], DF_excel['Disp4lambda'], ...
            DF_excel['Disp1k'], DF_excel['Disp2k'], DF_excel['Disp3k'], ...
            DF_excel['Disp4k'] = EX_and_STD()
123
124     # Append environmental conditions
125     for name in DF_env_cond2.columns:
126         DF_excel[name] = DF_env_cond2[name]
127
128
129     wb = xw.Book('RunFolder\HotSpots.xlsx')
130     wb.sheets.add("January2021_disp")
131     ws = wb.sheets["January2021_disp"]
132     ws['A1'].options(pd.DataFrame, index=False, expand='table').value = ...
            DF_excel
133     wb.save()
134     wb.close()
```



NTNU

Norwegian University of
Science and Technology