An indirect method for predicting bending moments with machine learning

D06-2022

Sverre W. Berge, Runar B. Eckholdt, Markus Ø. Leander
Sondre T. Løver, Jørgen W. Søbstad, Martin Sørensen

Bachelor thesis

Faculty of Technology, Natural Sciences and Maritime Sciences

University of
South-Eastern Norway

TechnipFMC

I Acknowledgements

We would like to express our gratitude to Halvor S. Gustad and Dag André Fjeldstad, our research supervisors, Per Thomas Moe and Lasse Moldestad for their outstanding guidance, enthusiastic encouragement, and meaningful critiques of this project.

We would also like to thank Kjell Enger, for his advice and assistance in keeping our progress on schedule, and giving us great feedback throughout the process.

Finally, we would like to extend our thanks to the University of South-Eastern Norway for letting us use components and machines to visualize our product and making the whole learning experience as joyful and pleasant as possible.

II Abstract

In the offshore industry, real-time monitoring of wellhead fatigue is increasingly in demand. The current method of measuring with strain sensors have proven to be expensive and prone to inaccuracies in a complex marine riser system, giving rise to the need for alternative methods.

The purpose of this project is to find an indirect method of measuring bending moments acting upon a subsea wellhead, which in turn will be used in fatigue calculations. This will be approached by training a machine learning model to estimate the bending moment with data from motion sensors placed on a marine riser.

Using artificial neural networks this project analyzed data, from both historical operations and simulations in the dynamic analysis software OrcaFlex, to estimate wellhead fatigue. Furthermore, conventional filters such as Kalman filters were created and utilized to preprocess data before the neural networks made predictions. A simplified mathematical model of the system using Lagrangian and Hamiltonian mechanics was partly used as input for the Kalman filter.
This gave rise to a recurrent neural network with a R^2 value of 0.95, using only accelerometers and gyroscopes. A convolutional neural network trained on simulated OrcaFlex data gave a R^2 value of 0.93, when validated against historical data.

This project proves to a large extent that a machine learning model can be used to predict wellhead fatigue with high accuracy, potentially making strain sensors redundant. Training convolutional neural networks on simulated data should be expanded further and replicated over a longer period of time. Further studies are needed to explore the use of Kalman filters based on Euler-Lagrange equations in artificial neural networks.

Contents

List of Figures

List of Tables

Acronyms

1D-CNN 1-Dimensional Convolutional Neural Network. 73, 135

2D-CNN 2-Dimensional Convolutional Neural Network. 73

ABC Abstract Base Class. 68

AHRS Attitude and heading reference system. 40

ANN Artificial Neural Network. 26, 65, 66, 68, 69, 116, 123, *Glossary:* artificial neural network

API Application Programming Interface. 26

BM Bending Moment. 12, 13, 29, 37, 41, 43, 48, 49, 67, 76, 104, 114–117, 119–121, 123, 128, 129, 131, 132, 136

BOP Blowout Preventer. 29, 49–52, 88, 102, 109, *Glossary:* blowout preventer

CNN Convolutional Neural Network. 73, 75, 101, 132, 135, *Glossary:* convolutional neural network

COB Crossover Box. 31, 37, 46, 47, 49, 56, 70, 114, 136, 204, 205, *Glossary:* crossover box

conv-1d 1-Dimensional Convolutional Layer. 73–76, 128

conv-2d 2-Dimensional Convolutional Layer. 73

CSV Comma-separated values. 56

CUDA Compute Unified Device Architecture. 26, *Glossary:* cuda

DF Data Frame. 26, 66

DSA Deepsea Atlantic. 28

DWS Deep Water Strain Sensor. 40, 43, 49, 54, 55, *Glossary:* deep water strain sensor

FBD Free Body Diagram. 88, 89

FMAP Feature-Map. 73, 76

FNN Feed-Forward Neural Network. 69, 70, 73, 75, 76, 123, 124, 131

HNN Hamiltonian Neural Networks. 136

Hs Significant Wave Height. 32, 57, *Glossary:* significant wave height

IFMAP Input Feature-Map. 73, 74, 76

LCG Load case file. 59, *Glossary:* Load case file

LFJ Lower Flex Joint. 29, 51, 88, 89, 95, 109

LMRP Lower Marine Riser Package. 29, 51, 88, 102, 109, *Glossary:* lower marine riser package

LSTM Long short-term memory. 72, 126, 135, *Glossary:* long short-term memory

MEMS Micro Electro Mechanical Sensors. 13, 40, 41, 131

ML Machine Learning. 13, 14, 26, 27, 29, 37, 45, 52, 54, 56, 57, 62, 99, 101, 104, 105, 107, 115–117, 123, 131, 132, 136

MR Marine Riser. 12, 29, 32, 51, 88, 102, 204, *Glossary:* marine riser

MRA Marine Riser Adapter. 29, 48, 49, 55, 88, 109, 110, 136, 204

MSE Mean Square Error. 70

ODE Ordinary Differential Equation. 86, 87, 91, 92, 94, 108, 111, 112

OFMAP Output Feature-Map. 73, 74, 76

ReLU Rectified Linear Unit. 70, 71, 125, *Glossary:* rectified linear unit

RES Result file. 50, 137, *Glossary:* Result file

RFC Rainflow Counting. 67

RFJ Reactive Flex Joint. 15, 29, 31, 48, 70, 76, 85, 88, 101, 102, 104–106, 109, 117, 120–122, 125, 128, 129, 132, 133, 135, *Glossary:* reactive flex joint

RNN Recurrent Neural Network. 70–72, 125–127, 132, 135, *Glossary:* recurrent neural network

SMU Subsea Motion Unit. 13, 31, 38, 40, 43, 46, 51, 52, 54–56, 70, 100, 102, 114, 116, 119, 122, 128, 137, 203, 204, *Glossary:* subsea motion unit

TOTALDIRM Historical mean wave direction. 57, 59

Tp Spectral Peak Period. 32, 57, *Glossary:* spectral peak period

TSL Target Segment Length. 46, 203–205

UFJ Upper Flex Joint. 29

UML Unified Modelling Language. 18, 27, 68

WAMS Well Access Management System. 29, 49, 132, 136, *Glossary:* well access management system

WaveDir principal wave direction. 57, 59

WH Wellhead. 12, 13, 15, 29, 32, 37, 48, 49, 51, 55, 67, 70, 88, 89, 95, 104, 109, 110, 132

Glossary

artificial neural network A node based, trainable neural network inspired by neurons in the brain [2, p. 801]. 26

blowout preventer A specialized valve use to seal, control, and monitor wells to prevent blowouts. 29

buoy OrcaFlex component: They can be used for many different purposes, in our case functioning as weightless SMU sensors. 51, 52, 100

convolutional neural network A Convolutional Neural Network is typically a Feed-forward neural network with an arbitrary amount of 1, 2 or 3-Dimensional Convolutional Layers at the front, typically used for feature extraction. 73

crossover box The crossover box gathers data for each of the locations where several deep water strain sensors are placed. 31

cuda Compute Unified Device Architecture. *"CUDA is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs"* [6]. 26

deep water strain sensor Sensors used to measure strains at specific locations on the subsea stack, See 4.1.2.3. 40

dilation Dilation decides the distance between each weight within the kernel used in convolutional neural networks. Used to skip values within the kernel size range. 74

drill pipe Connects the rig and the assembly at the seabed. Used for drilling, located on the inside of the pipes. 136

epoch A complete iteration through a dataset during neural network training. 68, 124

fatigue Fatigue is a progressive formation of cracks due to cycled stresses. These cracks are usually microscopic and fine, making them difficult to detect. The material could fail at stresses below their yield point. 12, 13, 15, 34, 37, 67, 99, 114, 115, 123, 126, 129, 135

kernel size Represents the dimensions of a kernel used in a convolutional layer. 74, 75

Load case file Grouped parameters load case file. Contains parameters that often gets changed, and parameters that change during different simulations. Used to setup multiple simulations, with different scenarios. 59

long short-term memory A specialized recurrent neural network architecture that is designed to enable preservation of information over many time steps. 72

lower marine riser package The upper section of the BOP stack that interfaces between the lower stack and the drilling riser. 29

marine riser Cylindrical conduits that are used to transfer crude oil to the rig. 12

non-saturated A function $f(x)$ where the following is true: $\lim_{x\to\infty} f(x) = \infty$. 70, 71

O2-system TFMCs in-house developed GUI based on Python to run simulations in OrcaFlex.. 25, 45, 50, 52, 56, 136, 137

overfitting Overfitting in machine learning is when the model is paying to much attention to the data it is training on, thus making poorer predictions on unseen data [2, p.673]. 62, 124, 125, 128, 131

reactive flex joint The RFJ is mounted outside the lower flex joint, and counteracts the bending moment in the marine riser. It is partly made of vulcanized elastomer, giving it a non-linear behaviour. 15

rectified linear unit Activation function: $f(x) = \max(0, x)$. 70

recurrent neural network A type of neural network that uses an internal state to predict sequences of data. 70

Result file Result extraction file. Can define the result extraction, meaning which data to gather and layout of the result files. 50

sigmoid Activation function: $\sigma(x) = \frac{1}{(1+e^{-x})}$. 76

significant wave height The mean of the highest third of the waves. 32

spectral peak period The wave period associated with the most energetic waves in the total wave spectrum at a specific point. 32

stack The stack is one of two or more units which control well pressure, and contain the Wellhead and Blowout Preventers.. 13, 29, 32, 37, 41, 48, 62, 66, 85, 102

stride The stride decides the distance a kernel used in a convolutional layer should move per calculation. 74

subsea motion unit A sensor that measures both acceleration and angular rate. The data can be combined and integrated to find the inclination off the motion model. 13

tanh Activation function: $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$. 125

wave train A group of waves with similar or equal wavelengths traveling in the same direction. 59

well access management system A live riser and wellhead monitoring software provided by TFMC. 29

Nomenclature

$\bar{y}$	Statistical mean
ℓ	Length
$\hat{y}$	Estimation of output
$\mathcal{H}$	Hamiltonian
$\mathcal{L}$	Lagrangian
$\mathcal{N}$	Normal distribution
ω	Rotational velocity
σ	Standard deviation
$\star$	Cross-Correlation Operator
θ	Angle
ε	Strain
b	Damping constant
E	Young's modulus
g	Gravitational acceleration constant
I	Second momenta of inertia
k	Spring constant
M	Bending moment
m	Mass
p	Momentum
R	Viscous damping
S	Stress
T	Kinetic Energy
V	Potential Energy
X, Y, Z	Coordinates based on global axis
x, y, z	Positions

Chapter 1

Introduction

1.1 Project description

Our group formed in May 2021 – seven months before the start of the bachelor project. A shared desire for a challenging project that makes an actual impact for its stakeholders led us to form this multidisciplinary group. Spanning a decade in age difference, from business owner to minimal work experience. Our differences are one of the key factors in making our group strong, as our contrasting experiences forge differing viewpoints allowing us to see more sides of each problem, leading to a better solution.

Our collaboration with TFMC began in August 2021, we held several workshops and meetings evolving and defining our project throughout the fall. These extensive preparations allowed us to work towards a purpose from day one of the project, and is a contributor to its success.

1.1.1 Background

The use of semi-submersible floating platforms for drilling operations in the North Sea has led to an increased focus on material fatigue monitoring of the subsea equipment, and the Wellhead (WH) mounted to the seabed.

The monitoring of material fatigue in the WH is the most critical, as it has a finite lifespan before being plugged and taken out of operation. Accurate measurements of material fatigue increase the lifespan of the WH because the most conservative range of the margin of error is used. Increased lifespan equals longer operation time, which in turn leads to higher return on investment for the operation.

The industry standard for measuring material fatigue is to mount strain sensors on the WH connector, the part of the Marine Riser (MR) that is connected to the WH, and calculate the Bending Moment (BM) acting upon the WH from the strain measurements. BMs, in turn, are used to calculate the material fatigue of the WH.

The strain sensors are very accurate, and they are in theory one of the best methods of measuring bending moments. They have, however, been proven to be expensive and prone to inaccuracies.

1.1.2 Problem

Due to the shortcomings of the direct measurements of BMs with strain sensors, TFMC approached us with a problem that could be translated into a bachelor project – to find an indirect method of measuring the BM acting upon a WH using motion sensors placed on other parts of the system.

This should be done through developing a Machine Learning (ML) model and train it on simulated data from dynamic simulations in the finite element analysis software OrcaFlex. The ML model should then, as a best case scenario, be able to predict the BM acting upon the WH to measure material fatigue in real-time.

TFMC has previously seen good results when using ML models in research on similar systems. This research did not, in contract, rely solely on motion sensors. TFMC therefore expressed an interest in investigating fusion algorithms for better angle approximations from the motion sensors at the massive stack sitting on top of the WH, both with a mathematical model, and with different kind of filters.

1.1.3 Literature review

Previous research has made numerous attempts to find the BM acting upon the WH through an indirect method. Reports such as [7] and [8] have given us a better understanding of the problem at hand.

This project builds on research done by TFMC, but also from Halvor S. Gustad [9]. The previous work and research they have done, has given us the opportunity to continue their research with ML on maritime structures.

The given problem is based on subjects from every discipline that has attributed to the project and can therefore easily be said to be a high-level multidisciplinary approach. Subjects such as Signal analysis, Cybernetics, Structural engineering, Fluid mechanics, Multivariate calculus, Algorithms & Object oriented programming and Object oriented analysis & Design are well used in this project, to name a few.

1.1.4 Current situation

In today's field of WH analysis, the offshore industry mostly rely on direct measurements for material fatigue calculations. The use of low cost and low power Micro Electro Mechanical Sensors (MEMS), referred to as Subsea Motion Unit (SMU) sensors, will show its limitations in regards to inclination measurements. During many operations it is the only available parameter to estimate material fatigue.

An indirect approach is needed in the case of sensor inaccuracy and sensor availability. This applies to both future work and the backlog of past operations. Because sensor resolution and accuracy plays a crucial role in determining the correct angles, an indirect method could make some sensors redundant. This can potentially save TFMC time on addressing inaccuracies in the data, and significant cost in installing and maintaining the sensors.

Chapter 2

Process & Methodology

This chapter describes tools, applications, project models and their affect on the process. Our process model and early planning has contributed to making the project go as smooth as possible, while keeping the schedule on time. Big projects requires a lot of planning and that is why this entire chapter is dedicated to process and methodology.

The chapter also dives deeper into the different types of data and operations we used in this project. The data plays a key role in ML focused tasks. We explain some of the terminology regarding the data, but can not go into depth about the specific information, since the data is regarded as sensitive.

2.1 About the project

2.1.1 Participants

Table 2.1: Participants

Name	Discipline	Role
Martin Sørensen	Mechanical Eng.	Product owner & Theoretical lead
Sverre Weum Berge	Mechanical Eng.	Design lead
Sondre Tiller Løver	Electrical Eng.	Electrical lead
Jørgen Winther Søbstad	Computer Eng.	Lead developer
Runar Bergum Eckholdt	Computer Eng.	Software architect
Markus Øvereng Leander	Computer Eng.	Developer & Scrum master

2.1.2 About TechnipFMC

TechnipFMC (TFMC) is a leading technology provider to the traditional and new energies industry, organized in two business segments - Subsea and Surface Technologies.
TFMC is a multinational company with more that 20,000 employees operating in 41 countries and with an annual revenue of more than 60bn NOK.

In Norway, TFMC have offices in Kongsberg, Lysaker, and Stavanger, in addition to smaller test sites and offices in other locations.

2.1.2.1 Structural analysis engineering

The department that commissioned our project and which we are collaborating with is the Structural Analysis Engineering department, located in Kongsberg and Lysaker.
Using applied mechanics, material science, and applied mathematics, the Structural Analysis Engineering department perform global riser analysis, WH analysis and fatigue analysis for external clients in the offshore industry.

2.1.2.2 Important personnel

During our project we been fortunate to have access to several experienced and highly competent engineers, listed below.

Halvor Snersrud Gustad

Halvor is an Analysis Engineer in the Structural Analysis Engineering department and a PhD-student in applied mathematics at NTNU.
He has been our external advisor for the project and has been available daily for questions and feedback, especially regarding the machine learning model and the mathematical model.

Dag André Fjeldstad

Dag André is a Principal Engineer in the Structural Analysis Engineering department.
He has been the external examiner for the project and has been available for questions and feedback whenever we have needed it, especially regarding the mechanical and electrical part of our project. In February he brought us to a test facility in Horten to perform tests on the Reactive Flex Joint (RFJ).

Lasse Moldestad

Lasse is a Principal Engineer in the Structural Analysis Engineering department.
He has been our main contact regarding OrcaFlex and has had frequent contact and meetings with our mechanical team. In addition, he has participated weekly in our status meetings, providing valuable input and insights.

Per Thomas Moe

Per Thomas is the Manager and former Chief Engineer of the Structural Analysis Engineering department.
He has been instrumental in the formatting of our project and defining TFMC's requirements.
In addition to having frequent contact with our product owner, he has participated weekly in our status meetings, providing valuable input and insights.

Björn Michael Scharoba

Björn is a Senior Software Developer at TFMC, heavily involved in the software part of in the Structural Analysis Engineering department.
He has participated in meetings with the computer engineering students to provide valuable insights regarding data processing and management.

Liwei Wang

Liwei is a Senior Software Engineer at TFMC, heavily involved in the software part of in the Structural Analysis Engineering department.
She has participated in meetings with the computer engineering students to provide valuable insights regarding data processing and management.

2.2 Work model

When choosing a model, we opted to go for an agile approach based on the already predefined Scrum. We found this to be the most beneficial considering the iterative ability of the model. It allowed us to adapt and improve our workflow based on experiences. With the development of multiple prototypes we can continue to test and validate concepts to later introduce them into new versions.

2.2.1 Role definitions

The predefined Scrum model include three predefined roles. These are the product owner, scrum master, and development team. These roles come with their own predefined tasks and fields that they have control over, which we have chosen to tailor to fit our needs and qualities.

2.2.1.1 Product Owner

The product owner is responsible for the vision of the group. The role requires good communication skills, decisiveness and high level of responsibility. Usually, the product owner is solely accountable when looking at the success of the solution. In our team everyone is equally responsible for the success of the solution, but the product owner is to keep the vision pointed and clear so we all unite towards a common goal. The product owner is also in charge of the creation of stories and their addition to the backlog.

Making sure there is good communication within the team, as well as with internal and external stakeholders. This is another responsibility of the product owner and is both for the sake of figuring out if the project is converging towards the ideas of the stakeholders, as well as if it will yield a positive response from the users. The product owner acts in this regard as a de facto product manager, and participates in the sprint reviews and retrospectives to get a better insights for prioritizing the product backlog.

2.2.1.2 Scrum Master

The scrum master's role as cited by [10] "*The Scrum Master acts as a coach to both the development team and the product owner. A Scrum Master also provides process leadership, helping the Scrum team and the rest of the organization develop their own high-performance, organization-specific*". It is the scrum master's role to both coach the team as well as the product owner. If a problem occurs within the development team, it is the scrum masters role to resolve it. The scrum master must be capable to communicate both on a higher level with the product owner, as well as on a lower level with the development team. The scrum master held the daily stand-up meeting as well as the retrospective. In these meetings the scrum master makes certain that conversations stay on topic, and that the time boxes are met.

2.2.1.3 The Development Team

This is where Scrum defines everyone in a software development team. "*Traditional software development approaches define various job types, such as architect, programmer, tester, database administrator, UI designer, and so on. Scrum defines the role of development team, which is simply a cross-functional collection of these types of people.*" [10, p. 16]. Being a multidisciplinary group, we considered it beneficial to divide the work and responsibilities more than this. We have divided the responsibility into more diverse roles to take advantage of our individual strengths.

2.2.1.4 Software Architect

The software architect has an overarching responsibility of the software. It comes with the responsibility to ensure that the architecture of the software is well planned and clean, this includes making sure that a common standard for Unified Modelling Language (UML) architecture is met. The software architect is also responsible for the GitHub repositories.

2.2.1.5 Lead Developer

The lead developer has the main responsibility to ensure that all software related plans stay on the right track. He is also responsible for confirming tasks by the other developers, verify code quality, and host software related meetings when necessary.

2.2.1.6 Theoretical Lead

The theoretical lead has the overarching theoretical responsibility. This is to make sure that the structural design of the system and the forces of nature acting upon it are modelled correct, which is vital to all other tasks in our project.

2.2.1.7 Design Lead

The design lead has the responsibility to have an overview of the design process, which means ensuring that the finite element model, loads and environment are correctly verified and implemented in OrcaFlex.

2.2.1.8 Electrical Lead

The electrical lead has the overall responsibility for handling signal data and filter design. He mainly focuses on signal analysis and document different filter designs and approaches, as well as to include the designs and ideas that works in the main software.

2.2.1.9 Weekly responsibility

Each week we decided upon one chair of the meeting and one referent. These roles rotated each week, making it so everyone got to try their hand at both roles. Their responsibility was to make a minutes of meeting, follow-up documentation, and the agenda for the meeting with our internal advisor.

2.2.2 Workload estimation

Workload estimation is the process of making assumptions of the time demanded to complete a task. With proper workload estimation, we could optimize the number of tasks each member is accountable for within a sprint.

Throughout the project we have met to discuss the method of workload estimation. There has been a common consensus that estimation techniques work for bigger projects, but might be less needed in smaller groups.

With the help of daily stand-up meetings, insight was gained in the process of other team members, and if they had any issues or obstacles. It is important to note that even though we opted not to go for the Scrum method of workload estimation, there was a custom implementation. Each week in the follow-up documentation, the group wrote the time used on

each task and a rough time estimate of the work the coming week. It was devised that it was best left to each member to estimate their individual tasks.

2.2.3 Stand-up meeting

Each morning there was a daily stand-up meeting. This meeting was led by the scrum master, and was to take no longer than 15 minutes. The questions asked during the meeting were as follows:

- What tasks did you do yesterday?
- What tasks will you be doing today?
- Have you encountered any obstacles with your current tasks?

During the meeting no discussions were allowed. The goal of the stand-up is for the group to get a better insight into the work of their colleagues. Once the meeting concludes, the team members can discuss solutions if any obstacles or other topics were brought up.

2.2.4 Epic

An epic as defined by the Scrum book. [10, p. 86] "*stories that are a few to many months in size and might span an entire release or multiple releases. Many people refer to these as epics*". Epics give a bigger picture and a high level overview. An epic should never be moved into a sprint before it is split up into smaller fractions, all the way down to tasks. Once they are small, you can chip away at the epic one task at a time.

2.2.5 Sprint

A sprint is a given time-period that contains planned work derived into tasks. We chose to do weekly sprints from Thursday to Thursday, but adaptable when the situation called for longer sprints. Once a sprint was summarized, a sprint retrospective was held.

2.2.5.1 Sprint retrospective

"*The sprint retrospective is an opportunity to inspect and adapt the process. During the sprint retrospective the development team, scrum master, and product owner come together to discuss what is and is not working with Scrum and associated technical practices.*" [10, p.27]. During the retrospective the group was asked the following questions

- What have we achieved during the sprint?
- What went well this sprint?
- What did not go as well?
- How will we move forward to solve the obstacles during the next sprint?
- What can we do to improve the process?

After the group complete the retrospective, the whole process was repeated. Improving one sprint at a time.

2.2.5.2 Sprint review

Once a week we held a meeting with the external stakeholders at TFMC. The meeting was held every Monday morning. Throughout our meetings the group members presented their current progress to the TFMC attendees, to receive a nominal feedback.

2.2.6 Workflow

We used three different workflow diagrams that will describe how issues move through the scrum board.

2.2.6.1 Bug

Bugs are the issue types where an error has been found and need attention. This could be related to the software, the OrcaFlex model, or anywhere else in the system.

We decided to use a separate diagram because there are some states the bugs should not enter. For example, it should not be possible that fixing the bug will be blocked by another issue. Therefore, the bug issue has its own diagram that is a simplified version of the task diagram.

2.2.6.2 Action

Action is another issue type that has it is own diagram. This issue type covers all the small tasks that involve taking action. Some examples are sending an email, requesting something from someone, or making a phone call. The only other state we need for action, other than "done", is "awaiting response".

2.2.6.3 Task

The task diagram covers all the issue types except bugs and actions. The issue types are task, epic, and story. They cover all the larger tasks during the project.

2.2.6.4 Scrum board status

The workflow of a task in the scrum board is seen in fig. 2.1. All the tasks in the "to do" status are issues that have yet to be started. "To do" is the base status for all issues.

When the assignee starts working on an issue, the issue should be moved to "in progress". The assignee has the responsibility to move them to this state.

Blocking is a status that shall be used if the progress on the issue is blocked by another issue. When changing an issue status to blocked, the assignee for the blocking task should be notified. The assignee working on the blocked task have the responsibility to notify the person themselves.

After the tasks have been finished, it needs to be tested and verified before they can be marked as done. The test that needs to be done should be described in the issue description when the issue is added to the backlog.

Completed is the status when a task is tested and verified. There is one exception, in the task scheme diagram there is one state outside of the main grouping; "delayed". This state is earmarked for when the workload for the sprint is too much. The lower priority issues should then be moved to delayed, so that higher priority tasks can get the attention needed.

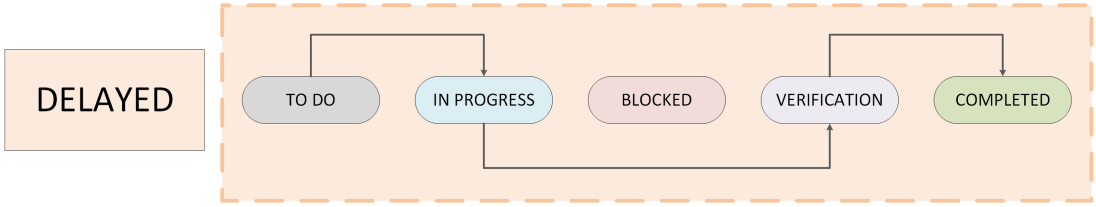


Figure 2.1: Workflow scheme for the project

2.2.7 Our model in retrospect

When looking back on the work model, we can confirm that an agile approach has the capability to increase the efficiency when spanning over longer time-periods. When a task or action was handled sub-optimally, the group noticed, took action, and implemented a solution within the next sprint. A sprint based system is said to be best developed for computer engineering teams, yet an agile model based on Scrum works well for a multidisciplinary team. The ability to partly deliver product over multiple iterations removes some of the problems of not delivering a product. Contrary to the waterfall model, if a deadline will not be met there is often already an underlying product. Perhaps with less features but nonetheless a product. Dividing up tasks into small periods of one to two weeks also helped manage what each individual would be doing. Since validation and testing occurs for each sprint, no massive amounts of progress will be lost and it is easy to make changes to the planning on the go. When beginning the day, each member would check their mail, time-schedule, and Jira board. Within five minutes, each member would be able to have mapped out their entire day.

2.3 Planning

When praising our agile model it is important to highlight the planning. The intertwined connection between proper planning and a good model is key. Even though we opted not to go for all the planning implementations that Scrum uses, we still used some of the features.

In the start of the project the group did not have TFMC accounts, this made it a bit difficult planning with all the employees from TFMC. But through good communication and the use of Discord's planning feature, we kept a good account of when and where meetings would be held. In terms of deadlines the team started using the Jira software early on. With its premium features, it gave the ability to create Gantt-like diagrams and also set up sprints over periods of time.

Once the TFMC accounts were in place, we migrated to their Teams platform and created an Outlook-schedule. With these accounts we could keep track of when our advisor and other important personnel could be scheduled for a meeting. It also gave us a common schedule where each individual could put at what times they where busy/available. When planning a week looking at this schedule resulted in being immensely helpful.

2.3.1 Scheduled meetings

Each week had two recurring meetings. The first being our meeting with our internal advisor, and second being with TFMC.
In the meeting with out internal advisor we would explain our progress, problems, and plans ahead. He would give us insightful feedback on what he might see as problematic. Each meeting was recorded in a minutes of meetings, which in combination with the Outlook-schedule made it easy

going through past tasks. This gave us an overview of the process and the pacing of the product as a whole.

In the meetings with TFMC we met with our external advisor and other important personnel. The group presented the progress of tasks and raised the attention to questions and problems that had occurred during the process. In these meetings the team often got insightful views on the problems and ideas the TFMC had. Getting advice from a lot of different TFMC employees was a key part of our validation process during the projects life cycle.

2.3.2 Long-term planning

When planning for long periods of time we used the Jira software in combination with the Outlook-schedule. This gave us access to a roadmap as well as marking key dates. When creating the different presentations and reports, we tried some different approaches.

Initially, we planned the major milestones of the project. At this time, we had limited knowledge of which roads the technical part of the project would take us down. Therefore, we created an overarching plan that mapped out all major deadlines, and generated epics for the respective technical disciplines.

2.3.3 Roadmap

The interactive roadmap in Jira makes it easy to interconnect our long-term planning with the short-term sprints and Scrum board. The major long-term goals and milestones of our project are defined in an epic (See 2.2.4), where tasks and stories are derived to the project backlog. The epics and their sub-tasks can be linked at all levels, which in practice means that we can plan which tasks are depended on each other early on. This is a great help when prioritizing the backlog for future sprints.

The roadmap is visualized as horizontal block on a timeline, and each member can personalize his roadmap-view to only show the tasks assigned to him, or to his technical team.

2.3.4 Outlook

When using a system that includes both mail and scheduling, the integration between the two makes for a smooth transition. Meetings were easily planned, scheduled, and added to calendars. When creating a meeting in Outlook the app helps you schedule it when all invitees are available. This functionality is key when dealing with multiple attendees. With each member actively using the calendar, scheduling meetings becomes a seamless process.

2.3.5 Working hours

In the period before the bachelor project started, the group created a contract that involved expected work ethics, as well as working hours. In this contract everyone agreed that they could expect a minimum of 30 hours of work each week. 30 hours each week would entail a minimum of 540 hours during the process.

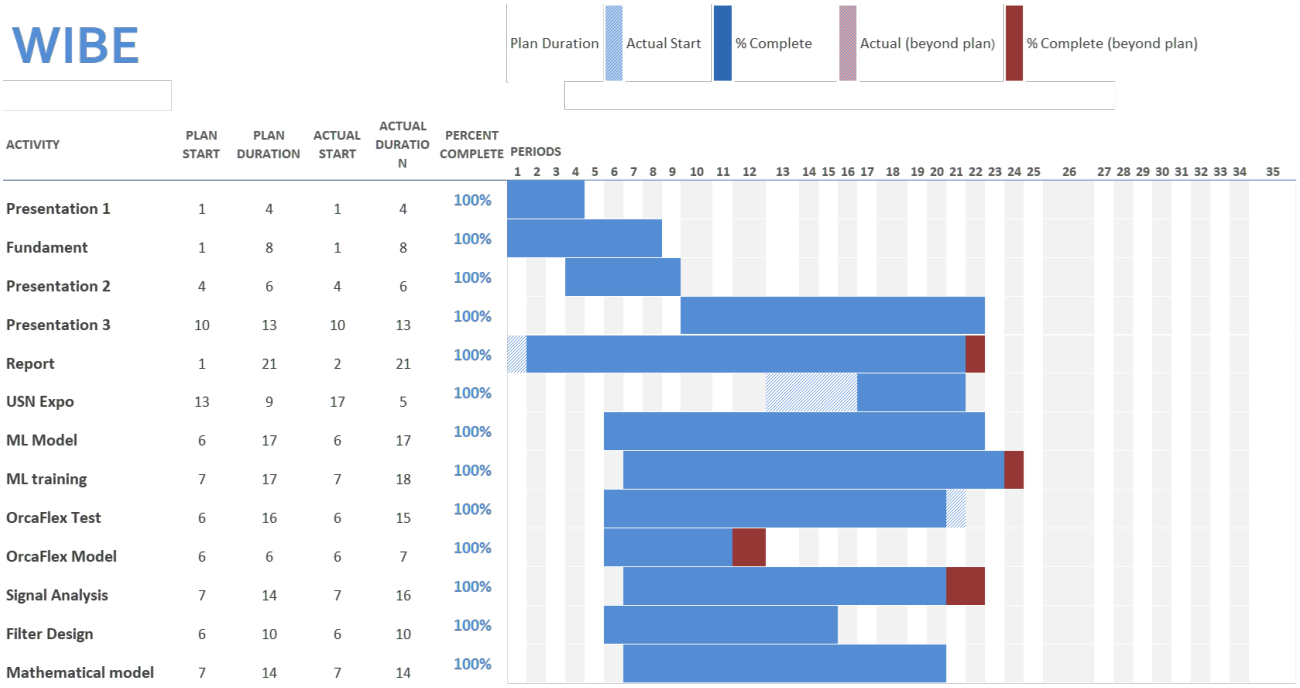


Figure 2.2: Gantt Diagram

2.4 Tools and materials

A short summary of the key tools used during the process.

2.4.1 Jira

We used the Jira software during our project. It comes with various tools to make administration of agile models easier.

2.4.1.1 Scrum board

Jira includes a digital scrum board that we customized to suit our needs. We created two views on the board. A view is a feature that allows us to sort tasks in specific ways, resulting in a great overview. Our main view groups tasks after their subordinated tasks. For example, all the tasks belonging to the epic Presentation 3, will be grouped together. The second view is a personal view, grouping tasks after assignee. Here the assignee sees all their current assigned tasks.

In addition we were able to implement the workflow as described in the workflow section (See 2.2.6).

2.4.1.2 Project pages

Using project pages we can create documents that is directly connected with other tools in the Jira software. We can for example directly link to tasks or other documents available in Jira.

2.4.1.3 Sprint planning

Sprints can be planned within the Jira software. When planning the sprint we can pick out tasks from the backlog, and assign them to different team members.

2.4.2 Overleaf

Overleaf is a web based LaTeX editor. It enables multiple editors on the same document. With LaTeX we can work on multiple files and compile them together to a large document. This makes the process easier to organize. We used Overleaf to setup a template to increase efficiency and quality in our follow-up documentation and minutes of meeting. In addition, we have written all documentation here.

2.4.3 Discord

For communication within the team we have been using Discord. With Discord we can use several chats dedicated to their own subjects. We can also pin important messages, which ensures that they do not disappear.

2.4.4 Microsoft Teams

We were using TFMC's Microsoft Teams group to share files and documents internally. Teams was used whenever we had a digital meeting. Teams as a platform is also used within TFMC to share internal and confidential files, and for us to cooperate with the project's stakeholders.

2.4.5 Google Drive

Once a week the documentation and information from Overleaf was backed up in a Google Drive folder. Towards the end of the process this backup happened daily.

2.4.6 GitHub

We used GitHub as version control for the software. With branching we are able to work with different parts of the code at the same time. We could also implement new functionality while still maintaining a stable build. When working with an agile approach this is very important. Jira supports a feature where you generate branches specific to the current task.

2.4.7 PowerPoint

We used PowerPoint for all of our presentations, and to create simple visualisations.

2.4.8 Visio

Visio was mainly used to create different types of visualisations.

2.4.9 Excel

Excel was used frequently throughout the whole period, to solve many different type of tasks. Our budget, requirements and risk analysis is some of the important tools setup in Excel. In addition, Excel has been used as an alternative to viewing datasets.

2.4.9.1 Time sheet

We have our time sheet setup in Excel, which had some useful functions for keeping a good overview and producing graphs.

2.4.10 OrcaFlex

OrcaFlex is a software for running dynamic analysis on offshore marine systems. It is tailor made to the purpose of our project, and we used it to both design parts of the model and the environment factors and forces impacting it.

2.4.10.1 O2-System

The TFMC provided O2-system simplifies the process when the need of running multiple simulations on the OrcaFlex model occur. This is definitely the case for our project, and this was therefore an essential tool.

2.4.11 Python

We used Python for all software related work.

2.4.11.1 Python libraries

Matplotlib

We use Matplotlib for most of our visualizations. John D. Hunter, the lead developer for Matplotlib wrote that, *"Matplotlib is a 2D graphics package used for Python for application development, interactive scripting, and publication-quality image generation across user interfaces and operating systems"* [11].

Pandas

Pandas is an open source data analysis tool [12] [13]. We use Pandas to import, manipulate, and save the data we work with. A Data Frame (DF) is a data structure provided by Pandas, it is a two-dimensional data matrix sorted with labelled rows and columns. The DF also has plenty useful in-built functions [12].

SKlearn

SKlearn is an ML Application Programming Interface (API) that includes implementations of the most common ML algorithms. It also comes with various tools for preprocessing data [14].

PyTorch

PyTorch is the ML framework that have been used throughout the project. It comes with a lot of the groundwork done for creating models of neural networks. By inheriting their module class we can build and design an Artificial Neural Network (ANN) to suit our exact needs. It also comes with Compute Unified Device Architecture (CUDA) support that allows us to use GPU for network training. [15]

Numpy

Numpy is a numeric mathematics module for Python. It includes a lot of handy functions for numeric calculations, and works well together with the previously mentioned APIs like PyTorch and Pandas.

Scipy

Scipy is a collection of mathematical algorithms and are built on the Numpy extension package.

Sympy

Sympy is a library for symbolic computations, and has been used to perform large calculations and to verify hand calculations.

2.4.12 Doxygen

Doxygen is a tool for generating documentation from source code. The tool allows us to write documentation for functions and classes as we program them. The tool outputs a page with an overview of classes, their functions, and the parameters. It also allows us to add code snippets that show examples.

2.4.13 UML

"*UML is a visual language that can be used in developing software systems. It is a specification language.*" [16, p.8]. The computer team found it helpful to use UML models to speed up the development process, and to gain an overview of complex systems.

2.4.14 MATLAB

MATLAB is a tool we used to analyse data in regards to signal processing, statistics and ML. Numerous books are written on filter design and signal analysis for implementing them into the MATLAB coding language. This makes MATLAB the preferred software for testing and building different kinds of programs.

2.5 Operation

There are many historical operations that were available to use when working on this project. This section outlines the operation we chose, and why we chose it.

2.5.1 Operational information

TFMC recommended that we should focus on a drilling operation performed on the Askeladd J-1 (NO 7120/8-J-1) well, located in the Snøhvit field. Snøhvit is a gas field in the Barents sea as shown in fig. 2.3.

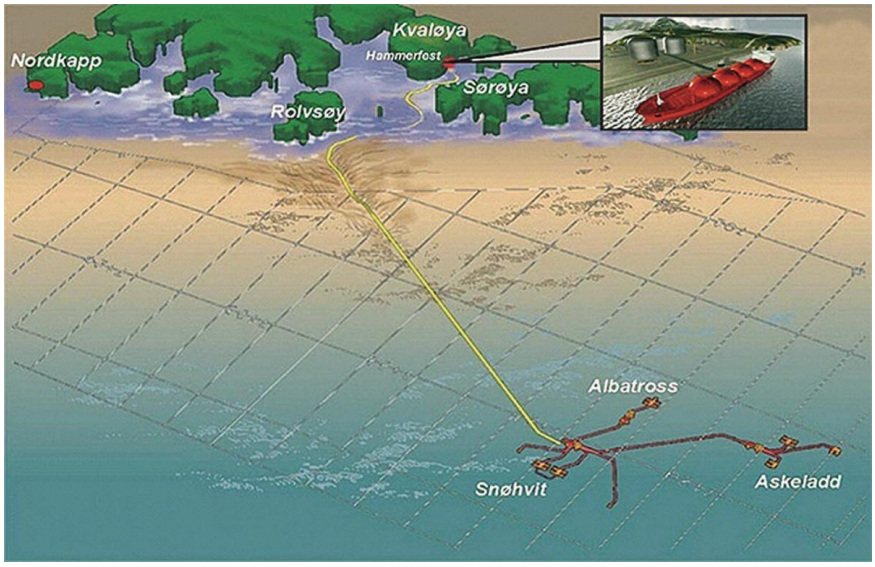


Figure 2.3: Snøhvit field

The operation is part of a series of operations with the Deepsea Atlantic (DSA) rig shown in fig. 2.4 - a semi-submersible rig owned by Odfjell Drilling and operated by Equinor.

Figure 2.4: Deepsea Atlantic semi-submersible rig

2.5.2 Askeladd J-1 operation

The operation was chosen because it had a proven a record of good data, making it an excellent foundation for training a ML model for our purpose. The key operational information is listed in table 2.2.

The operation was performed with a Well Access Management System (WAMS) equipped on the stack.

Table 2.2: Askeladd J-1 operational information [4]

Field:	Askeladd North
Well name:	NO 7120/8-J-1
Water depth:	270m
RFJ:	RFJ-1
Customer:	Equinor
Stack name:	BOP1
Stack-up:	BOP on WH
Start date:	26.12.2019
End date:	23.01.2020

2.5.3 Askeladd J-1 stack-up

The stack-up for a drilling operation consists of the Upper Flex Joint (UFJ) connected to a tensioner system above the water line, connected to the MR through the UFJ. The MR consists of several riser joints, both standard and buoyancy joints. The buoyancy joints are specialized riser joints that provide uplift and reduce its submerged weight. The number of standard and buoyancy joints differ from operation to operation based on the environment.

The MR is connected to the stack through the Marine Riser Adapter (MRA) and the Lower Flex Joint (LFJ). The RFJ is mounted on the outside of the LFJ and its purpose is to counteract forces to minimize the BM acting upon the WH.

The stack consists of the Blowout Preventer (BOP) and Lower Marine Riser Package (LMRP) and is connected to the WH through the WH connector. See fig. 2.6 for the full stack-up or fig. 2.5 for a zoomed-in stack.

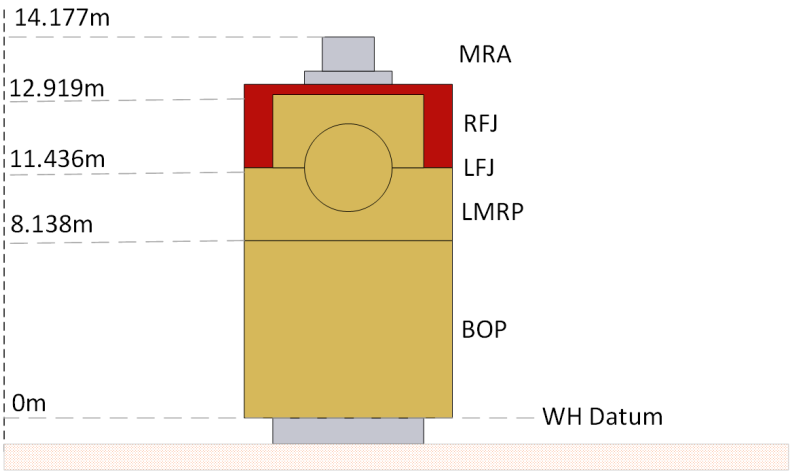


Figure 2.5: Askeladd J-1 BOP stack (not to scale)

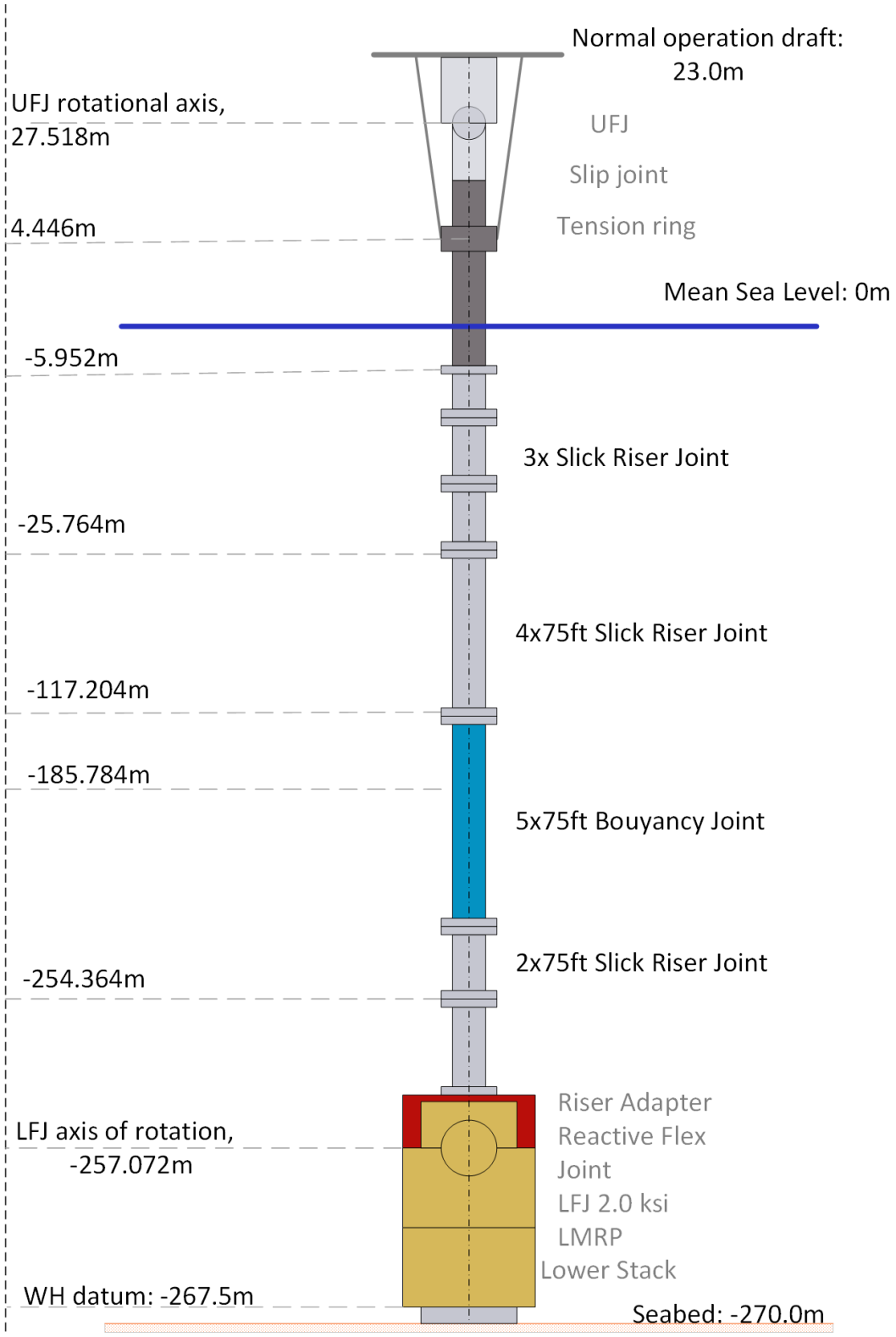


Figure 2.6: Askeladd J-1 stackup (not to scale)

2.6 Data

The data listed in this section is the data that was received externally, from TFMC, or their clients. It is the most important external input to our models.

2.6.1 Historical sensor data

The historical sensor data we had access to are limited to the Askeladd operation, which is described in table 2.2. There is an extensive amount of data exported from the Crossover Box (COB) and SMU sensors, the parameters that we have used in our project are outlined in table 2.3. The RFJ status is an important parameter because the system behaves differently when it is turned on or off. The status of the RFJ for the Askeladd J-1 operation is shown in table 2.4

Table 2.3: Historical sensor data

Parameter	Tag	SMU1	SMU2	SMU3	COB1	COB2
Time (10Hz)	time	Yes	Yes	Yes	Yes	Yes
Acceleration x-dir	accelx	Yes	Yes	Yes	-	-
Acceleration y-dir	accelx	Yes	Yes	Yes	-	-
Acceleration z-dir	accelx	Yes	Yes	Yes	-	-
Rotational velocity x-dir	rotvelx	Yes	Yes	Yes	-	-
Rotational velocity y-dir	rotvely	Yes	Yes	Yes	-	-
Inclination x-dir	inclinx	Yes	Yes	Yes	-	-
Inclination y-dir	incliny	Yes	Yes	Yes	-	-
Bending moment x-axis	bendingmo-mentx	-	-	-	Yes	Yes
Bending moment y-axis	bendingmo-menty	-	-	-	Yes	Yes
Strain DWS1	strain1	-	-	-	Yes	Yes
Strain DWS2	strain2	-	-	-	Yes	Yes
Strain DWS3	strain3	-	-	-	Yes	Yes
Strain DWS4	strain4	-	-	-	-	Yes

Table 2.4: RFJ activity

Start time	End time	Status
26.12.19 15:00	28.12.19 09:00	On
28.12.19 09:00	28.12.19 09:45	Off
28.12.19 09:45	31.12.19 08:45	On
31.12.19 08:45	01.01.20 05:40	Off
01.01.20 05:40	02.01.20 07:22	On
02.01.20 07:22	23.01.20 14:00	Off

2.6.2 Historical wave data

The historical wave data we have used is Equinor hindcast data, which is validated historical weather and ocean data. The hindcast data that was chosen is statistical, and gives data points on several parameters with a frequency of three hours.

For our purpose, modelling the forces acting upon a WH through the MR and stack, the parameters Significant Wave Height (Hs) and Spectral Peak Period (Tp) is enough to represent the magnitude of the net forces acting upon the WH with an acceptable accuracy.
In addition, we have used the mean wave direction to model the direction of the waves.

2.6.2.1 Significant wave height

The significant wave height is the highest third of the waves, and the parameter represent the average height of the highest waves in a wave group well. [17].
It is defined as:

$$H_s = \frac{1}{N/3} \sum_{i=1}^{N/3} H_i \tag{2.1}$$

Where N is the number of individual wave heights, and H_i is a series of wave heights ranked from highest to lowest. [18]
For non-breaking waves the spectral significant wave height is [17]

$$H_s \approx H_{m_0} = [\bar{E}/g\rho]^{1/2} \tag{2.2}$$

Where $\bar{E}$ is the average wave energy, g the gravitational acceleration constant and ρ the density of the seawater. [17]

2.6.2.2 Peak wave period

The peak wave period is the wave period with the highest energy. As a rule of thumb, it can be defined as:

$$T_p \approx \frac{1}{f}\sqrt{H_{m_0}} \tag{2.3}$$

where $f = 1/T$ is the wave frequency for a time series of individual waves, and H_{m_0} is the spectral significant wave height (2.2)

2.6.2.3 Mean wave direction

The mean wave direction, θ_m, is defined as the mean of all the individual wave directions in a time-series representing a certain sea state. [17]

Chapter 3

Risks & Requirements

In this chapter the process of ascertaining risks and consequences is in focus. Risk analysis and management gives the opportunity to be aware of potential pitfalls and hindrances we might face during the project. It is also put forth how requirements were derived and the variation in validation the group took.

3.1 Risk analysis

Risk analysis is about identifying potential risks for the project. A risk analysis has two factors, how likely it is to happen, and how large the consequence is. With a good risk analysis the risks with a high value for both parameters can be identified, and potentially managed.

		Consequence				
		Negligible 1	Minor 2	Moderate 3	Major 4	Catastrophic 5
Likelihood	5 Almost-Certain	Medium 5	Medium 10	High 15	High 20	High 25
	4 Likely	Low 4	Medium 8	Medium 12	High 16	High 20
	3 Possible	Low 3	Medium 6	Medium 9	Medium 12	High 15
	2 Unlikely	Low 2	Low 4	Medium 6	Medium 8	Medium 10
	1 Rare	Low 1	Low 2	Low 3	Low 4	Medium 5

Figure 3.1: Risk analysis matrix

We used a risk matrix with scores on each parameter between 1 and 5. Ranging from "rare" to "almost certain" in likelihood, and "negligible" to "catastrophic" in consequence. As seen in fig. 3.1 the combination of the two results in a risk level from low to high.

The risk of undiscovered major bugs or errors in the delivered product is the only risk that was rated high in our analysis. When performing the risk analysis, we could not predict the final progress of the project. The best case scenario being that a model which could be used in the real-time fatigue monitoring was delivered. Because of the dynamic linking in Python, bugs can easily remain unidentified. This could for example be a bug drifting over time, making estimations less and less reliable. Or simply a bug that crashed the fatigue analysis system.

Wrong placement of sensors in the dynamic simulation is one of many medium risks we identified. It is an error that is difficult to identify after the mistake is made.

Another risk worth mentioning is that of not being able to reach the desired goal within given time. Our most important goal is to train the machine learning model on the simulated data. That goal might not be reached if too much time is spent on preprossessing the data, working with the neural networks, or creating the dynamic simulations.

For the full risk analysis (See A.5).

3.1.1 Risk management

When the risks are identified they can potentially be mitigated. As the project is time critical, we need to prioritize between the risk level and the time it takes to mitigate the risk. We decided not to focus on the high level risk of unidentified bugs until it was certain that we would reach a part of the project, where we would build code for a critical system.
Even if we had done measures to reduce the probability of bugs, it is a common saying that it is easier to prove the existence, rather than the absence, of bugs.

Backup is a necessity that mitigates several risks. A backup is taken of the Overleaf project every Friday. In the last phase of the project, backups were taken every day. The code is accessible on cloud via GitHub, and in addition, each computer engineering student have their own local copy. The simulation is located on a server, meaning that if one of the mechanical engineering student's laptop fails, it will not cause loss of simulation data.

3.2 Project requirements

Requirement table

Each table contains one overarching requirement as seen in fig. 3.2. They are sorted as project (PJ), Computer Eng. (CE), Electrical Eng. (EE), and Mechanical Eng. (ME). Requirements have three levels of importance: LOW, MEDIUM, and HIGH, and they will be prioritized in that regard. Our requirements can have sub-requirements, for example requirement RQ-CE-1 have the sub-requirement RQ-CE-1.1.

Criteria

The second part is the criteria, which serve the purpose as check-points to verify the requirement before completion. This was done in an attempt to minimize the amount of tables needed without affecting the quality of our requirements themselves. The criteria have two levels of necessity: "must" and "optional". All "must" criteria have to be validated before the requirement can be marked as complete. We found this to be a well suited system which covered our needs as a multidisciplinary project.

Testing

We also have a test table associated with each requirement. This was used to log when a criteria status changed.

R-ID	Requirement description	Importance	Status	Source
RQ-PJ-1	The product will use an Artificial Neural Network (ANN) as an indirect method to find the bending moment on a subsea wellhead.	HIGH	DONE	TFMC
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-PJ-1-C	The accumulated fatigue calculated from predicted data should follow the trend and rough magnitude of the historical fatigue.	must	DONE	09.05.2022
T-PJ-1-C	The model should be able to predict the bending moment when the RFJ is enabeled.	must	DONE	09.05.2022
T-PJ-1-B	The prediction is done without the COB2 sensor.	must	DONE	05.05.2022
T-PJ-1-B	The prediction is done without any of the DWS sensors.	optional	DONE	05.05.2022
T-PJ-1-A	RQ-CE-1 must be approved.	must	DONE	29.04.2022

Figure 3.2: Main project requirement

3.3 Defining the requirements

Requirements are defined in meetings with our main stakeholder, TFMC. They talked about what they wanted and needed, and we proposed a list of requirements reflecting their needs that we felt we could deliver. They were confirmed and made into tables as previously described.
We also defined our own requirements around what we needed to build regarding the framework around the ML part of the task. This was also verified by TFMC.

3.3.1 Validation

Our requirements was in a fleeting state during our project, a bi-product of the highly theoretical nature of our task. They sometimes changed during the weekly meetings with TFMC. These changes was due to new insights gained as the progress was discussed. However, we did have a main goal to work towards the whole time. To make sure we stayed on the right track, we had weekly status meetings with TFMC (See 2.3).

3.3.2 The end product

Our main task is to use ML, filters, and simulations to develop an indirect method to measure the BM acting upon a WH, which is used for WH fatigue calculations. This should be done due to the future removal of a COB sensor placed at the lower stack responsible for measuring strain around the WH, which is essential for measuring fatigue. The end product can in other words be seen as a way to replace the physical COB sensor with the indirect method we develop.

The way to get there was in many ways up to us, as long as the goal was kept in mind. The path was formed along the way, influenced by weekly meetings with our stakeholders, internal advisor, and discussions within the group.

Chapter 4

Design

This chapter includes the main body of technical work performed in the project. The design for all the different models have been an iterative process, and this chapter includes the iterations of note within each relevant section. Due to the multidisciplinary nature of the project, the chapter does not read chronologically. Some of design iterations outlined here may be the result of tests and results described in their respective chapters.

4.1 Global and inertial axis

To be able to tell if the acquired sensor data can be fused and turned into understandable and readable data, we need to understand how the sensor data can be converted from the inertial frame to the global frame.

Distinguishing between global and inertial axis frames is a known problem, especially in the maritime and aerospace industry [19] and [20, p. 151].

4.1.1 Orientation

Before diving into the filter setup, we need to have some basic understanding about how orientation affects the sensors. SMU sensors measure displacement, velocity, and acceleration compared to its own inertial frame. To get a better understanding of what this really means we can take a look at fig. 4.1

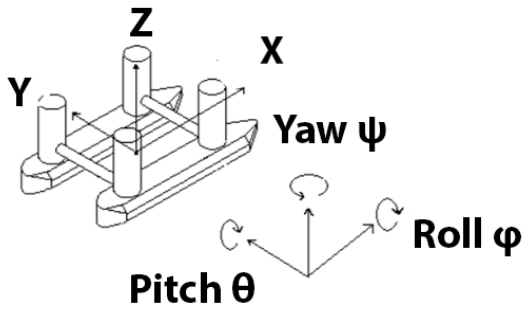


Figure 4.1: Orientation based on local axis

We can visualize the sensors orientation with three axes **XYZ** and their respective rotation around these. This is usually called:

$$\begin{aligned} Roll &= \phi \\ Pitch &= \theta \\ Yaw &= \psi \end{aligned}$$

Take notice that these parameters move with the axes. Using fig. 4.1 as an example, it can be thought of as tilting our head to the right or towards the positive x-axis. If we look at the orientation around the same axis, nothing has changed. This is because the roll around the y-axis is the only thing that has changed. Therefore, we need a way to describe the rotation felt by the sensor in its own inertial frame converted over to the global axis, where displacement or tilt in the x-direction is measured by the respective sensor.

4.1.1.1 Euler angles

Leonard Euler was one of the first people to grasp this problem of rotating a body in free space [21]. He constructed something we use today known as Euler angles and uses a rotation matrix to construct the new angles based on a known rotation. In (4.1,4.2,4.3) the rotational matrices are presented

$$\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{4.1}$$

$$\mathbf{R}_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{4.2}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

The rotation matrix creates a $\mathbb{R}^3$ matrix that could be combined for finding the combined rotation in three planes (4.4).

$$\mathbf{R}_{xyz} = \begin{bmatrix} cos\theta cos\psi & cos\theta sin\psi & -sin\theta \\ sin\phi sin\theta cos\psi - cos\phi sin\psi & sin\phi sin\theta sin\psi + cos\phi cos\psi & sin\phi cos\theta \\ cos\phi sin\theta cos\psi + sin\phi sin\psi & cos\phi sin\theta sin\psi - sin\phi cos\psi & cos\phi cos\theta \end{bmatrix} \tag{4.4}$$

4.1.1.2 Quaternions

Quaternions is another way to visualize rotation in free space. The difference here from Euler angles is the extraction of the directional vector and combining them with the xyz axis named ijk in quaternions. We can describe the rotation with (4.5)

$$\mathbf{q} = (a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k}) \tag{4.5}$$

Where a_1 is the scalar part of the vector and b_1, c_1, d_1 is the rotational vectors. It will also be useful to know the quaternion multiplication rules seen in (4.6)

$$\mathbf{q} \times \mathbf{p} = (a_1 + b_1\mathbf{i} + c_1\mathbf{j} + d_1\mathbf{k})(a_2 + b_2\mathbf{i} + c_2\mathbf{j} + d_2\mathbf{k}) \tag{4.6}$$

That gives us the same numbers as if we would multiply them as usual for imaginary numbers. Therefore the multiplication becomes (4.7)

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{i}\,\mathbf{j}\,\mathbf{k} = -1 \tag{4.7}$$

We will not go into depth in what quaternions are and they can be further read about in articles such as [22].

The reason why we need to address these two kinds of rotational transformations is because most Attitude and heading reference systems (AHRS) uses quaternions, and not Euler angles, to estimate rotation. The reason for choosing quaternions over Euler angles is the phenomena known as Gimbal lock [23], where we will lose a degree of freedom caused when two rotational axes line up in the same location.

A great feature about this project is that the system only experiences angles upwards of $\pm\,3^\circ$. It can therefore be assumed that the Gimbal lock will not occur, and linearization can be used on much of the system.

Since quaternions does not tell us much when looking directly at them, the formula for converting to Euler angles is shown in (4.8)

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{pmatrix} \arctan\frac{2q_2q_3 - q_0q_1}{2q_0^2 - 1 + 2q_3^2} \\ \arctan\frac{2q_1q_3 - q_0q_2}{\sqrt{1-(2q_1q_3+q_0q_2)^2}} \\ \arctan\frac{2q_1q_3 - q_0q_3}{2q_0^2 - 1 + 2q_1^2} \end{pmatrix} \tag{4.8}$$

4.1.2 Sensors

The sensors used on the configuration contain a total of three SMU sensors, and a set of seven Deep Water Strain Sensors (DWS) sensors mounted in two different locations. We can find documentation about the SMU sensors in [1] and the DWS sensors in [24].

4.1.2.1 Gyroscope

The gyroscopes found in MEMS are typically created to measure m/s or rad/s. They are also built to operate after the same properties as described in 4.1.1.

The gyroscope can also be used to measure inclination once it has been integrated. The problem with integrating a sensor is the drift that accumulates over time from noise and biases. There are also other problems with the gyroscope, like the drift caused by earths rotation around its own axis that is only 7.27×10^{-5} rad/s [25].

During the project, the opportunity never came to measure the biases in the sensors. As a backup to the SMU sensors, similar motion sensors from Bosch [26] were introduced. The biases is taken from 600 samples when the gyroscopes were laid flat on a table and the normal distribution curve is put on top in fig. 4.2. We can see their statistical parameters in table 4.1.

Table 4.1: Gyroscope statistics, normal distribution.

Parameter	ϕ	θ	ψ
σ	0.0915	0.1037	0.0759
$\bar{y}$	0.0105	0.0045	0.0047
σ^2	0.0084	0.0107	0.0058

These properties could be beneficial in the making of a Kalman filter because they represent the different biases from the MEMS.

4.1.2.2 Accelerometer

The accelerometer is a sensor that is used to measure gravity and acceleration, and usually gives the acquired data in g or m/s^2. We can also do a double integration of these sensors to find displacement or angle. The problem with the accelerometer is that it can not distinguish between acceleration due to gravity and acceleration due to displacement.

As mentioned in (See 4.1.2.1). The stack experiences very static movements and can therefore be used to estimate the orientation with (4.9)

$$\begin{bmatrix} \phi \\ \theta \end{bmatrix} = \begin{pmatrix} \arctan \frac{-Acc_x}{\sqrt{Acc_y^2 + Acc_z^2}} \\ \arctan \frac{-Acc_y}{Acc_z} \end{pmatrix} \tag{4.9}$$

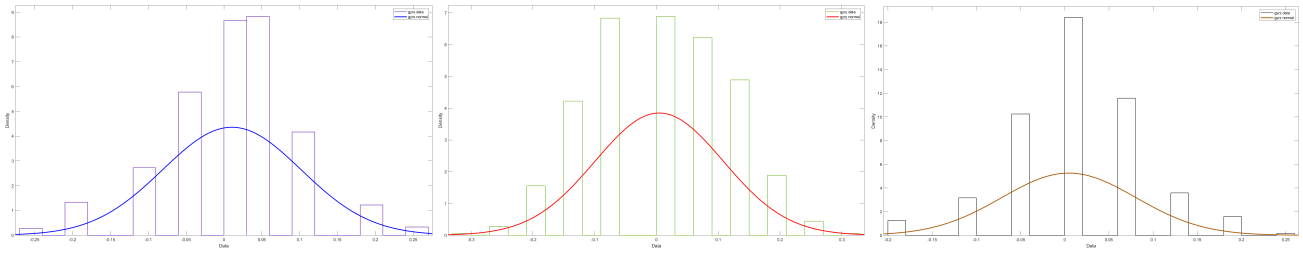
Where Acc denotes the acceleration due to gravity. Note that the yaw can not be determined with only the accelerometer, but can be found with an addition of a magnetometer. The yaw denotes the torsional movement and is not a factor that is necessary for finding the indirect BM.

In fig. 4.3 we can see the normal fit and their biases for the accelerometer when it is laid flat on a table for 600 samples. Their statistical parameters can be seen in table 4.2

Table 4.2: Accelerometer statistics, normal distribution.

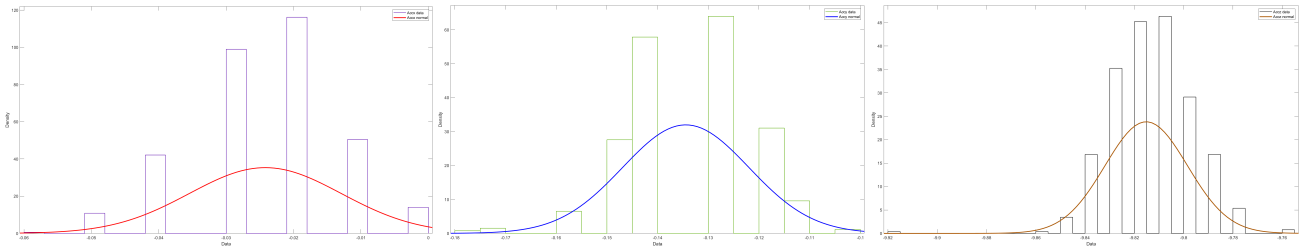
Parameter	x	y	z
σ	0.0113	0.0125	0.0167
$\bar{y}$	-0.0242	-0.1344	-9.8151
σ^2	0.0001	0.0002	0.0003

As mentioned in regards to the gyroscope data, these parameters could be beneficial for the implementation of a Kalman filter. Take note that the x-axis and y-axis for all the sub figures represent the data point values and the amount of repetitiveness respectfully.



(a) ϕ (b) θ (c) ψ

Figure 4.2: Gyroscope bias



(a) x (b) y (c) z

Figure 4.3: Accelerometer bias

4.1.2.3 Deep water strain sensor

The DWS sensor measures displacement due to increases or decreases in the total length of the sensor. It measures the changes in phase from a given signal and can therefore be seen as a linear variable differential transducer, and is not the same as the more familiar family of strain sensor known as the strain gauges.

4.1.2.4 Sensor accuracy

When using the SMU sensors to estimate the BM, their ranges and inaccuracies has to be established. From report such as [27] and [28] we can find their ranges and biases.

In table 4.3 their combined data and their inaccuracies can be seen.

Table 4.3: SMU and DWS Parameters

Transducer	Resolution	Accuracy	Non linearity	Uniqueness
Accelerometer	0.001 m/s^2	-	$\pm$0.05%	Yes
Gyroscope	0.01°/s	-	$<$0.1%	Yes
Inclination	$\leq$0.003°	Dy.($<$0.5°RMS), St.($\leq$0.003°)	-	No
DWS	0.25 μm/m	$\pm$2.5 μm/m	-	Yes

We will take this into consideration when designing our model, because even though we are trying to find the indirect BM, the measured data can have some deviation from what actually happens at that point.

In fig. 4.4 the inaccuracies of the direct BM measurements can be seen from the strain sensors. The deviation expands to roughly 4500 Nm when scaled to match the BM and can from the figure conclude that the deviation is so insignificant that there are no question in trusting the reliability of the strain sensors when they are working .

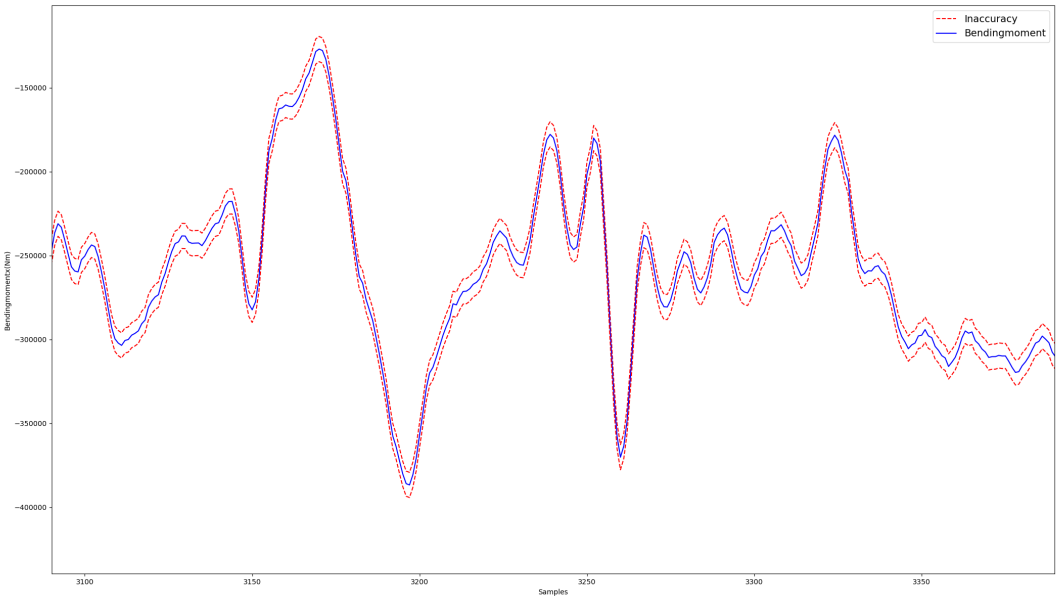


Figure 4.4: Inaccuracy in bending moment measurements

4.2 OrcaFlex-model

Our ML model needs realistic simulated training data, to achieve this we are in need of a finite element analysis software capable of running dynamic simulations.
The software must be able to set a specified environment, in our case a sea state. This is important as we wish to simulate certain historical days, and create new fictional scenarios. The 3D-model itself needs to have the capability to set defining parameters, to ensure that the model becomes accurate. Considering these requirements, the OrcaFlex software fits our purpose.

An engineer working on an efficiency study of the Reactive Flex Joint had created a model of the Askeladd J-1 stack-up in OrcaFlex. This is the first time this operation has been modelled in OrcaFlex, which helped us a fair bit. The initial plan was to create a model based an operation on the Johan Sverdrup field, and re-scale the inputs to fit our operation. The provided model was not created for our purpose, which means that we had to modify it to suit the need of our project.

The Askeladd J1 model was delivered to us in the O2-system (See 2.4.10.1), this meant that we had a template to build upon. It helped us a lot in the early stages. Especially since this is an internal software, meaning online guides are non-existent. The operator needs OrcaFlex knowledge to successfully use the O2-system. TFMC and Orcina provided us with great learning material, resulting in an effective learning period.

4.2.1 OrcaFlex model construction

The OrcaFlex model could be viewed as a long line stretching from a vessel at sea level, all the way down into the seabed. This line is split into many smaller components, and then into segments, grouped into sections. Each line section has a type which states what kind of physical properties it has, this could be viewed in fig. 4.5.

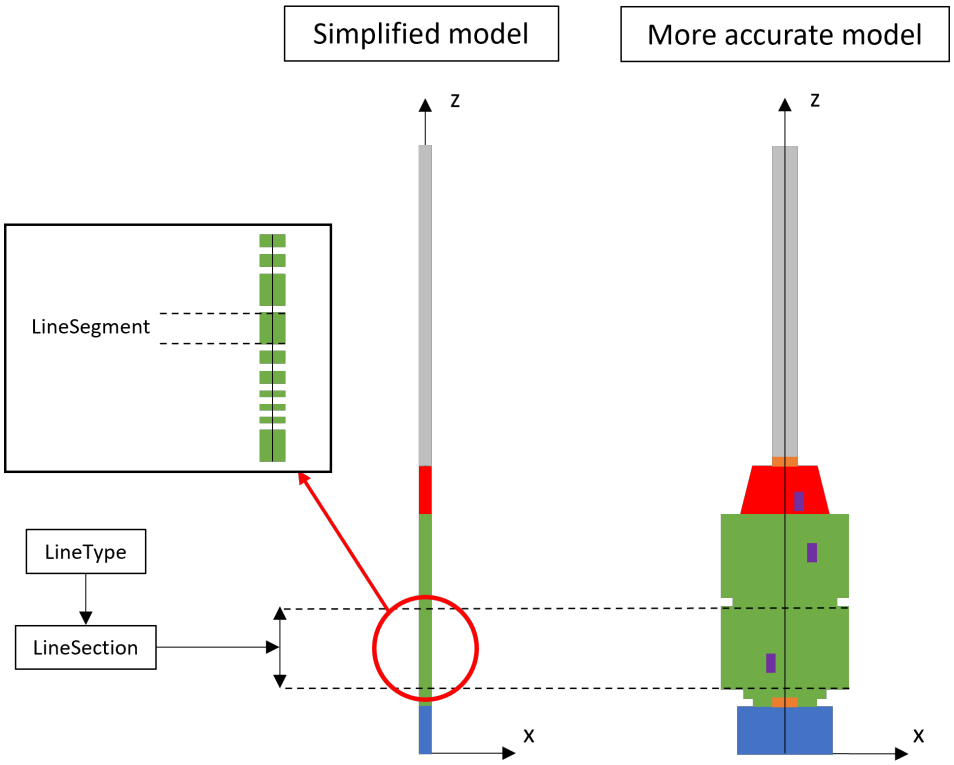


Figure 4.5: OrcaFlex model, line components

4.2.1.1 Line components

To increase accuracy of the output from the OrcaFlex model, some adjustments to the line components should be implemented. We changed those lines where the sensors are located. It was important to maintain the correct lengths from the original model.
These changes are adjustments to length, Target Segment Length (TSL) and adding or removal of line sections.

Target segment lengths

Changing the TSL of the line sections, affects the simulation time and results. Decreasing the TSL adds more line segments, which increases the accuracy of the model, but at the expense of simulation time. In terms of prioritization between the two, more accurate results had a higher degree of importance. It is essential to maintain the original length of the different line types, we did some calculations regarding this, as seen in appendix (B.3.1).

At all the SMU locations, we created small line sections at $0,01$m in their z-axis positions. The two COB senors received the same treatment, except that the line sections size was increased to $0,11$m. This was done to match the COB sensors actual measuring area, as seen in appendix (B.15).

Stress diameters

To be able to gather correct data from the COB sensors, we needed to specify the stress diameters for the line types where the COB sensors were located. The inner and outer stress diameter can be found in table 4.4.

Table 4.4: COB-details [5]

Tag	COB1	COB2	Units
Offset x	0	0	m
Offset y	0	0	m
Offset z	0	0	m
Sensors	3	4	pcs
Outer diameter	0.56	1.162	m
Inner diameter	0.48	1.131	m
Mounting distance	0.044	0.044	m
Sensor length	0.104	0.104	m
Sensor 1 offset x	0.28	0.581	m
Sensor 2 offset x	-0.14	0	m
Sensor 3 offset x	-0.14	-0.581	m
Sensor 4 offset x	-	0	m
Sensor 1 offset y	0	0	m
Sensor 2 offset y	0.242	0.581	m
Sensor 3 offset y	-0.242	0	m
Sensor 4 offset y	-	-0.581	m

4.2.2 Sensor placement

To test and calibrate the output from the sensors in OrcaFlex, they need to be placed correctly, and we must be able to extract the correct parameters from each sensor.

Strain sensors are mounted at the WH connector and MRA to monitor the BMs directly as shown in fig. 4.6. Furthermore, we can see that one motion sensor is placed above the RFJ while two are placed below at two separate locations on the stack. [1, p.4]

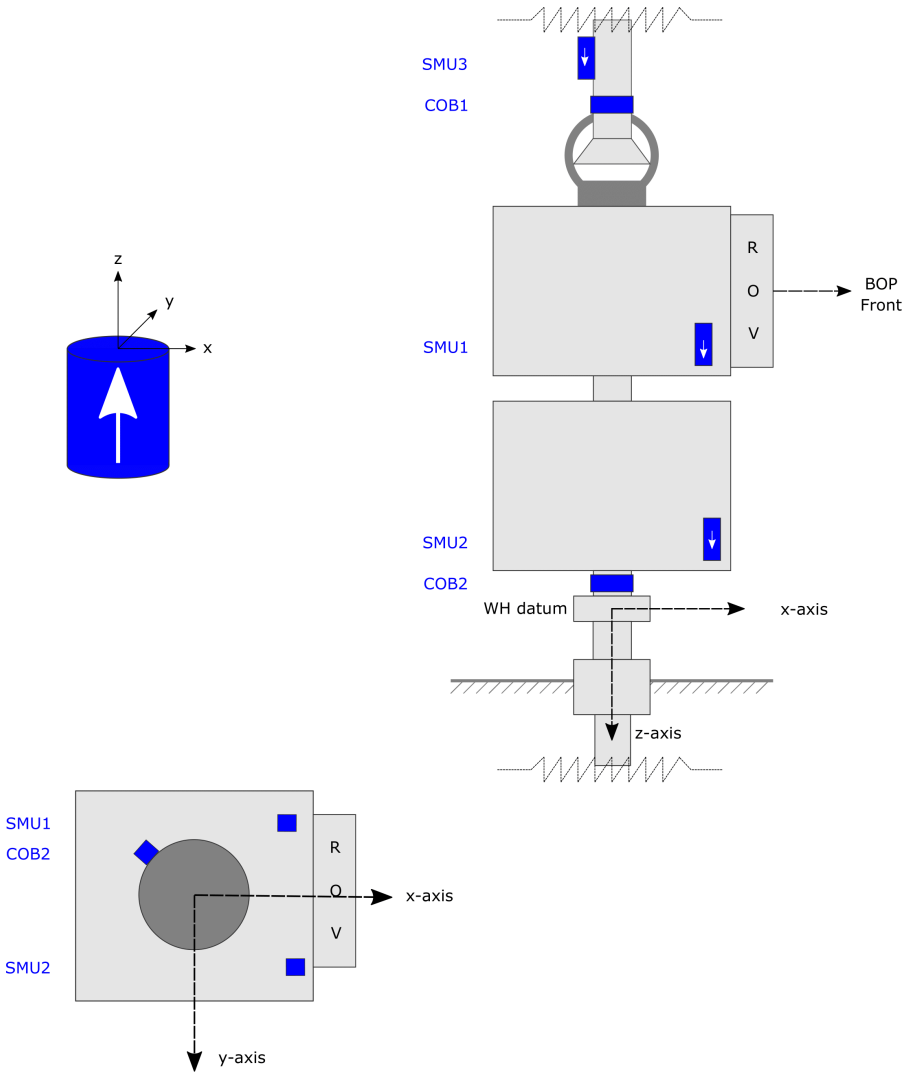


Figure 4.6: Stack with sensors [1, p.5]

4.2.2.1 Strain sensors

The DWSs are placed around the outer part of the hollow circular cross-sections on the WH connector and MRA. To evaluate the BM and tension in the cross section where the strain sensors are placed, at least three measurements are needed. It is common to use three or four equidistant sensors, which is illustrated in fig. 4.7. [1, p.6].

The COB1 sensor has a total of three DWS sensors and the COB2 sensor has four. These are equidistantly placed around the respective circular cross-sections. The sensors are described in further detail in section (See 4.1.2.3).

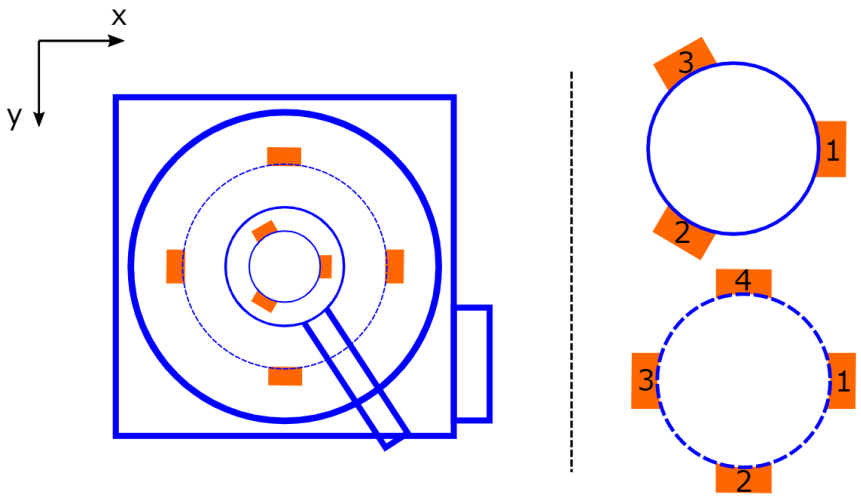


Figure 4.7: Strain sensor placements in xy-plane. [1, p.6]

Initial placement of strain sensors

The strain sensors had to be placed in the correct vertical distance on the outer part of the cross-section at same angles as in the historical operation. TFMC does not log the vertical position of the COB. We initially placed the COB sensors at the bottom of the MRA and right above the WH datum at the WH connector. We moved the sensors closer to their actual position later in the design process, as new information became available to us. These positions can be seen in appendix (B.2.2).

Changes due to axis calibration

The COB setup in the OrcaFlex model differs from fig. 4.7, which is similar to what the WAMS system is based upon. The DWS1 sensor must be defined in the same manner in OrcaFlex as it was in the actual operation. Its location has a direct relation to the measured BM. We found out through result analysis that the rig and the BOP had the same orientation. This was later confirmed by the rig owner. They stated that the rig and BOP always had the same orientation. If the BOP needs to be reoriented for a different operation, the rig must be rotated to compensate. This means that the port side on the rig and BOP are oriented in the same direction. In CAD models and schematics of the BOP, the subsea panel is located on the port side, which is defined as the y-axis. The x-axis was oriented towards the forward of the BOP. (See appendix (B.2.3, fig. B.18). The OrcaFlex model was then altered to match this, resulting in the COB sensors setup seen in fig. 4.8.

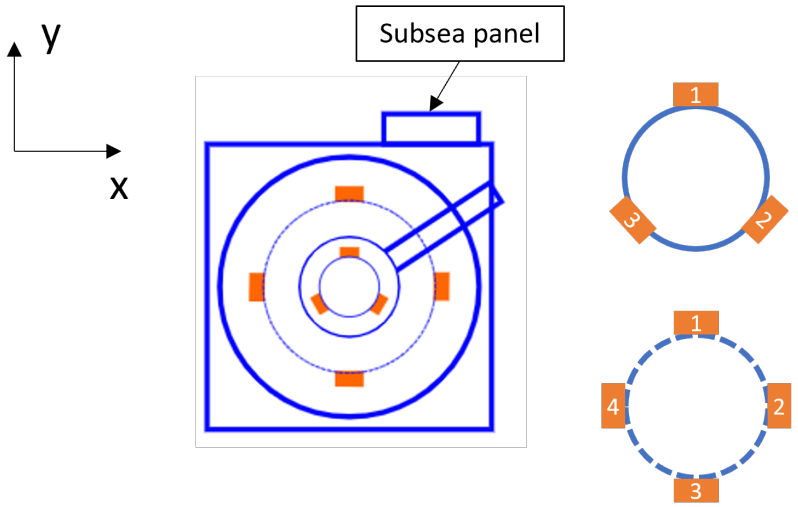


Figure 4.8: BOP axis orientation in OrcaFlex, based on fig. 4.7

Because of how the line components are constructed in OrcaFlex, we get a local BOP axis equal to the one seen in fig. 4.12c. Both the z and x-axis have their positive direction rotated with 180°. This needs to be accounted for in the O2-system when setting up the Result file (RES), as seen in table 4.5 and in fig. 4.9. The "outer" tag describes the sensor placement on the pipe, as these sensors are all placed on the outside of the circular cross-section.

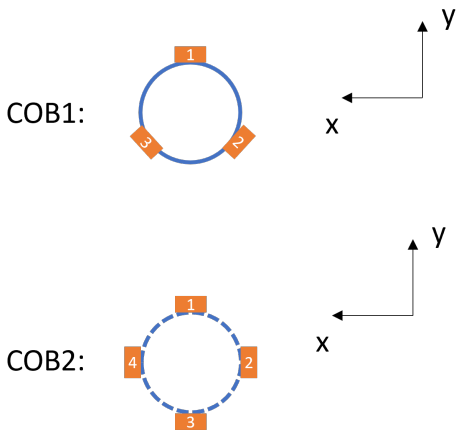


Figure 4.9: O2-System, COB sensors setup

Table 4.5: O2 system, COB sensor result file setup

Sensor	DWS1	DWS2	DWS3	DWS4
COB1	Outer90	Outer210	Outer330	-
COB2	Outer90	Outer180	Outer270	Outer0

4.2.2.2 Motion sensors

The SMU are placed in three locations in the system as shown in fig. 4.6, above the rotational axis of the LFJ, on the LMRP and on the BOP.

The sensors are described in detail in section (See 4.1.2).

Initial placement of motion sensors

The location of the SMU sensors was provided in the Askeladd J-1 operational documentation [4]. The values are listed in table 4.6, and the offsets in xyz-direction are relative to the vertical center axis of the MR for the x- and y-direction, while the values for offset in z-direction is relative to the WH datum as shown in fig. 2.5.

Our initial solution viewed the stack as one rigid body, therefore only the location on the z-axis for the sensors were used. Meaning that the sensors were placed directly on the z-axis.

Table 4.6: SMU Sensor Placement

Sensor	x[m]	y[m]	z[m]
SMU1	1.288	0.928	-8.188
SMU2	0.679	-0.754	-2.193
SMU3	-1.062	0.323	-11.682

New method for extracting SMU data

When we compared our initial solution to historical data, the SMUs acceleration in the z-direction had different motion behavior. Therefore, we found a new solution taking both the x and y location into account. This was done by creating new Buoy (6D) elements connected to the respective line, based on the z-axis location of the sensor. These buoy elements had negligible properties and had an initial position based on historical sensor location, as seen in table 4.6. However, both the z and x values needed to have their sign changed. This is due to the BOP local axis, as seen in fig. 4.12c.

4.2.3 Design not used in ML training

The changes to the axis orientation discussed in section 4.2.2.1, caused the sensors to end up in the wrong position and orientation, compared to the historical setup seen in fig. 4.10. This was unfortunately not discovered until after the simulations for the ML training was complete. The data from these simulations can be altered to follow these new changes (See 4.1.1).

4.2.3.1 SMU positions

All the simulations were run on the setup seen in fig. 4.11a. The sensors should be flipped over the horizontal x-axis and rotated with 90° against the clock, to be located in the same position as the historical setup. In terms of changes in the O2-system, this results in firstly having to swap the historical x and y values, seen in table 4.6. Because of the BOP local axis, we also need to change the sign on both the x and z values, to compensate for opposite positive axis direction. The result of these changes can be seen in fig. 4.11b, and by comparing it to the historical setup in fig. 4.10, we can see that the positions are now correct.

4.2.3.2 SMU orientation

The actual sensor orientation can be seen in appendix B.2.5 and B.20. From these figures it can be seen that fig. 4.10 matches the actual setup. By comparing the fig. 4.10 and fig. 4.11a, we can see that the orientation of the SMU sensors in OrcaFlex were not similar. The SMU axes need to be flipped over the x-axis, and then rotated 90° counter clockwise. However, the old simulations can be altered using the method described in section (See 4.1.1). To fix this in the O2-system, we need to change the orientation of the buoy elements representing the SMUs, these changes can be seen in table 4.7.

Table 4.7: SMU Orientation around axes

Iteration	y[deg]	x[deg]	z[deg]
Old, fig. 4.11a	180	0	180
New, fig. 4.11b	0	0	90

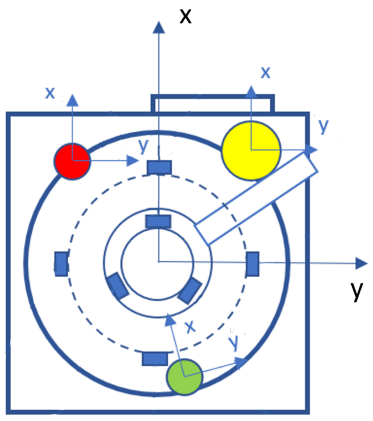
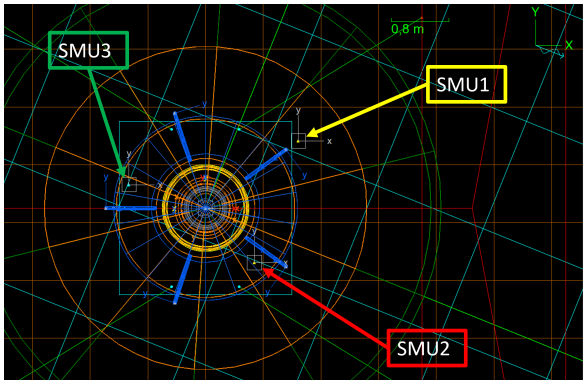
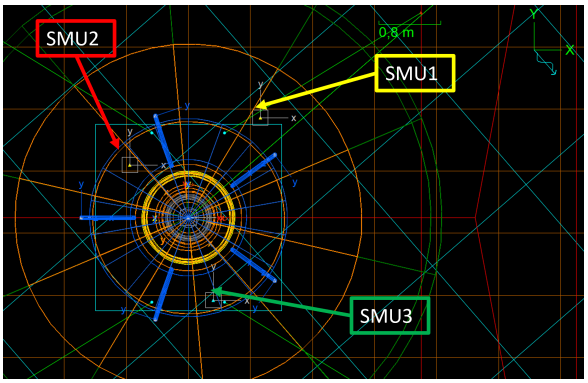


Figure 4.10: Historical position and orientation of sensors, oriented to match OrcaFlex.

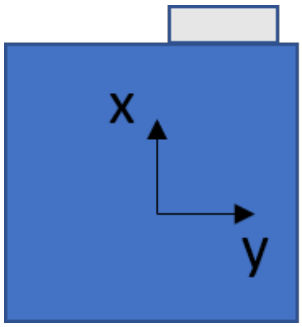


(a) Old

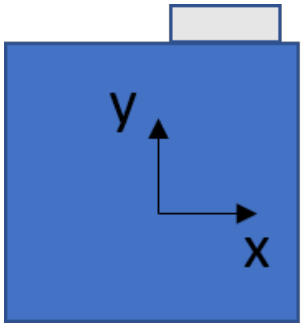


(b) New

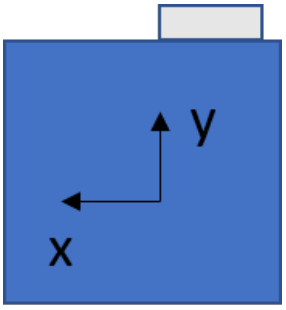
Figure 4.11: SMU position and orientation in OrcaFlex



(a) WAMS BOP, local axis



(b) Historical setup BOP, local axis



(c) OrcaFlex BOP, local axis

Figure 4.12: BOP local axis. *Port* is up, *Starboard* is down, *Forward* is right, and *Aft* is left.

4.2.4 Extracting sensor data

An important step towards reaching our goal of training the ML model on simulated data from OrcaFlex, was to find a method to extract the same parameters at the same locations as the SMU and DWS sensors. As previously discussed in 4.1.2.2, they are believed to be equal to the measurement.

4.2.4.1 Acceleration parameter

Acceleration is measured in xyz-direction and is represented in the historical data with a parameter for acceleration in xyz-direction for each SMU sensor.

A motion sensor fixed to the marine riser to measure acceleration normal to the local axis will measure the sum of two components: the true lateral acceleration plus a gravitational component proportional to the riser angle to the vertical direction. [29, p.13]

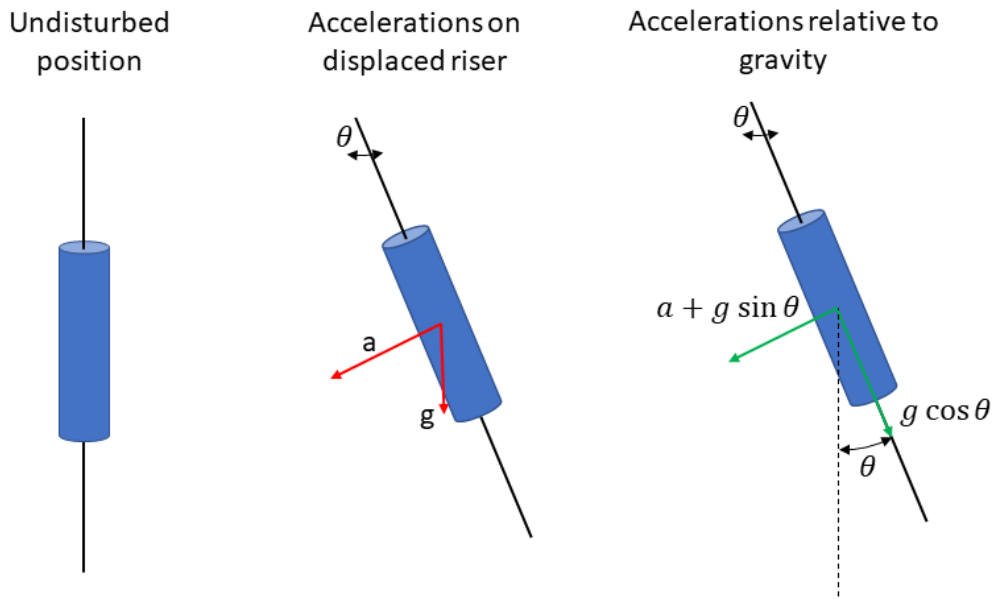


Figure 4.13: Source of gravity contamination

Assuming small displacements, a simple sinusoid mode shape and a single mode vibration at constant amplitude and frequency, the relative magnitudes can be quantified.
The lateral displacement of the riser is then given by: [29, p.13]

$$x = A \cdot \sin\left(\frac{n \cdot \pi \cdot z}{L}\right) \cdot \cos\left(\omega t\right) \tag{4.10}$$

where x is displacement normal to the riser axis, A is the maximum vibration amplitude, n is mode number, z is distance along the riser axis from one end, L is riser length, ω is frequency of vibration and t is time. [29, p.13]
Lateral acceleration is given by: [29, p.13]

$$a = -A \cdot \omega^2 \cdot \sin\left(\frac{n \cdot \pi \cdot z}{L}\right) \cdot \cos\left(\omega t\right) \tag{4.11}$$

The angular deflection in the riser (assumed small angles) resulting from vibration is given by: [29, p.13]

$$\theta = \frac{dx}{dz} = \left(\frac{A \cdot n \cdot \pi}{L}\right) \cdot \cos\left(\frac{n \cdot \pi \cdot z}{L}\right) \cdot \cos(\omega t) \tag{4.12}$$

A motion sensor fixed to the riser measuring acceleration normal to the riser axis will give a reading given by: [29, p.14]

$$R = a + g \cdot \sin(\theta) \tag{4.13}$$

For small displacements this reduces to:

$$R = a + g \cdot \theta \tag{4.14}$$

The only meaningful comparison that can be made between measurement and simulation is of accelerations including gravity contamination [29, p.15]. Based on this we will extract direct measurements of acceleration relative to gravity in x, y and z-direction from each SMU-sensor location in OrcaFlex.

4.2.4.2 Rotational velocity parameter

Rotational velocity is measured in xy-direction and is represented in the historical data with a parameter in the xy-direction for each SMU sensor.

OrcaFlex does not offer a rotational velocity parameter for direct extraction, meaning that an indirect method to extract the parameter we needed was necessary. Given that the rotational velocity is the time derivative of the angle (4.15)

$$\dot{\theta} = \frac{d\theta}{dt} \tag{4.15}$$

We know the frequency of our simulation, which implies that the rotational velocity can be found by extracting the dynamic angle at each time-step. This parameter gives us the position of the vector that defines the rotation from the static orientation to the instantaneous orientation [30]. We can find it by deriving the angle of the system by the frequency.

$$\vec{R} = [\vec{R}_x, \vec{R}_y, \vec{R}_z] \tag{4.16}$$

The rotation is about the direction of the vector $\vec{R}$ and has magnitude $|\vec{R}|$. [30]. The parameters we extract from OrcaFlex is $\vec{R_x}$ and $\vec{R_y}$ for all SMU sensors.

4.2.4.3 Strain parameter

Strain is measured at the specific location of each DWS sensor, and it is the strain due to bending which is measured by the sensors.

The parameter we wanted to extract was the strain at the given positions on the outer surface of the MRA and WH connector, as shown in fig. 4.7. OrcaFlex does not provide this parameter directly, which meant that we had to find an indirect method of extraction.

We found this method by extracting the bending stress at each location of the DWS sensors, and by using Hooke's law (4.17) to find the strain due to bending.

$$S_b = E\varepsilon \rightarrow \varepsilon = \frac{S_b}{E} \tag{4.17}$$

4.2.4.4 Python tools

The methods for extracting data from OrcaFlex had to be made applicable to be of value. We need to extract twenty-two parameters from the COB and SMU sensors with a 10Hz frequency for simulations lasting up to three hours, and to perform the conversion for strain (See 4.2.4.2) and rotational velocity (See 4.2.4.3).

The original plan was to extract results from the O2-system, but due to the delayed access we had to find another method in the first stage of the project. The temporary solutions was to manually extract results for each individual sensor from the OrcaFlex-software for each simulation.

When the access to use the O2-system was granted, we modified the tools to work with its output.

Converter

The purpose of the converter was to merge all result files into one Comma-separated values (CSV) file for each sensor, and to apply the conversions to the strain (See 4.2.4.2) and rotational velocity (See 4.2.4.3) to get the correct parameters that is comparable to the historical data (See 2.6.1).

In addition to making the OrcaFlex-data comparable to historical data, the converter combines the simulated data to make it easy to implement in the ML model.

Plotter

The purpose of the plotter was to get a quick overview of the behaviour of the sensors in the OrcaFlex simulations, to compare them to each other and to their respective counterparts in the historical data. This tool has helped us a great deal throughout the project, and it is part of the reason why we were able to calibrate and verify the behaviour of the sensors in OrcaFlex.

4.2.5 Cardinal direction

Implementing historical weather data into our OrcaFlex model is important, as this creates more realistic simulations for the ML model. There are quite a few available parameters in these datasets, we are using a ThorsetHaugen spectrum for our environment. [31] This spectrum provides a realistic representation of a sea state. While being a quite simplified model, as it is characterized by only two parameters, Hs and Tp, as discussed in section 2.6.2.1. [32]

The historical mean wave direction (TOTALDIRM), θ_m, is one of the other available parameters in the historical weather datasets. Implementing this would result in more realistic simulations. In the historical data this parameter is defined as degrees in compass direction, clockwise from north. The historical orientation of the rig, relative to cardinal coordinates can be seen in fig. 4.14. The rig had an orientation of 240° relative to north, as seen in appendix [B.2.4, fig. B.19]. In the actual operation the rig has its own local axis system, while the compass coordinates can be viewed as the global axis. The OrcaFlex model was constructed in such a way, that the global axis and the local axis of the rig were equal. Wave directions in OrcaFlex are based on the global axis, therefore we needed to rotate the cardinal directions. The compass is rotated to the historical orientation of the rig at 240° clockwise, as seen in fig. 4.15.

The Principal wave direction (WaveDir) in OrcaFlex varies between 0° and 360°, but it is defined as 0° to 180° and then from -180° to 0°. For example 200° is defined as -160° in OrcaFlex. Some simple calculations are needed, due to the compass orientation. Three periods of historical weather data can be seen in table 4.8. TOTALDIRM are solved by using (4.18), to fit the OrcaFlex definition for wave direction.

$$\text{WaveDir} = \begin{cases} \text{North} - \text{TOTALDIRM}, & \text{WaveDir} < 180° \\ \text{North} - \text{TOTALDIRM} - 360°, & \text{WaveDir} > 180° \end{cases} \tag{4.18}$$

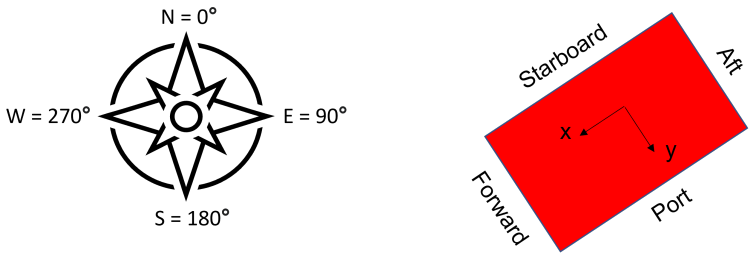


Figure 4.14: Historical rig orientation relative to cardinal direction

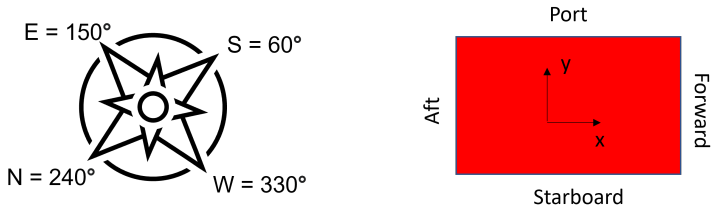


Figure 4.15: OrcaFlex rig orientation relative to cardinal direction

Table 4.8: Examples from historical weather data

Year	Month	Date	Time	Hs	Tp	TOTALDIRM
2019	12	30	3	3.3	12.43	290
2020	1	1	21	3.0	13.19	134
2020	6	29	3	1.0	4.43	42

4.2.6 Design for batch simulations

It is not enough to only implement the historical weather into our model's environment. For the simulation to be realistic we need to have an irregular sea state. This is done by altering parameters, as seen in table 4.9.

Table 4.9: Changes to implement an irregular sea state

Type of change	Parameter name	Original value	New value
Update	WaveNumberOfSpectralDirections	1	8
Add	WaveDirectionSpreadingExponent	-	6
Rename	WaveNumberOfComponents	-	+PerDirection
Update	WaveNumberOfComponentsPerDirection	200	100
Remove	WaveSpectrumMaxComponentFrequencyRange	0,05	-

Wave number of spectral directions

The parameter must be defined in the range 1 to 99. Described as the direction in which the wave progresses, measured positive counter-clockwise from the global X-axis when viewed from above. For example, 0° results in a wave travelling in the positive X-direction, while 90° means a wave movement in the positive Y-direction. The principal wave direction set by historical weather data is still dominant, more on this (See 4.2.5). A visual example of this parameter can be seen in fig. 4.16. [33]

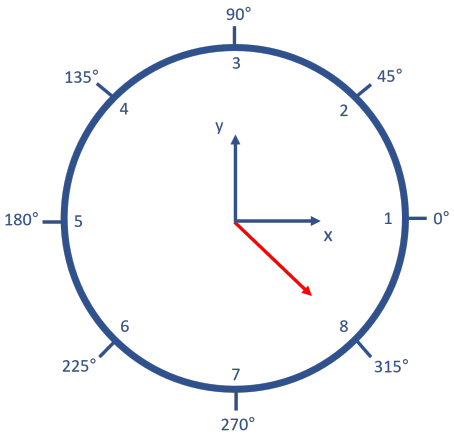


Figure 4.16: Auxiliary figure, number of wave directions = 8, principal wave direction = 315°

Wave direction spreading exponent

It is not enough to just specify static wave directions, to achieve irregularities in the sea state. A spreading exponent must be set, this has to be in the range 0 to 100. OrcaFlex uses a spreading exponent of $\cos^n$. Our OrcaFlex model used a directional spreading spectrum, as seen in (4.19)

$$S_d(\theta) = K(n) \cdot cos^n(\theta - \theta_p) \qquad for -\frac{\pi}{2} \leq \theta - \theta_p \leq \frac{\pi}{2} \tag{4.19}$$

Where $K(n)$ is a normalising constant, which changes based on the set spreading exponent, n. θ is the wave direction, which is randomly set based on the specified number of wave directions. For example in fig. 4.16, θ can be all the values on the outside of the circle. θ_p is the principal wave direction, which is the dominant angle for the waves. Determined by the parameter WaveDir from the Load case file (LCG) , set by TOTALDIRM. [33]

Wave number of components per direction

This parameter defines the number of wave components per direction. Since the OrcaFlex model is using a directional spreading spectrum (4.19). Is the number of wave components given per direction. Otherwise, it would be the total number of wave components for the wave train. [34]

Wave spectrum max component frequency range

"*Places an upper limit on the width of the frequency range represented by each wave component.*" [34]

4.3 Framework and preprocessing

A high level overview of the modules in our software solution can be seen in fig. 4.17. We ended up with four modules in our solution. In each module we have focused on the Bertrand Meyers Open-Closed Principle. It states that if a software system should be easy to change, it should be designed in a way where behaviour is changed by adding new code, rather then changing existing code [35, p.59].

Originally the *Manager* depended on the Agent. In the training phase the manager is our main component, and we want the main component to be stable. The agent will not be stable as we will be trying out different neural networks. Using terminology from [35] the agent will be flexible. If a stable component depends on a flexible component the flexible component will no longer be easy to change. Following the stable dependencies principle we fix this by making them both depend on an interface instead. [35, p 124-125]

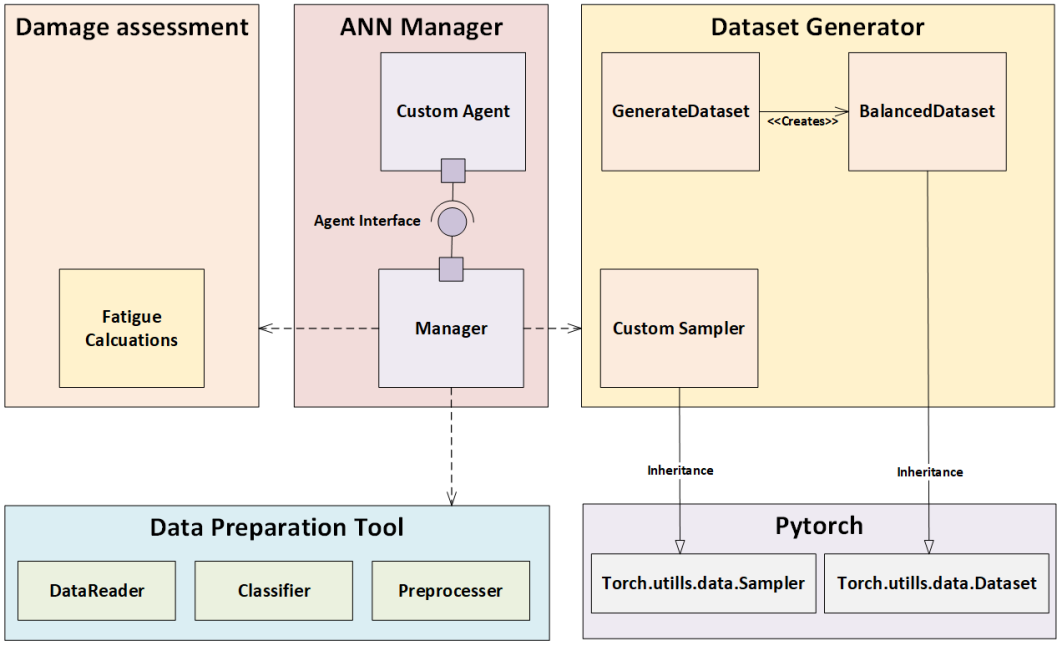


Figure 4.17: Architectural overview of final solution

4.3.1 Data preparation tool

The data preparation tool is a component we built to prepare the data before training the network. Initially we thought of it as a component where we would send in input and output data. It would then return a dataset with necessary adjustments.

The method planned for creating datasets, and how it would operate, depended on the machine learning algorithm. Therefore it was separated as its own component.

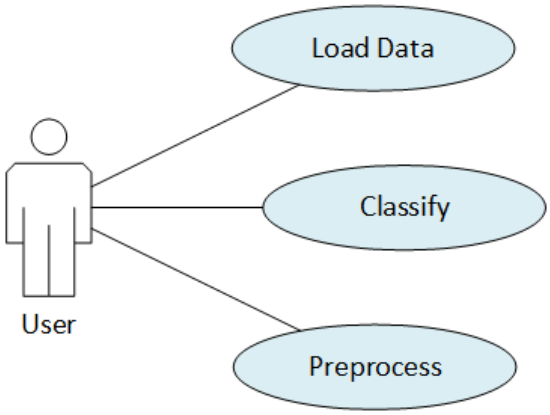


Figure 4.18: Use case diagram for Data Preparation Tool

The use case diagram in fig. 4.18 represents how a user will make use of the data preparation tool. The main functionality is to load, classify, and preprocess the data.

A common problem in machine learning is overfitting. If the weather conditions remain unchanged, the system can end up oscillating in a similar motion over a long time. Therefore we believe a model trained to predict the system is particularly exposed for overfitting.

Therefore a classifying method was implemented to classify signals, so that a method to balance the data could be done.

4.3.1.1 Classifying the data

The first part of classifying the data is splitting the signals up in classifiable parts. The data consists of discrete continuous signals of several sensor measurements. Most of the forces the system experiences is caused by waves. We therefore wanted to classify different types of waves out of the signal. We decided to split the signals up based on the signal periods. The user of the classifier can choose to classify over a signal that consists of an arbitrary amount of signal periods.

Labels is a way to categorize data by giving similar data a label, for example a number. At this stage of the process we had huge amounts of unlabeled, chopped up signals which we must categorize into labels, where each label describe the same kind of situation the stack is experiencing. Unsupervised ML is a popular choice in such situations.

The algorithm used is the K-Means algorithm. It starts with initializing random cluster points. The distance is calculated between the clusters and all points, and is grouped to the cluster with the shortest distance. Then it calculates the mean distance between all the grouped points and use this as a new cluster point. It all is repeated with the new clusters for n iterations. We used SKlearns implementation of the K-Means algorithm, which supports the Elkan algorithm that significantly increases the clustering speed [36].

Using the K-Means algorithm implies that there is a need to transform the signal chunks into n-dimensional points in space. Three different solutions to this problem was attempted.

Fourier transform method

The first method we tried was to use the Fourier transform to get the frequency domain. The output from this function is an array of X-values, and a 2-dimensional array of Y-values. The Y-value array contains imaginary numbers, seen in (4.20)

$$y_n = y_{real} + y_{imag} \tag{4.20}$$

This is then passed into a function to merge the signal period to a point, and convert it to polar form, (4.21, 4.22, 4.23).

$$p = (X, Y) \tag{4.21}$$

$$r = \sqrt{y_{real}^2 + y_{imag}^2} \tag{4.22}$$

$$\theta = \arctan\left(\frac{y_{imag}}{y_{real}}\right) \tag{4.23}$$

We then find the X and Y coordinates of the point, which is done by summing all values over a given time period, as seen in (4.24, 4.25).

$$X = \sum_{t=0}^{T} y_{imag(t)} \tag{4.24}$$

$$Y = \sum_{t=0}^{T} r(\cos\theta(t) + \sin\theta(t)) \tag{4.25}$$

The end result is an iterable containing points equal to the amount of signal periods. These are then passed into the K-Means clustering algorithm mentioned above.
Overall, the results are decent, but still not quite what we were looking for.

Polynomial Method

The second method attempted to merge a signal into a single point representing the data, was the polynomial method; fitting the signal to a polynomial function.

$$f(x) = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$$

After finding the coefficients of the polynomial, they can be used as points in a n-dimensional space cluster.

We started with a septic polynomial. Our initial results were not good at all. The constant part of the polynomial describes the "height" of the polynomial, and it was much larger than all the other coefficients. Therefore, we observed a trend were all signals that started at the same height was labeled as equal signals, because the distance between the points was much larger among that axis.

After scaling all the axis to the same scale results improved slightly. However they were still far from the performance of the Fourier transform method. A hypothesis formed that when the septic polynomial was scaled, the problem got reversed. Now the terms of little importance gained as much importance as the most important terms. We reduced the polynomial to a quadrinomial. The resulting polynomial did not fit as good to the signal, however, the clustering results improved greatly.

The next problem occurred when the signal periods become to long. For the curve fitting algorithm we simply used the sample number as the x value. Keeping the y value at bay with larger x values. Therefore, the polynomial was not fitted well with these signals. We solved this by scaling the x values from 0 to 1.

If we wanted to use more terms and not wash out the importance of some of the more important terms, it could attempted to scale each coefficient individually.

The coefficients in the polynomial can be found by using a method called ordinary least squared. Every signal measurement can be written as: $y_i = \theta_0 + \theta_1 \cdot x_i + \theta_2 \cdot x_i^2 + \ldots + \theta_m \cdot x_i^m + \epsilon_i$. Where θ is the coefficients and ϵ is the error. Finding the coefficient will be to find the θ value that gives the lowest square error. Seen in (4.26) [37]

$$\begin{aligned}
\hat{\theta}_0 &= \min_{\theta_0} \sum_{i=1}^{n} (y_i - \theta_0 - \theta_1 \cdot x_i - \theta_2 \cdot x_i^2 - \ldots - \theta_m \cdot x_i^m)^2 \\
\hat{\theta}_1 &= \min_{\theta_1} \sum_{i=1}^{n} (y_i - \theta_0 - \theta_1 \cdot x_i - \theta_2 \cdot x_i^2 - \ldots - \theta_m \cdot x_i^m)^2 \\
\hat{\theta}_2 &= \min_{\theta_2} \sum_{i=1}^{n} (y_i - \theta_0 - \theta_1 \cdot x_i - \theta_2 \cdot x_i^2 - \ldots - \theta_m \cdot x_i^m)^2 \\
&\vdots \\
\hat{\theta}_m &= \min_{\theta_m} \sum_{i=1}^{n} (y_i - \theta_0 - \theta_1 \cdot x_i - \theta_2 \cdot x_i^2 - \ldots - \theta_m \cdot x_i^m)^2
\end{aligned} \tag{4.26}$$

The entire signal can be written in matrix form as (4.27)

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^m \\ 1 & x_1 & x_1^2 & \ldots & x_1^m \\ 1 & x_2 & x_2^2 & \ldots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^m \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{4.27}$$

Also written as (4.28)

$$\overrightarrow{y} = \mathbf{X}\overrightarrow{\theta} + \overrightarrow{\epsilon} \tag{4.28}$$

Combining it with the ordinary least squares we get (4.29)

$$\overrightarrow{\hat{\theta}} = \arg\min((||\overrightarrow{y} - \mathbf{X}\overrightarrow{\theta}||)^2) \tag{4.29}$$

Scipy has a good implementation of this that we decided to use.

Legendre polynomial method

Another method is using Legendre polynomials. The method is quite similar to where we use the coefficients of the polynomial. The difference lay in the structure of the polynomial. The Legendre polynomial can be defined as (4.30).

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}(x^2 - 1)^n \tag{4.30}$$

When implementing the Legendre polynomial into the clustering method we went through the same steps as when implementing the polynomial. This was to confirm that what we learned from using the polynomial was true for Legendre as well. The main difference from polynomial was that the curve fitted better to non smooth signals.

We used the Numpy implementation of Legendre curve fitting.

4.3.1.2 The preprocesser

The preprocesser is simply a component where we gathered all preprocessing tools and filters. You give it some data and tell it what filter to use, and it returns the preprocessed data.

4.3.1.3 Creating datasets

Dataset quality is essential when training an ANN. We have multiple classes of ANNs we want to explore during this project, which require datasets in their own way. Thus, we must develop a method to easily make datasets for the various classes, while maintaining the same quality.

The data we will use to train our ANN models are vast and multidimensional. It therefore requires large amounts of preparation and manipulation before it can be used. The output from this component can be viewed as the *port* to the models we will make and train later in the project. Therefore excessive planning was important before any programming began. The use case can be seen in fig. 4.19

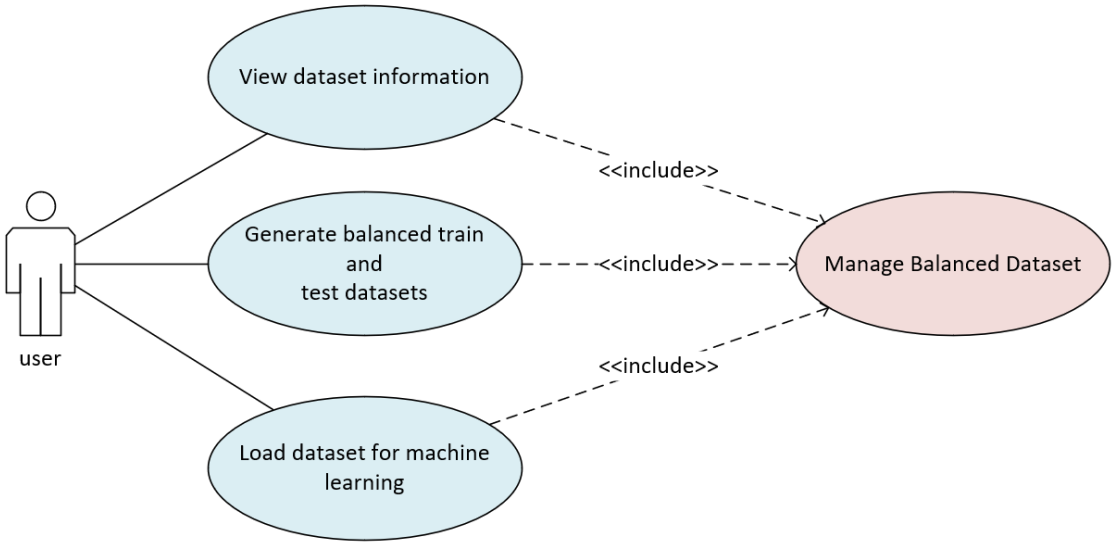


Figure 4.19: Use case diagram for the dataset generator

Balancing the data

An essential part of preparing data for training is balance [38]. It is important that the network experiences equal amounts of various situations, which in our case is the physical movements of the subsea stack.

Our previous steps have analyzed the signals, split them up in labeled categories that describe the same kind of physical motion and situation over a dynamic time span. The balancing component then takes that data and generate datasets containing the desired amount of DFs, with an equal amount of each label represented in the dataset.

This is important so that the model will not be overfitted to specific situations [39]. How the dataset is organized might differ from algorithm to algorithm, and class to class. Therefore the generator should be able to generate datasets regardless of class and algorithm.

Programming the generator

The generator first takes in a DF that includes a column containing a label for each row, made in our classifier. It then prepares the data, splits it up, and iterates over it to evenly distribute the data into balanced arrays. After this process it is stored in a custom object, which inherits from PyTorch's default dataset class [15]. PyTorch is quite picky when it comes to the way data should be passed into their systems. All matrices used must be stored in n-dimensional arrays called *tensors*, provided by PyTorch [15].

Creating batches

To load sample data into an ANN we must use something called a data loader. PyTorch already have a solution for that, which we ended up using [15]. The data loader wraps our dataset in an iterable which can be used to load batches of data into the model. We wanted to customize how the loader selects the data, thus we created custom samplers to fit our different network models as well.

4.3.1.4 Damage assessment

The reason we need the BM is due to its importance in calculating fatigue. Therefore, we made a software component as a way to further verify our predictions. The code is written based on the method used to calculate fatigue on real operations. The main class consists of three main functions. The calculations are executed in the order listed subsequently below.

Rainflow counting

Rainflow Counting (RFC) takes a series of BMs and returns the number of cycles per BM amplitude, within the defined bins [24]. A transformation to angles from BMx and BMy can be used to focus on specific hotspots, locations in the system most exposed to fatigue damage. The decision to not include the transformation was made, as we directed our focus to predicting BMx, thus it would not aid us in the task.

The RFC function is implemented as a window of customizable size, which is moved over a section of the BM series with a configured step size. Each window counts the amount of cycles within the defined amplitudes, and store them in bins. The bin and step size is configured such that overlap occurs. The result from each RFC is stored in a "bucket". We also store the total duration of each bucket, which is used for later calculation and visualization purposes.

MN-Curve

The second stage is to define the fatigue capacity of the specified hotspot of the WH system, which is done using MN-Curves, a generalization can be seen in (4.31),

$$logN = a - b \cdot logM \tag{4.31}$$

N represents the number of cycles which the WH hotspot can endure a BM of M. The results are stored in an unique iterable associated with each bucket in question. This is done for all buckets gathered during the RFC stage.

Damage calculation

The last step can be seen as two parts; damage rate, and fatigue. We first calculate the damage rate per bucket using Miners Rule, seen in (4.32)

$$D_r = \sum_i \frac{n_i}{N_i} \tag{4.32}$$

It iterates over the bins in each bucket collected in the RFC stage. Where i is the current bin, n the number of cycles in the bin, and N is the endurance retrieved in (4.31). The result from this summation represents the damage rate for that specific time window. The calculated damage rate is stored in an iterable which represents all damage rates over the total time period of all buckets. To find the accumulated fatigue, we to use the Riemann sum seen in (4.33)

$$D_t = \sum_j D_{r,j} \cdot dt_j \tag{4.33}$$

This is done using the iterable of damage rates calculated in the previous stage. dt is duration of each bucket retrieved from the RFC function, in years.

4.3.2 ANN manager

The ANN manager is a module created to simplify the training process and bind all the other modules together. By using the module, the user can start training, load data, and create a dataset. Which is then placed into a PyTorch dataloader. For each epoch the training results is saved. It also enables the possibility to queue files that should be trained on.

For the solution we did not want to define one ANN algorithm that should be tested, rather a solution where the manager could view the ANN as a black box, and insert data and get predictions out. The ANN manager is able to save and load different models. It is also able to deliver the data to the network and return the results. It stores trained models and plots. A simple UML diagram of the manager can be seen in fig. 4.20

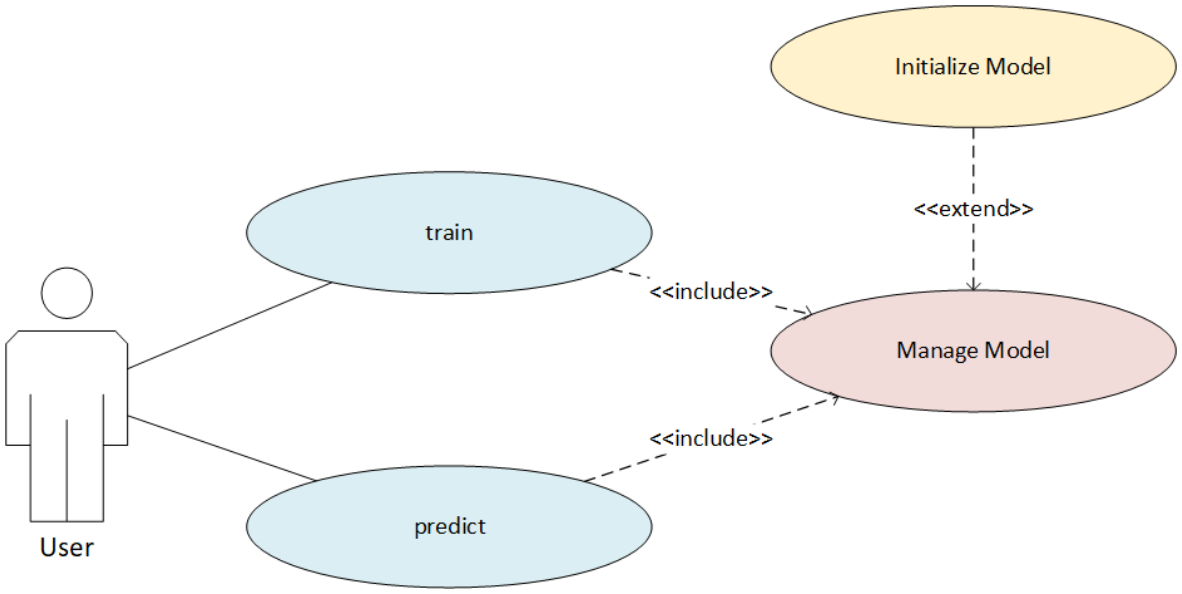


Figure 4.20: Class diagram for ANN Manager

The goal was to manage the models of the ANN. The manager contains two classes, manager and interface. The interface is an Abstract Base Class (ABC). The ABC [40] provides a blueprint for other classes. They do not contain any strongly defined implementation, but instead provide an interface to make sure that the derived classes are implemented similarly. Since the idea was to try different algorithms, it is important to have an interface, so the implementation of all algorithms contain the key features needed.

When deciding upon an interface there was two choices, to use protocols [41] or the ABC library [40]. We chose to use the ABC library, mainly due to its error handling features. When using ABCs you get errors when creating an instance if not all abstract methods are implemented. With protocols, you only get errors when trying to call an instance. Python being a dynamic language, does not in its base form support interfaces. Static languages linking happens during compiling and ABC allows for us to mimic this behaviour. This makes error handling much easier for the user.

4.4 Machine learning model

This section will cover the three different architectures of ANNs used in the project.

4.4.1 Feed-Forward neural network

Feed-Forward Neural Networks (FNN) are the simplest form of ANNs. It has a set of input nodes, a set of layers with n-nodes and output nodes. It only has connections in one direction feeding forward as the name implies. There are no form of recursion. A simple three layer deep FNN can be seen in fig. 4.21

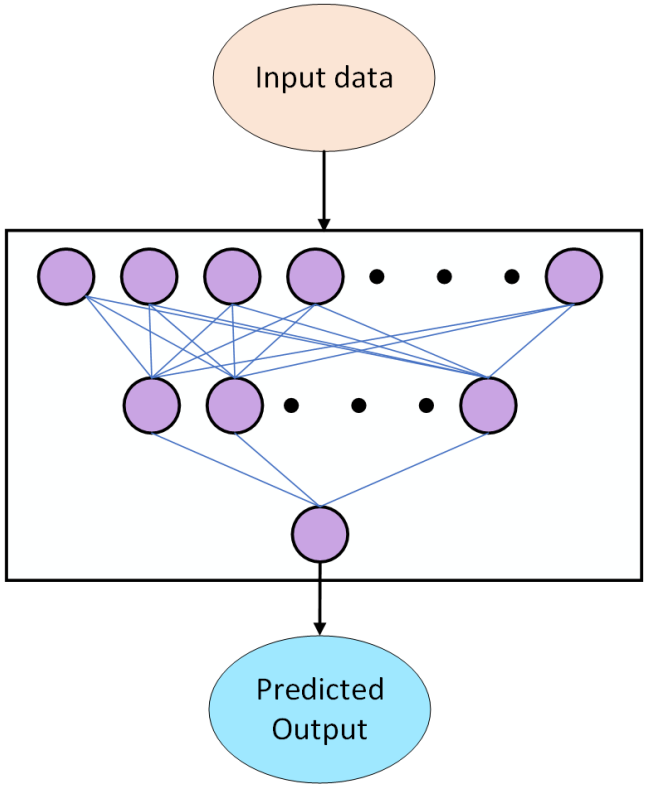


Figure 4.21: General visualisation of Feed-Forward Neural Network

Each node computes a function of its values and passes it to the next layer. A node within an ANN sometimes goes under the name unit. The equation (4.34) expresses the output of a unit *j*. The equation is explained as "*Let a_j denote the output of unit j and let $w_{i,j}$ be the weights attached to the link from unit i to unit j then we have;*

$$a_j = g_j\left(\sum_i w_{i,j}\, a_i\right) \equiv g_j(in_j) \tag{4.34}$$

where g_j is a non-linear activation function associated with unit j and in_j is the weighted sum of inputs to unit j" [2, p.803].

Each layer in the PyTorch implementation applies a linear transformation and adds a learned additive bias b as seen in (4.35). Note that this bias is optional, we found it best to use bias in all our networks.

$$y = x\boldsymbol{A}^{\mathsf{T}} + b, \qquad i = 1, \ldots, n, \tag{4.35}$$

4.4.1.1 Activation function in the network

Each node has an activation function that decides if the node is to fire. Fire in this context means passing its values further in the network. An activation function decides if a neuron is to be activated when a value exceeds an arbitrary threshold. Take for example the commonly used function Rectified Linear Unit (ReLU) [2, p.803] seen in (4.36).

$$ReLU(x) = \max(0, x) \tag{4.36}$$

For all positive values the neuron activates. Activation functions often have activation as a value between 0 and 1. But ReLU is non-saturated, meaning there is no upper limit to the value.

4.4.1.2 Bias

A bias tells the relevance of a node, and the weights tell how high the weighted sum needs to be before you get any meaningful activation. Each neuron have its own bias. When training a neural network it actually just entails the network learning the weights and biases that approximate the closes to the correct output. Adding the bias b to 4.34 we get:

$$a_j = g_j \left(\sum_i (w_{i,j}\, a_i) + b_j \right) \tag{4.37}$$

4.4.1.3 Loss function in the network

Calculating the loss is an important choice. We decided to use PyTorch's implementation of Mean Square Error (MSE) (See 4.38)

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{4.38}$$

MSE loss measures the difference between each element in the target y and estimate $\hat{y}$ [15]

4.4.2 Recurrent neural network

Unlike a regular FNN, the Recurrent Neural Network (RNN) feed output back into the input of the neural network. The output of one input in the RNN is therefore dependant on the last input. We can therefore say that the internal signals of the RNN sustains a memory. [2, p.802]

This is what makes the RNN excellent at predicting sequences. By applying a set length of sequence as the width of a regular FNN we could also train it to predict sequences. However it would be limited to a finite sequence length. Because of the loop in the network feed in recurrent we achieve a recursive back-propagation equation. Therefore, the only limit to the sequence length is the vanishing of gradients. [2, p.824-825]

The data we are working with is non-linear. Two equal inputs can give two different outputs. There are two reasons for this. The first is when the RFJ is on. The second reason is that there is a distance between the SMU sensors and the COB sensors. As the forces is applied at the top of the system and we predict at the bottom, it takes some time before the forces reaches the WH connector. For this reason, context for the input data is important, and it explains why we need a more specialized network like the RNN.

4.4.2.1 Building the network

The Python library PyTorch comes with a RNN module that can be used. Using the predefined RNN module we got no results of the training. There was an uncertainty of how the module managed the input. It is important that the network receives the input in correct sequence and resets its hidden state between each sequence.

To get rid of the uncertainty, we built it ourselves following PyTorch's guide [42]. As seen in fig. 4.22 the network consists of two linear network layers shown as blue boxes. The output of the hidden layer is fed back into the network concatenated by the input.

As it is proven that a non-saturated activation functions reduce the exploding gradient problem, we added a ReLU activation function on the output of the hidden layer [43].

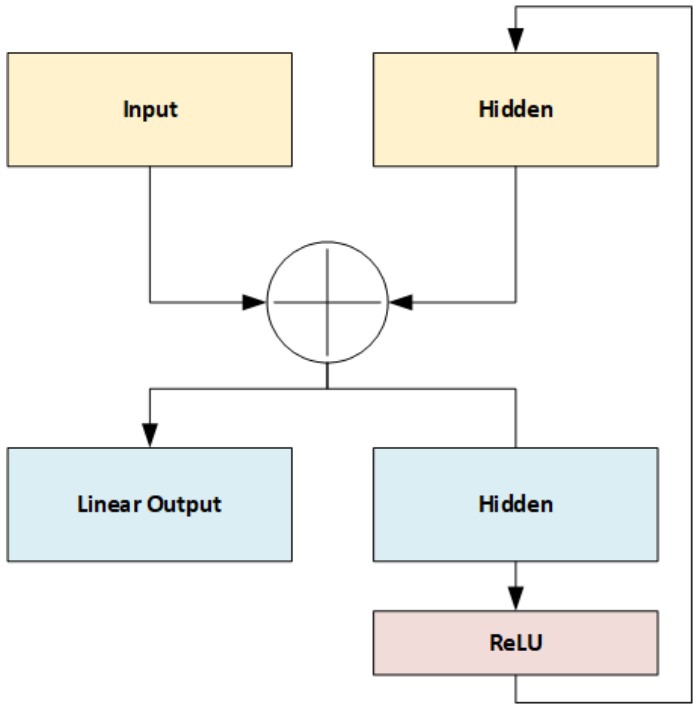


Figure 4.22: The structure of the RNNCell

There are three designs that we wanted to test for RNN. First of all, we wanted to test using a regular simple RNN cell to predict. In addition to this, we wanted to test a deeper RNN network where we start with a RNN cell, and add two linear layers after it.

Residual RNN

The last thing we wanted to test with RNN was a residual network. It is a structure that reduces the vanishing gradient problem. It is done by either concatenating or summing the input with one of the later layers in the network. It will create a "shortcut" for the input in the network.

The residual network consists of three subsequent RNN cells, followed up by six linear layers. The input have two "shortcuts" as well. One to the end of the RNN layers, and one to the end of the linear layers. There is also a connection between the end of the RNN layer to the end of the linear layers.

LSTM

We had not planned for this last design. However, because of some spare time at the end of the project and lacking results in the other RNN designs, we tried the Long short-term memory (LSTM) architecture.

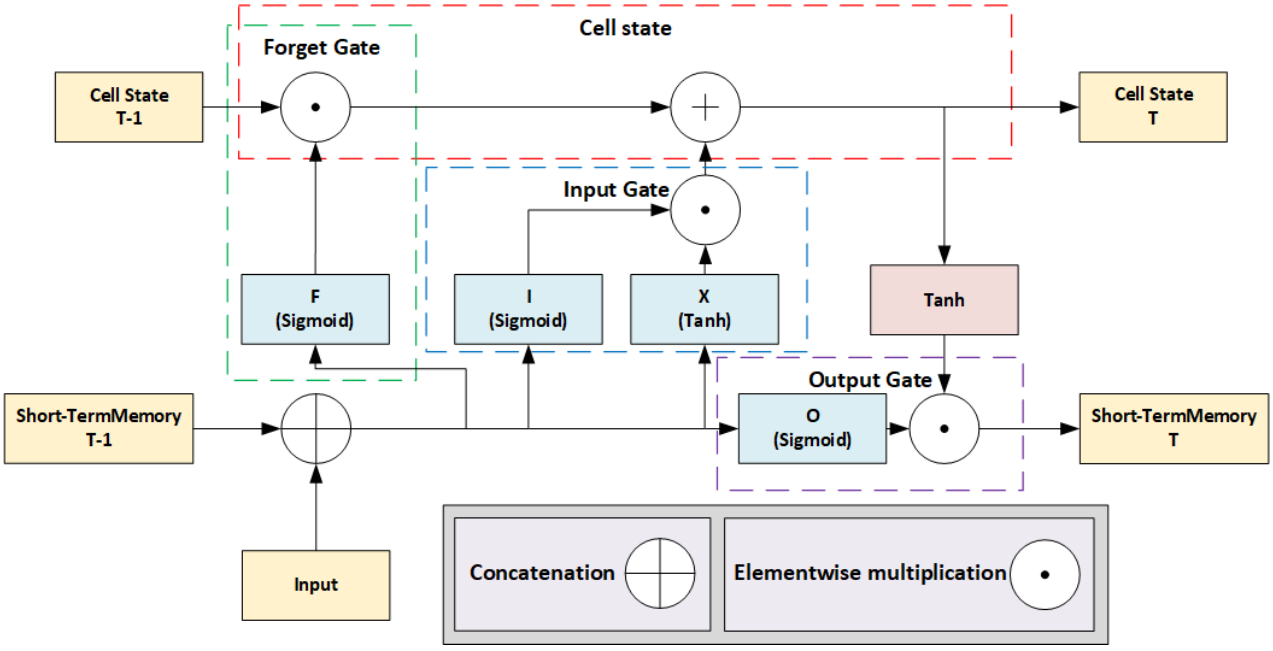


Figure 4.23: Visual representation of a LSTMcell based on equations from [2]

The architecture includes three flow gates that changes how the information is preserved over time. Fig. 4.23 is a visual representation of the equations for LSTM. The blue boxes is a linear feed forward layer with an activation function.

Marked with green dashed lines we have the forget gate. It determines if the elements in the memory cells is remembered or forgotten. [2, p.862]

The blue dashed box represents the input gate. It determines if each element in the memory cell is updated. Unlike the RNN, updates are done with addition instead of multiplication. This is the reason why we get a more stable gradient with this architecture as "*the gradients do not accumulate multiplicative over time*" [2, p.826].

The last gate is marked with a purple dashed box and is the output gate. It determines if the elements is transferred to the short-term memory. [2, p.862]

4.4.3 Convolutional neural network

A Convolutional Neural Network (CNN) is typically a FNN with a collection of 2-Dimensional Convolutional Layers (conv-2d) at the front. The layers represent a simplified model of the mammalian visual cortex [44]. Great results in visual tasks, such as object detection and face recognition has been achieved with 2D Convolutional Neural Networks (2D-CNN) in the recent years. Each layer typically consist of multiple Feature-Maps (FMAP). In a picture, the initial Input Feature-Maps (IFMAP) could for example be the red, green, and blue values in the image. The amount of FMAPs typically change layer by layer, where the unique information/features are preserved [45]. Each layer will be "scanned" by dynamic filter matrices called kernels, their values will be learned during training.

Recently, 1D Convolutional Neural Networks (1D-CNN) have shown promising results in the field of signal processing, such as structural health monitoring [46], patient-specific ECG heartbeat classifier [47], and bearing fault diagnosis using raw signals [48]. Malek et Al. write that *"CNNs are able to extract powerful feature for regression on 1D signals"* [49, p. 14].

We decided to explore 1D-CNN, as we are dealing with huge amounts of raw sensor data from multiple sensors.

4.4.3.1 1D Convolutional neural network model

The 1D-CNN share many of the same qualities as the 2D-CNN, the main differences are found in the layers, which we now will briefly explain. PyTorch has built in functionality for 1-Dimensional Convolutional Layers (conv-1d)s, defined as (4.39)

$$\mathbf{O}_{(C_{\text{out}},L_{\text{out}})} = \mathbf{B}_{(C_{\text{out}})} + \sum_{k=0}^{C_{\text{in}}-1} \mathbf{W}_{(C_{\text{out}},k)} \star \mathbf{I}_{(C_{\text{in}},L_{\text{in}})} \tag{4.39}$$

where $\mathbf{O}, \mathbf{B}, \mathbf{W}, \mathbf{I}$ are the Output Feature-Map (OFMAP), Bias, Kernel and IFMAP, respectively. $\star$ is the cross-correlation operator. We have broken the equation down into steps in the illustration seen in fig. 4.24.

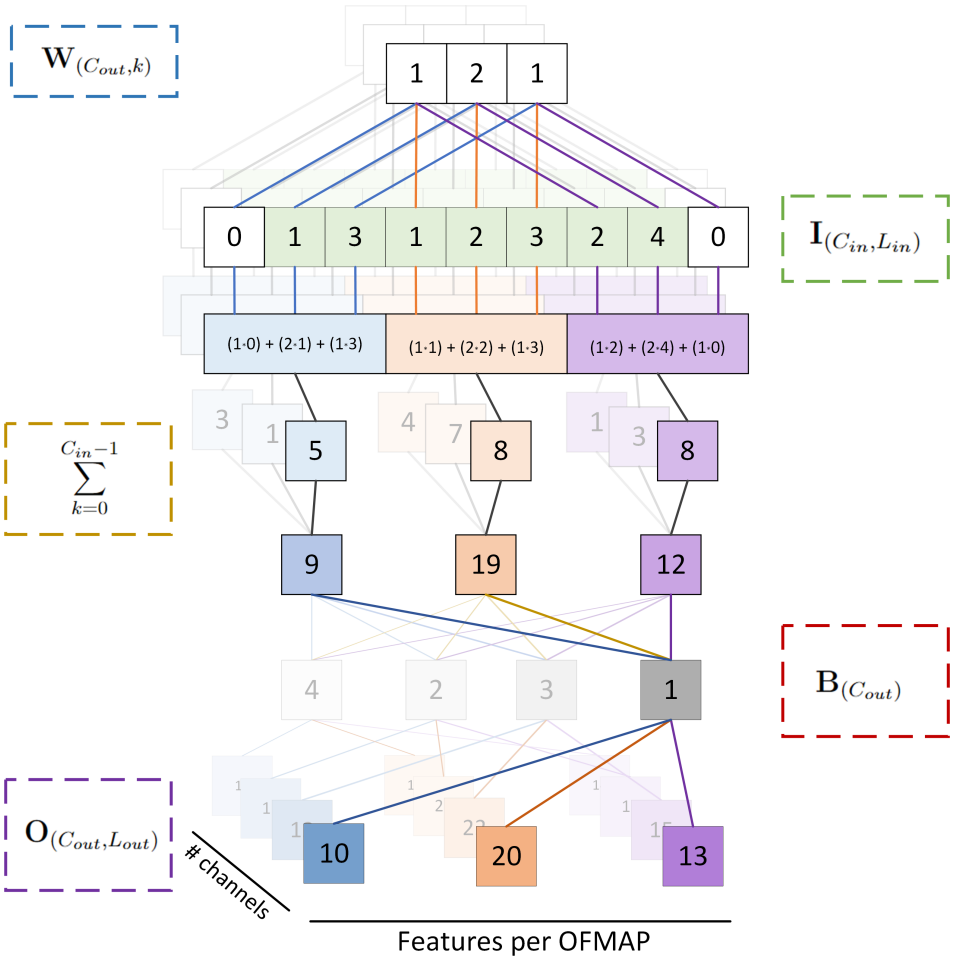


Figure 4.24: General visualisation of 1D Convolutional Layer in PyTorch.

In this case the kernel size and stride is set to 3. Input and output size is set to 3 and 4, respectively. The depth is the channels, and width the size per channel.

The first part is the kernel, represented as the $\mathbf{W}$ surrounded by blue stitched lines in fig. 4.24. Each IFMAP channel has its own unique and teachable kernel. The 1d-kernel is a collection of values, which will be optimized/learned during training. It can be viewed as a dynamic filter matrix. The calculation is rather simple, and it is controlled by three main parameters; stride, kernel size, and dilation. The kernels ($\mathbf{W}$) will take the product of each overlapping number in their respective IFMAP($\mathbf{I}$), and then sum the resulting products to produce a new value. The white boxes seen at each side of the IFMAP($\mathbf{I}$) is the padding, which is used to fill out the edges. We decided to pad with zeroes, but other values can be used as well.

The next step is to sum the values cross channel, depicted by yellow stitched lines in fig. 4.24.

The final part is to calculate the new OFMAPs ($\mathbf{O}$), where the amount of channels is equal to the amount of biases ($\mathbf{B}$). The values for each bias will be learned during training. All values from the previous step will be summed with each bias, producing a new OFMAP per bias present. In other words, the bias size is responsible of the number of OFMAPs at the end of each conv-1d layer.

Designing the network

Fig. 4.25 is a simplified visualisation our CNN design. Each section of green boxes in the figure represents a conv-1d layer described in fig. 4.24, note that the history is of various sizes between each layer. The flat layer between the last conv-1d layer and FNN is just a flattening layer, which can be viewed as an adapter between the last conv-1d layer and the input layer of the FNN.

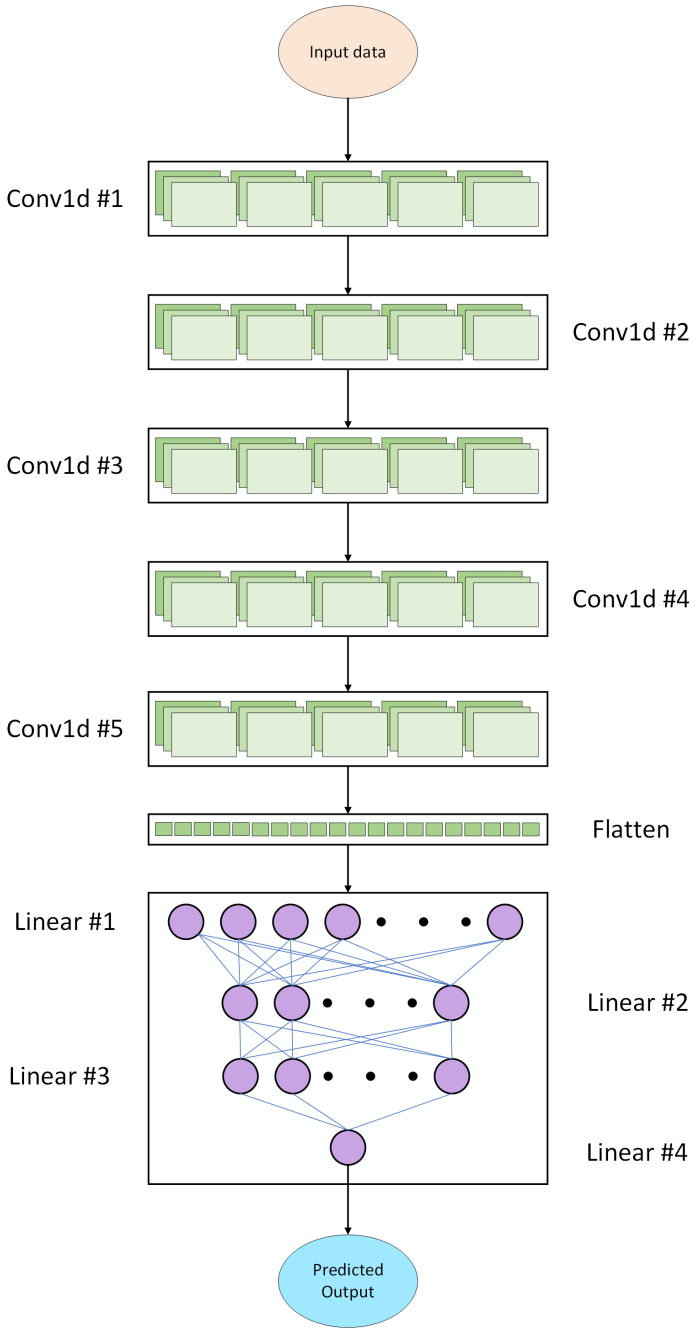


Figure 4.25: Simplified visualisation of our final Convolutional neural network.

Our final design is inspired by the TICNN proposed by Zhang et Al. [48], they suggest a convolutional architecture where the kernel size in the first layer is wide, and the next drastically shrink down to a small and consistent kernel size. The reasoning behind this choice relates to the

non-linear behaviour of the elastomer in the RFJ. Internal tests show us that the BM was affected by the previous stress positions it had experienced; in other words, the hysteresis and relaxation time of the elastomer. Table 4.10 is an overview of network layers, note that the activation, normalization, pooling, and dropout functions between the layers are not listed in the table, but they are in the model.

Table 4.10: CNN layer general overview

Layer type	Input shape	Output shape	Kernel size	Stride	Padding
conv-1d	2x640	16x40	64	8	31
conv-1d	16x40	32x20	3	1	1
conv-1d	32x20	64x10	3	1	1
conv-1d	64x10	64x5	3	1	1
conv-1d	64x5	64x1	3	1	0
Linear	64	32	-	-	-
Linear	32	16	-	-	-
Linear	16	4	-	-	-
Linear	4	1	-	-	-

The input to our network consists of two IFMAPs, specifically the inclination and rotational velocity produced by our Kalman filter (Sec 4.5.2). The signals are then passed into the first convolutional layer, one for each channel which makes up the initial two IFMAPs of our model. The FMAP depth change quite a lot during the journey through the layers, as seen in table 4.10. Notice that while the depth grows, the number of features shrinks.
At the end of the convolutional layers, we are left with 64 OFMAPs with 1 feature each, which is the input to the FNN at the end of the model. The FNN consists of 4 fully connected linear layers, using the Sigmoid activation function between layers 1-3. The output from the last layer is the predicted BM.

4.5 Fusion filters

There are numerous ways of finding the angles from accelerometer and gyroscope data. On this project, we decided to design many different types of inclination filters. The Kalman filter is the one that has yielded the greatest results from other researches such as [50]. The rest of the filters will only be described in short detail and could be read more about in other reports referred to in their sections.

4.5.1 Complementary

A complementary filter is based on filtering both the accelerometer and gyroscope data. It is known from (See 4.1.2.1) that gyroscopes are exposed to drift, while the accelerometers (See 4.2.4.1) are exposed to hard overshoots from the dynamic changes.

The complementary filter takes into consideration the sudden changes of the accelerometer with a low-pass filter and the drift of the gyroscope with a high-pass filter.

The filter works in the way of finding the most successful cut off frequency for both signal in a way that they always amplify the signal by one. That is why they complement each other. The equation for calculating the angle with a complementary in the s-domain is presented in (4.40).

$$\theta = \left(\frac{1}{1+\alpha s}\right)\theta_{Acc} + \frac{1}{s}\left(\frac{\alpha}{1+\alpha s}\right)\omega_{Gyr} \tag{4.40}$$

Where θ_{Acc} is the angle found in (See 4.2.4.1), ω_{Gyr} is the angular velocity found in (See 4.1.2.1) and $\frac{1}{s}$ is the integrator of the signal in the s-domain. The α is the coefficients for the low and high-pass filter.

The complementary filter is one of the simplest filters to implement because they have very little parameter changes under the testing. It will also be able to tell us a lot about which sensors the inclination measurements trust the most.

4.5.2 Linear Kalman

The Kalman filter is a much more complex filter to integrate in the system, but has some excellent features such as computing the best fusion of multiple sensors based on the covariance matrix. The filter was first introduced by Rudolf Kalman in 1960 [51] and was not received well in the signal processing community. The filter has proven in later research to be the most optimal estimator of linear systems [20].

Before setting up a Kalman filter, some state space matrices must be established. We recognize this from [52] where A is the state matrix, B is the input matrix, C is the output matrix and D is the feed forward matrix. This representation is used to look at the system in control form and can be used to transform the state equations of input variables as seen in (4.41).

$$\dot{\boldsymbol{x}} = A\mathbf{x} + B\mathbf{u} \tag{4.41}$$

The corresponding output of the system, that is linked to the input variables are written with (4.42)

$$y = C\mathbf{x} + D\mathbf{u} \tag{4.42}$$

Since we assume a constant acceleration model for the linear Kalman filter, we can visualize the equation of motion converted for angles in (4.43)

$$\theta_{n+1} = \theta_n + \theta_n \Delta t + \int \alpha \Delta t \tag{4.43}$$

We can therefore assume that the next time step angles can be computed with (4.44)

$$\begin{bmatrix} \hat{\phi}_{n+1} \\ \hat{\dot{\phi}}_{n+1} \\ \hat{\ddot{\phi}}_{n+1} \\ \hat{\theta}_{n+1} \\ \hat{\dot{\theta}}_{n+1} \\ \hat{\ddot{\theta}}_{n+1} \end{bmatrix} = \begin{bmatrix} \hat{\phi}_n + \hat{\dot{\phi}}_n \Delta t + \frac{1}{2}\hat{\ddot{\phi}}_n \Delta t^2 \\ \hat{\dot{\phi}}_n + \hat{\ddot{\phi}}_n \Delta t \\ \hat{\ddot{\phi}}_n \\ \hat{\theta}_n + \hat{\dot{\theta}}_n \Delta t + \frac{1}{2}\hat{\ddot{\theta}}_n \Delta t^2 \\ \hat{\dot{\theta}}_n + \hat{\ddot{\theta}}_n \Delta t \\ \hat{\ddot{\theta}}_n \end{bmatrix} \tag{4.44}$$

The matrix above (4.44) is called a state transition matrix and can be converted to the state space form, where the matrix to the right represent the A matrix and the matrix to the left represents the state vectors $\mathbf{x}$.

The new $A\mathbf{x}$ can be written as(4.45)

$$A\mathbf{x} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \hat{\phi}_n \\ \hat{\dot{\phi}}_n \\ \hat{\ddot{\phi}}_n \\ \hat{\theta}_n \\ \hat{\dot{\theta}}_n \\ \hat{\ddot{\theta}}_n \end{bmatrix} \tag{4.45}$$

Now that we have found the state transition matrix we can luckily leave the control matrix known as $B\mathbf{u}$ out of the equation, because the filter does not have a control form this will later be discussed (See 4.6)

One thing we do need, however, is the process noise known as Q. The state space model for the input vectors will look like(4.46)

$$\dot{\boldsymbol{x}} = A\mathbf{x} + Q \tag{4.46}$$

The process noise matrix can be found easily when the update sampling frequency are the same and we assume a constant acceleration model. We can find the process matrix with (4.47)

$$Q = \begin{bmatrix} \sigma^2_{\phi} & \sigma^2_{\phi\dot{\phi}} & \sigma^2_{\phi\ddot{\phi}} & 0 & 0 & 0 \\ \sigma^2_{\dot{\phi}\phi} & \sigma^2_{\dot{\phi}} & \sigma^2_{\dot{\phi}\ddot{\phi}} & 0 & 0 & 0 \\ \sigma^2_{\ddot{\phi}\phi} & \sigma^2_{\ddot{\phi}\dot{\phi}} & \sigma^2_{\ddot{\phi}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\theta} & \sigma^2_{\theta\dot{\theta}} & \sigma^2_{\theta\ddot{\theta}} \\ 0 & 0 & 0 & \sigma^2_{\dot{\theta}\theta} & \sigma^2_{\dot{\theta}} & \sigma^2_{\dot{\theta}\ddot{\theta}} \\ 0 & 0 & 0 & \sigma^2_{\ddot{\theta}\theta} & \sigma^2_{\ddot{\theta}\dot{\theta}} & \sigma^2_{\ddot{\theta}} \end{bmatrix} \tag{4.47}$$

We can also find Q by tweaking the input equation and adding a matrix that takes into account the acceleration model(4.48,4.49) respectively.

$$Q = AQ_aA^T \tag{4.48}$$

$$Q_a = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.49}$$

This will give us the new matrix Q as (4.50)

$$Q = \begin{bmatrix} \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ \frac{1}{2}\Delta t^3 & \Delta t^2 & \Delta t & 0 & 0 & 0 \\ \frac{1}{2}\Delta t^2 & \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & \frac{1}{2}\Delta t^3 & \Delta t^2 & \Delta t \\ 0 & 0 & 0 & \frac{1}{2}\Delta t^2 & \Delta t & 1 \end{bmatrix} \times \sigma_\alpha^2 \tag{4.50}$$

Where σ_α^2 is the standard deviation of the acceleration. With the input vectors in mind, we can go over to the observation matrices. These matrices are used to calculate the variance and uncertainty in the measured signals.

We remember from (4.1.1) that the manipulated data is converted to rotation and rotational velocity. The matrix for the output is seen in (4.51)

$$\begin{bmatrix} \phi_n \\ \dot{\phi}_n \\ \theta_n \\ \dot{\theta}_n \end{bmatrix} \tag{4.51}$$

The state vectors are smaller than the previous states derived in (4.44) since there are no angular acceleration data. This is not a problem, since the C matrix can be adjusted to comply with the rest of the system as seen in (4.52).

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.52}$$

The $\mathbf{x}$ is also the same as for the input variables. The last thing we need is a random noise vector that is very much the same process as the process noise, but luckily for us these variables were calculated (See 4.1.2.1, 4.2.4.1). The final output variables will be (4.53)

$$\boldsymbol{y} = C\mathbf{x} + R \tag{4.53}$$

and the R matrix can be found with (4.54).

$$R = \begin{bmatrix} \sigma_\phi^2 & 0 & 0 & 0 \\ 0 & \sigma_{\dot{\phi}}^2 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & 0 & \sigma_{\dot{\theta}}^2 \end{bmatrix} \tag{4.54}$$

That makes the R matrix (4.55)

$$R = \begin{bmatrix} 0.0001 & 0 & 0 & 0 \\ 0 & 0.0084 & 0 & 0 \\ 0 & 0 & 0.0002 & 0 \\ 0 & 0 & 0 & 0.0107 \end{bmatrix} \tag{4.55}$$

Now we have all the matrices to compute the Kalman filter.

The only thing we need to add is the initial values for the uncertainty matrix and the output values. Since the Kalman can scope in on the correct values very fast, we can set the uncertainty matrix to be fairly high with values of 100 in the 6×6 matrix, for the initial values we always play it safe and choose start values to be equal to 0 in the 6×1 matrix.

The first step of the filter is to send the first value through a prediction equation shown in (4.56)

$$P_{(n|n-1)} = AP_{(n-1|n-1)}A^T + Q \tag{4.56}$$

The returned value here is called P for Prediction. The subscripts here denotes n given n-1 or $P_{n|n-1}$. It can also be used as P_n that denotes the nth iteration. After this step we need to find the Kalman gain seen in (4.57)

$$K_{(n)} = P_{(n|n-1)}C^T(CP_{(n|n-1)}C^T + R)^{-1} \tag{4.57}$$

The next step is to send the Kalman gain into the estimate equation seen in (4.58)

$$\hat{x}_{(n|n)} = \hat{x}_{(n|n-1)} + K_{(n)}(y_{(n)} - C\hat{x}_{(n|n-1)}) \tag{4.58}$$

After that we can send the estimator and Kalman gain into the uncertainty equation shown in (4.59). Here the I symbol notes a identity matrix for the 6 ×6 matrix.

$$P_{(n|n)} = (I - KC)P_{(n|n-1)}(I - KC)^T + KRK^T \tag{4.59}$$

The final step is to update the state at what the Kalman filter believes is the corrected rotation, rotational velocity, and angular acceleration seen in (4.60)

$$\hat{x}_{(n+1|n)} = A\hat{x}_{(n|n)} \tag{4.60}$$

And then repeat the loop all over again for the next iteration. The Kalman filter is a complex procedure and needs computer powered aid to be calculated for many samples. We will see in the next section how the filter compares to the inclination measurements made by the sensors described (See 4.1.2.4).

4.5.3 Extended Kalman

The extended Kalman filter takes advantage of turning the linear filter into a non-linear filter. The problem with this is that it loses it optimal state estimator properties and can be out performed by other filters such as the Madgwick filter, and the Mahony filter [53].

The different steps here from the linear Kalman filter is the linearization of the A, B and C matrix. Therefore, the state space equation must be written in the form shown in (4.61)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, u) \tag{4.61}$$

For small signal linearization as in our example, we can assume they have the same equilibrium values at $\dot{\mathbf{x_0}}$ to be equal to 0 and $\mathbf{f}(\mathbf{x_0}, u_0)$ [52].

We can therefore assume that $\mathbf{x} = \dot{\mathbf{x}_0} + \Delta\mathbf{x}$ and the new expanded equation becomes (4.62)

$$\dot{\mathbf{x}}_0 + \Delta\dot{\mathbf{x}} \approx \mathbf{f}(\mathbf{x}, u) + A\Delta\mathbf{x} + B\Delta u \tag{4.62}$$

Now we can linearize the A and B matrix with partial derivatives seen in (4.63)

$$A = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \qquad \text{and} \qquad B = \frac{\partial \mathbf{f}}{\partial u} \tag{4.63}$$

and subtract the $\dot{\mathbf{x}}_0$ from the equation that yields (4.64)

$$\Delta\dot{\mathbf{x}} \approx A\Delta\mathbf{x} + B\Delta u \tag{4.64}$$

The same thing needs to be done with the output vector equation, and with the above in mind we can set up the output equation as seen in (4.65).

$$\Delta y \approx C\Delta\mathbf{x} + D\Delta u \tag{4.65}$$

When working with the Kalman filter, it is usually done with the incremental variable changes rather than the total measurements. The equation therefore becomes (4.66) [20].

$$[y - C(\mathbf{x}^*)] \approx C\Delta\mathbf{x} + D\Delta u \tag{4.66}$$

Where the $\mathbf{x}^*$ represents the trajectory of that exact time step. As we can see in the equations above, the extended Kalman filter is even more complex to implement than the linear model, and needs sufficiently more computing power to run. We will see in the next sections if this non-linear filter will give better results than the linear one.

4.5.4 Madgwick

The Madgwick filter was developed in 2009 by Sebastian Madgwick as part of his Ph.D and his report can be seen in [3]. The filter takes into consideration the quaternions as discussed earlier (See 4.1.1.2), and will only be covered briefly in this project.

The quaternions from the gyroscope is looked upon as raw data and only needs to be rotated into the correct axis from inertial to global axis seen in (4.67)

$$\omega = [0, \omega_\phi, \omega_\theta, \omega_\psi] \tag{4.67}$$

The rotation around the inertial axis can be turned to match the global axis, but also needs to be integrated to find the orientation in time shown in (4.68, 4.69)

$$\mathbf{q}^S_{E(t)} = \frac{1}{2}\hat{\mathbf{q}}^S_{E(t-1)} \otimes \omega t \tag{4.68}$$

$$\dot{\mathbf{q}}^S_{E(t)} = \hat{\mathbf{q}}^S_{E(t-1)} + \dot{\mathbf{q}}^S_{E(t)}\Delta t \tag{4.69}$$

Where $\otimes$ denotes quaternion multiplication shown in (See 4.1.1.2)

Note here that the quaternions based on the rotation to global axis is noted with $\mathbf{q}^S_E$, the previous estimate as $\hat{\mathbf{q}}^S_E$, and the quaternion derivative noted as $\dot{\mathbf{q}}^S_E$.

The accelerometer data can be turned into quaternions with (4.70)

$$\hat{\mathbf{Acc}} = [0, Acc_\phi, Acc_\theta, Acc_\psi] \tag{4.70}$$

and combine the data from the accelerometer to form the $\mathbf{f}_g$ equation seen in (4.71)

$$\mathbf{f}_g(\hat{\mathbf{q}}^S_{E(t)}, \hat{\mathbf{Acc}}) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) - Acc_\phi \\ 2(q_1q_2 + q_3q_4) - Acc_\theta \\ 2(\frac{1}{2} - q_2^2q_3^2) - Acc_\psi \end{bmatrix} \tag{4.71}$$

and $\mathbf{J}_g$ equation seen in (4.72)

$$\mathbf{J}_g(\hat{\mathbf{q}}^S_{E(t)}) = \begin{bmatrix} -2q_3 & 2q_4 & -2q_1 & 2q_2 \\ 2q_2 & 2q_1 & 2q_4 & 2q_3 \\ 0 & -4q_2 & -4q_3 & 0 \end{bmatrix} \tag{4.72}$$

The last thing needed for the equation loop to work, is the gradient of the $\mathbf{f}$ equation denoted $\nabla\mathbf{f}$ shown in (4.73)

$$\nabla\mathbf{f} = \left(\mathbf{J}_g(\hat{\mathbf{q}}^S_{E(t-1)})^T\mathbf{f}_g(\hat{\mathbf{q}}^S_{E(t-1)}, \hat{\mathbf{Acc}})\right) \tag{4.73}$$

and the estimated orientation from the quaternions shown in(4.74).

$$\mathbf{q}^S_{E\nabla,t} = \hat{\mathbf{q}}^S_{E(t-1)} - \mu_t\frac{\nabla\mathbf{f}}{||\nabla\mathbf{f}||} \tag{4.74}$$

That can be combined to create the filter seen in fig. 4.26

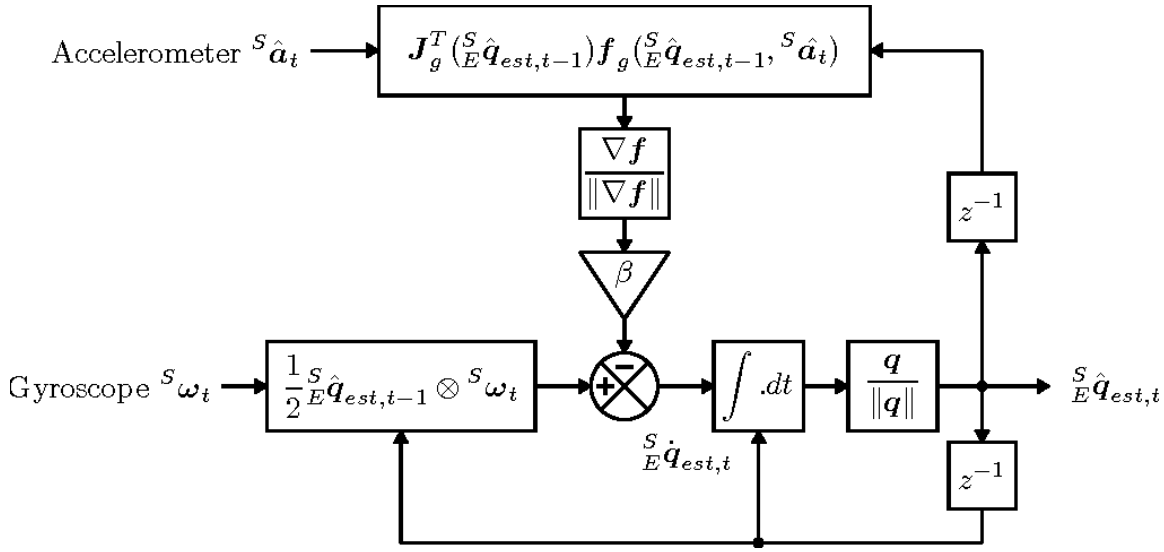


Figure 4.26: Block diagram for the Madgwick filter [3]

The only thing missing here is the gyroscope drift with a filter gain noted β discussed in section (See 4.1.2.1). It can be see from the above equations, that the Madgwick filter is just as hard as the Kalman filter to implement and will be exciting to see if it yields a better result in the next sections.

4.5.5 Mahony

The Mahony filter is sort of a fusion between the Madgwick and the Kalman filter, but resembles the complementary filter the most. The fusion filter takes in the accelerometer data and gyroscope data, and converts them into quaternions. The gyroscope data can be represented with (4.75)

$$\omega = \hat{\omega} + \mathbf{b}_g + \mathbf{n}_g \tag{4.75}$$

Where the $\mathbf{b}_g$ notes the bias in the gyroscope mentioned in previous sections. The normal distribution of the bias is noted with a Q and is similar to the one mentioned in the Kalman section. We can therefore write the bias as (4.76)

$$\dot{\mathbf{b}} = \mathbf{b}_g(t) \sim \mathcal{N}(0, Q_g) \tag{4.76}$$

The same can be said for the accelerometer, but the axes needs to be rotated for converting them into the global frame just like we did with in the orientation section (See 4.1.1). The accelerometer model can therefore be modelled as seen in (4.77).

$$\mathbf{a} = \mathbf{R}^T(\hat{\mathbf{a}} - g) + \mathbf{b}_a + \mathbf{n}_a \tag{4.77}$$

And the noise covariance matrix for the accelerometer can be added in the same way as we did with the Kalman filter presented with (4.78)

$$\dot{\mathbf{a}} = \mathbf{a}_g(t) \sim \mathcal{N}(0, Q_a) \tag{4.78}$$

This is all the parameters needed to represent the system and finding the Mahony angles. The next step is the computation part, which is after they are converted to quaternions. Starting of with finding the orientation error which was found in (4.70) that states:

$$\mathbf{f}_g(\hat{\mathbf{q}}^S_{E(t)}) = \begin{bmatrix} 2(q_2q_4 - q_1q_3) \\ 2(q_1q_2 + q_3q_4) \\ 2(\frac{1}{2} - q_2^2q_3^2) \end{bmatrix} \tag{4.79}$$

and calls the new function $\mathbf{e}$ for error. Then we compute the error for the system with the help of the accelerometer data and error for the integral of the system with (4.80,4.81)

$$\mathbf{e_{t+1}} = \hat{\mathbf{a}} \times \mathbf{f}_g(\hat{\mathbf{q}}^S_{E(t)}) \tag{4.80}$$

$$\mathbf{e_{i,t+1}} = \mathbf{e_{i,t}} + \mathbf{e_{i,t+1}}\Delta t \tag{4.81}$$

We can see that the error integral is just an numerical integration of the error function and resembles the same properties as an PI controller. This is roughly what the Mahony filter tries to estimate, it uses feedback to estimate its current position and needs a gain function to decide which parameter it should rely mostly upon. The $\mathbf{K_p}$ and $\mathbf{K_i}$ parameters are found from [52, p. 229] and can therefore be used to find the updated location of the gyroscope with (4.82)

$$\omega_{t+1} = \omega_{t+1} + \mathbf{K_p}\mathbf{e}_{t+1} + \mathbf{K_i}\mathbf{e}_{i,t+1} \tag{4.82}$$

The next step is to find the orientation increment for the gyroscope with the same equation used in the Madgwick section at (4.68)

$$\dot{\mathbf{q}}^S_{E(t)} = \frac{1}{2}\hat{\mathbf{q}}^S_{E(t+1)} \otimes \omega_{t+1}$$

The last step is to numerically integrate the quaternions, similar to the Madgwick filter in (4.69)

$$\mathbf{q}_{E(t+1)}^{S} = \hat{\mathbf{q}}_{E(t+1)}^{S} + \dot{\mathbf{q}}_{E(t+1)}^{S}\Delta t$$

This filter is a bit more complex than the complementary filter but still much more computing friendly than the Kalman filter. The last two filters only takes into consideration the integration of the gyroscope and the tilt of the accelerometer, but needs to be addressed since they can maybe give surprising results when we are working with non-linear systems like the stack is when the RFJ is enabled.

4.5.6 Integration

The integration technique uses the same orientation conversion as we described in earlier sections, but uses a technique from the Taylor expansion of polynomials to decompose the measurements. The technique behind it can be further read about in [54].

The thing to notice here is the use of the expansion of Euler number (See 4.83)

$$e^{x} = \sum_{k=0}^{\infty} \frac{1}{k!} x^{k} \tag{4.83}$$

With this equation we can manipulate the measurements to fit the gyroscope seen in (4.84)

$$e^{\frac{1}{2}\Delta t\Omega(\omega)} = \sum_{k=0}^{\infty} \frac{1}{k!} \left(\frac{1}{2}\Delta t\Omega(\omega)\right)^{k} \tag{4.84}$$

This technique has been known since the first research paper was realised in 1966 [55], and helped the submarines keep their course and heading over many days under the water.

The last thing we will investigate is the tilt function acquired from the accelerometer sensors.

4.5.7 Tilt

The Tilt filter only uses the conversion function we described in (See 4.9) that states:

$$\begin{bmatrix} \phi \\ \theta \end{bmatrix} = \begin{pmatrix} \arctan \frac{-Acc_x}{\sqrt{Acc_y^2 + Acc_z^2}} \\ \arctan \frac{-Acc_y}{Acc_z} \end{pmatrix}$$

As mentioned this is a extremely computing friendly inclination estimator and is affected by sudden movements since the accelerometer also senses the acceleration due to other things than the gravity.

4.6 Mathematical model

When deciding on the method to create the simplified mathematical model, the choice fell upon Hamiltonian and Lagrangian systems. In a Hamiltonian system without explicit time dependence, when no external energy is added to the system, the total energy is conserved.
For a conservative system, the Hamiltonian can be interpreted as the total energy of the system. [56, p.311]

$$\mathcal{H} = T + V \tag{4.85}$$

In reality and in our system the total energy is not conserved. Yet, the Hamiltonian approach is preferred because

1. It gives a good approximation for short time-steps
2. There are several good methods to integrate Hamiltonian Ordinary Differential Equations (ODE) numerically
3. It is of particular interest to TFMC to explore this path towards a simplified mathematical model

4.6.1 Method

Before diving into the calculations, the methods and theories that are used in these will be outlined in this section.

4.6.1.1 Lagrangian

Prior to expanding on the Hamiltonian (4.85), we have to take a step back and explore the Lagrangian, which plays an important role in its formation.

If the motion takes places in a conservative field, meaning that the total work done between two points is independent of the path taken, the potential energy (V) is a function of coordinates and not the velocities [57, p.178-183]. We can therefore find differential equations of motions using the Lagrangian.

In a conservative system where the forces are derivable from a potential, the Lagrangian function of the system becomes [56, p.282-284]

$$\mathcal{L} = T - V \tag{4.86}$$

4.6.1.2 Generalized momentum

The generalized momentum is defined as [56, p.284]

$$p_\alpha = \frac{\partial T}{\partial \dot{\theta}_\alpha} \tag{4.87}$$

In a conservative system with the potential energy depending only on the generalized coordinates, we can express the generalized momentum in terms of the Lagrangian (4.86) as [56, p.284]

$$p_\alpha = \frac{\partial \mathcal{L}}{\partial \dot{\theta}_\alpha} \tag{4.88}$$

4.6.1.3 Hamiltonian

William Hamilton introduced Hamiltonian mechanics in the 19th century as a mathematical reformulation of classical mechanics. Its original purpose was to express classical mechanics in a more unified and general manner. Over time, though, scientists have applied it to nearly every area of physics from thermodynamics to quantum field theory. [58, p.1]

Hamiltonian mechanics expand on the Lagrangian, which is used to derive a set of n second order differential equations of motion. The Lagrangian method extends into fields like relativistic quantum mechanics and to describe relativistic motion, but there are cases where a set of $2n$ first order differential equations would be more useful, and cases where momentum is a better method to describe motion than velocity. [59, p.1]

The Hamiltonian formulation of the laws of mechanics gives us an alternative way of representing the motion. Because the coordinates q_α and the momenta p_α are placed on an equal footing, it is natural to form a $2n$-dimensional space that will be spanned by the set of generalized coordinates. This is called the phase space of the mechanical system and allows a much simpler representation of the motion than the configuration space. [60, p.114]

To solve the system numerically we are interested in acquiring first order ODEs. To achieve this, we will use the Hamiltonian.

The Hamiltonian (4.85) can be defined in terms of the Lagrangian (4.86) as [56, p.311]

$$\mathcal{H} = \sum_{\alpha=1}^{n} p_\alpha \dot{\theta}_\alpha - \mathcal{L} \tag{4.89}$$

and must be expressed as a function of the generalized coordinates θ_α and the generalized momentum p_α. To accomplish this we must eliminate the generalized velocities $\dot{\theta}_\alpha$ from (4.89).

In this case, we can write the of the system as $\mathcal{H}(p_\alpha, \theta_\alpha)$ [56, p.311]

4.6.1.4 Hamilton's canonical equations

The equations of motion of the system can be written in skew symmetric form in terms of the Hamiltonian [56, p.311]

$$\dot{p}_\alpha = -\frac{\partial \mathcal{H}}{\partial \theta_\alpha} \tag{4.90}$$

$$\dot{\theta}_\alpha = \frac{\partial \mathcal{H}}{\partial p_\alpha} \tag{4.91}$$

These are called Hamilton's canonical equations (4.90, 4.91) and serve to indicate that the p_α and θ_α play similar roles in a general formulation of mechanical principles. [56, p.311]

4.6.2 Free body diagram of the system

To visualize the simplified model on which the calculations have been performed, this section includes two Free Body Diagrams (FBD) of the system, for different iterations of the mathematical model.

4.6.2.1 Inverted double pendulum

The system can be viewed as an inverted double pendulum, as seen in fig. 4.27. Starting from the bottom, the WH datum is placed in origo. The WH is connected to the RFJ which is represented by the concentrated mass m_1, the accumulated mass from the WH to the rotational axis of the LFJ, including the BOP and LMRP.

The second part of the pendulum (ℓ_2) is the MRA and one standard joint of the MR - the mass m_2 representing the accumulated mass from the rotational axis of the LFJ to the spot on the MR where we have chosen to cut it.
The tension in the MR generated by over-pull from the rig is represented by the force $F_{tension}$.

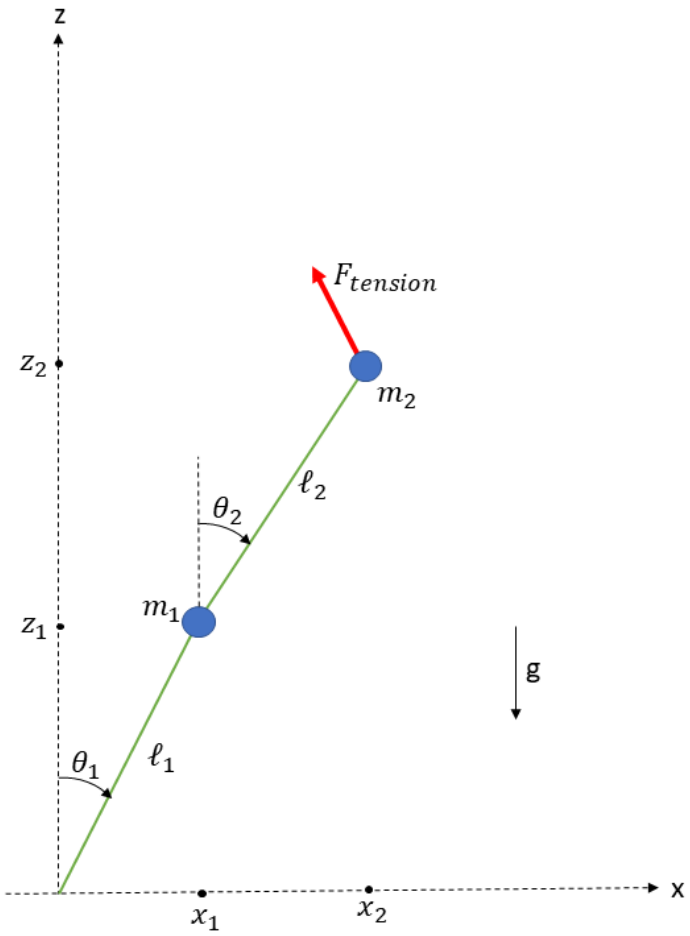


Figure 4.27: Inverted double pendulum

4.6.2.2 Positional equations

The positions of the two masses at the ends of each rod in the FBD shown in fig. 4.27 as the points (x_1, z_1) and (x_2, z_2) are given by

$$x_1 = \ell_1 \sin\theta_1 \tag{4.92}$$

$$z_1 = \ell_1 \cos\theta_1 \tag{4.93}$$

$$x_2 = \ell_1 \sin\theta_1 + \ell_2 \sin\theta_2 \tag{4.94}$$

$$z_2 = \ell_1 \cos\theta_1 + \ell_2 \cos\theta_2 \tag{4.95}$$

4.6.2.3 FBD with torsional springs

The stiffness of the WH and LFJ can be represented by torsional springs as illustrated in fig. 4.28, k_1 and k_2 respectively.
Torsional springs exert torque or rotary force, and is a good representation of the bending stiffness – the resistance of a member against bending deformation – in mathematical models.

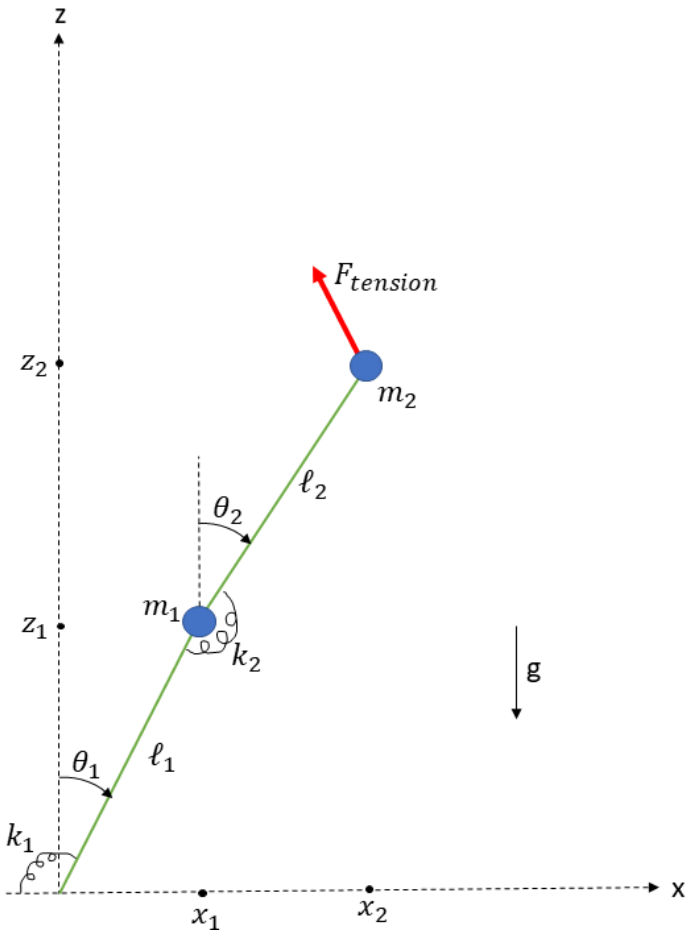


Figure 4.28: Inverted double pendulum with torsional springs

4.6.3 Full derivation of the system

The angles in fig. 4.27 can most likely be assumed to be small. Regardless, we performed a full derivation of the system to make an informed decision on the matter before moving forward with testing and analysis.
The initial derivation is performed without considering the torsional springs in fig. 4.27 because we initially wanted a foundational equation for the pendulum.

4.6.3.1 Energy equations

Potential energy

The potential energy (V) of the system shown in fig. 4.27 is found by combining the potential energy of the masses m_1 and m_2

$$V = -m_1 g z_1 - m_2 g z_2 \tag{4.96}$$

Substituting the values for the positions z_1 and z_2 given by (4.93, 4.95) in to (4.96) we get the expression for the potential energy of our system (4.97)

$$V = -(m_1 + m_2) g \ell_1 \cos\theta_1 - m_2 g \ell_2 \cos\theta_2 \tag{4.97}$$

Kinetic energy

The kinetic energy (T) of the system shown in fig. 4.27 is found by combining the kinetic energy each mass m_1 and m_2

$$T = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 \tag{4.98}$$

The velocity of each mass in our system can be expressed as the change in position due to time ($\frac{d}{dt}(x, z)$), and (4.98) is expanded to

$$T = \frac{1}{2} m_1 (\dot{x}_1 + \dot{z}_1)^2 + \frac{1}{2} m_2 (\dot{x}_2 + \dot{z}_2)^2 \tag{4.99}$$

Substituting the values for the positions z_1, x_1, z_2 and x_2 given by (4.93, 4.92, 4.95, 4.94) in to (4.98) we get the expression for the potential energy of our system (4.100) (See appendix D.1.1 for the complete derivation).

$$T = \frac{1}{2}(m_1 + m_2)\ell_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 \ell_2^2 \dot{\theta}_2^2 + m_2 \ell_1 \ell_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \tag{4.100}$$

4.6.3.2 Lagrangian

Substituting the expressions for potential and kinetic energy (4.97, 4.100) into the Lagrangian (4.86) the Lagrangian expands to (4.101)

$$\begin{aligned} \mathcal{L} = (&\frac{1}{2}(m_1 + m_2)\ell_1^2 \dot{\theta}_1^2 + \frac{1}{2} m_2 \ell_2^2 \dot{\theta}_2^2 + m_2 \ell_1 \ell_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + \\ &(m_1 + m_2) g \ell_1 \cos\theta_1 + m_2 g \ell_2 \cos\theta_2 \end{aligned} \tag{4.101}$$

Euler-Lagrange equations

If some of the forces in the system are conservative, while others are non-conservative, we can write the Euler-Lagrange equations as [56, p.284]

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}\right) - \frac{\partial \mathcal{L}}{\partial \theta_1} = 0 \tag{4.102}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}\right) - \frac{\partial \mathcal{L}}{\partial \theta_2} = 0 \tag{4.103}$$

By performing the derivations of the Lagrangian (4.101) the Euler-Lagrange equations (4.102,4.103) expand to (4.104,4.105) (See appendix D.1.2 for the complete derivation).

$$(m_1+m_2)\ell_1^2\ddot{\theta}_1+m_2\ell_1\ell_2\ddot{\theta}_2\cos(\theta_1-\theta_2)+m_2\ell_1\ell_2\dot{\theta}_1^2\sin(\theta_1-\theta_2)+\ell_1(m_1+m_2)g\sin\theta_1=0 \tag{4.104}$$

$$m_2\ell_2^2\ddot{\theta}_2 + m_2\ell_1\ell_2\ddot{\theta}_1\cos(\theta_1-\theta_2) - m_2\ell_1\ell_2\dot{\theta}_1^2\sin(\theta_1-\theta_2) + \ell_2 m_2 g\sin\theta_2 = 0 \tag{4.105}$$

The ODE (4.104, 4.105) of second order describe the motion of the system.

4.6.3.3 Generalized momentum

We can find the generalized momentum for our system by substituting the Lagrangian (4.101) into (4.88)

$$p_1 = \frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = (m_1+m_2)\ell_1^2\dot{\theta}_1 + m_2\ell_1\ell_2\dot{\theta}_2\cos(\theta_1-\theta_2) \tag{4.106}$$

$$p_2 = \frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = m_2\ell_2^2\dot{\theta}_2 + m_2\ell_1\ell_2\dot{\theta}_1\cos(\theta_1-\theta_2) \tag{4.107}$$

4.6.3.4 Hamiltonian

By solving the generalized momentum (4.106,4.107) for the angular velocity $\dot{\theta}_\alpha$ and substituting these expressions in to the expression for kinetic energy (4.100), we get an expression for the kinetic energy independent of the angular velocity parameter (See appendix D.1.3 for the complete derivation).
The Hamiltonian (4.85) becomes

$$\mathcal{H} = \frac{m_2\ell_2^2p_1^2 + \ell_1^2(m_1+m_2)p_2^2 - 2m_2\ell_1\ell_2p_1p_2\cos(\theta_1-\theta_2)}{2\ell_1^2\ell_2^2m_2[(m_1+m_2)\sin^2(\theta_1-\theta_2)]} - \\ (m_1+m_2)g\ell_1\cos\theta_1 - m_2g\ell_2\cos\theta_2 \tag{4.108}$$

Hamilton's canonical equations

The Hamiltonian for our system is a function of the initial conditions $(\dot{\theta}_\alpha, \dot{p}_\alpha)$, and Hamilton's canonical equations (4.90,4.91) become four first order ODEs (4.109,4.110,4.111,4.112) [61, p.10]

$$\dot{\theta}_1 = \frac{\partial \mathcal{H}}{\partial p_1} = \frac{\ell_2 p_1 - \ell_1 p_2 \cos(\theta_1 - \theta_2)}{\ell_1^2 \ell_2 [(m_1 + m_2)\sin^2(\theta_1 - \theta_2)]} \tag{4.109}$$

$$\dot{\theta}_2 = \frac{\partial \mathcal{H}}{\partial p_2} = \frac{\ell_1 p_2 (m_1 + m_2) - \ell_2 p_2 m_2 \cos(\theta_1 - \theta_2)}{\ell_1 \ell_2^2 m_2 [(m_1 + m_2)\sin^2(\theta_1 - \theta_2)]} \tag{4.110}$$

$$\dot{p}_1 = -\frac{\partial \mathcal{H}}{\partial \theta_1} = -(m_1 + m_2) g \ell_1 \cos\theta_1 - C_1 + C_2 \tag{4.111}$$

$$\dot{p}_2 = -\frac{\partial \mathcal{H}}{\partial \theta_2} = -m_2 g \ell_2 \cos\theta_2 + C_1 - C_2 \tag{4.112}$$

$$C_1 = \frac{p_1 p_2 \sin(\theta_1 - \theta_2)}{\ell_1 \ell_2 [(m_1 + m_2)\sin^2(\theta_1 - \theta_2)]}$$

$$C_2 = \frac{\ell_2^2 m_2 p_1^2 + \ell_1^2 (m_1 + m_2) p_2^2 - \ell_1 \ell_2 m_2 p_1 p_2 \cos(\theta_1 - \theta_2)}{2\ell_1^2 \ell_2^2 [(m_1 + m_2)\sin^2(\theta_1 - \theta_2)]^2} \sin[2(\theta_1 - \theta_2)]$$

4.6.4 Small angle approximation

Because the angles in our system (See 4.6.2) are very small ($\pm 3^\circ$) we can use small angle approximation for $\sin\theta$ and $\cos\theta$ (4.113, 4.114)

$$\sin\theta \approx \theta \tag{4.113}$$

$$\cos\theta \approx 1 - \frac{1}{2}\theta^2 \tag{4.114}$$

4.6.4.1 Energy equations

We apply the small angle approximations (4.113,4.114) to the expressions for potential and kinetic energy (4.97, 4.100) found earlier (See appendix D.2.2 and D.2.1 for the complete derivation).

Potential energy

$$V = \frac{gl_1m_1\theta_1^2}{2} - gl_1m_1 + \frac{gl_1m_2\theta_1^2}{2} - gl_1m_2 + \frac{gl_2m_2\theta_2^2}{2} - gl_2m_2 \tag{4.115}$$

Kinetic energy

$$T = \frac{l_1^2m_1\dot{\theta}_1^2}{2} + \frac{l_1^2m_2\dot{\theta}_1^2}{2} + l_1l_2m_2\dot{\theta}_1\dot{\theta}_2 + \frac{l_2^2m_2\dot{\theta}_2^2}{2} \tag{4.116}$$

4.6.4.2 Lagrangian

Combining the small angle approximation potential and kinetic energies (4.115, 4.116) the Lagrangian (See 4.6.3.2) becomes

$$\begin{aligned}\mathcal{L} = -\frac{gl_1m_1\theta_1^2}{2} + gl_1m_1 - \frac{gl_1m_2\theta_1^2}{2} + gl_1m_2 - \frac{gl_2m_2\theta_2^2}{2} + gl_2m_2 + \\ \frac{l_1^2m_1\dot{\theta}_1^2}{2} + \frac{l_1^2m_2\dot{\theta}_1^2}{2} + l_1l_2m_2\dot{\theta}_1\dot{\theta}_2 + \frac{l_2^2m_2\dot{\theta}_2^2}{2}\end{aligned} \tag{4.117}$$

Euler-Lagrange equations

The Euler-Lagrange equations (4.102, 4.103) derived from the small angle approximated expression for the Lagrangian (4.117) becomes (See appendix D.2.3 for the complete derivation).

$$l_1\left(gm_1\theta_1 + gm_2\theta_1 + l_1m_1\ddot{\theta}_1 + l_1m_2\ddot{\theta}_1 + l_2m_2\ddot{\theta}_2\right) = 0 \tag{4.118}$$

$$l_2m_2\left(g\theta_2 + l_1\ddot{\theta}_1 + l_2\ddot{\theta}_2\right) = 0 \tag{4.119}$$

4.6.4.3 Generalized momentum

The generalized momentum (See 4.6.3.3) derived from the small angle approximated expression for the Lagrangian (4.117) becomes

$$p_1 = \frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = l_1^2 m_1 \dot{\theta}_1 + l_1^2 m_2 \dot{\theta}_1 + l_1 l_2 m_2 \dot{\theta}_2 \tag{4.120}$$

$$p_2 = \frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = l_1 l_2 m_2 \dot{\theta}_1 + l_2^2 m_2 \dot{\theta}_2 \tag{4.121}$$

Kinetic energy

By solving the generalized momentum (4.120,4.121) for the angular velocity ($\dot{\theta}_\alpha$) and substituting these expressions into the expression for kinetic energy (4.100), we get an expanded expression for the kinetic energy (See appendix D.2.4 for the complete derivation).

$$T = \frac{p_2^2}{2 l_2^2 m_2} + \frac{p_2^2}{2 l_2^2 m_1} - \frac{p_1 p_2}{l_1 l_2 m_1} + \frac{p_1^2}{2 l_1^2 m_1} \tag{4.122}$$

4.6.4.4 Hamiltonian

Combining the kinetic and potential energies (4.122, 4.115) the Hamiltonian (See 4.6.3.4) becomes (See appendix D.2.5 for the complete derivation)

$$\begin{aligned} \mathcal{H} = \frac{g l_1 m_1 \theta_1^2}{2} - g l_1 m_1 + \frac{g l_1 m_2 \theta_1^2}{2} - g l_1 m_2 + \frac{g l_2 m_2 \theta_2^2}{2} - g l_2 m_2 + \\ \frac{p_2^2}{2 l_2^2 m_2} + \frac{p_2^2}{2 l_2^2 m_1} - \frac{p_1 p_2}{l_1 l_2 m_1} + \frac{p_1^2}{2 l_1^2 m_1} \end{aligned} \tag{4.123}$$

Hamilton's canonical equations

The Hamiltonian from small angle approximation (4.123) allows us to derive new first order ODEs (4.124,4.125,4.126,4.127) in the form of Hamilton's canonical equations (See 4.6.3.4)

$$\dot{\theta}_1 = \frac{\partial \mathcal{H}}{\partial p_1} = -\frac{p_2}{l_1 l_2 m_1} + \frac{p_1}{l_1^2 m_1} \tag{4.124}$$

$$\dot{\theta}_2 = \frac{\partial \mathcal{H}}{\partial p_2} = \frac{p_2}{l_2^2 m_2} + \frac{p_2}{l_2^2 m_1} - \frac{p_1}{l_1 l_2 m_1} \tag{4.125}$$

$$\dot{p}_1 = -\frac{\partial \mathcal{H}}{\partial \theta_1} = -g l_1 m_1 \theta_1 - g l_1 m_2 \theta_1 \tag{4.126}$$

$$\dot{p}_2 = -\frac{\partial \mathcal{H}}{\partial \theta_2} = -g l_2 m_2 \theta_2 \tag{4.127}$$

4.6.5 Torsional springs

Torsional springs are used to represent the stiffness of the WH and LFJ as shown in fig. 4.28. The spring force is defined from Hooke's law. Because the displacement in our system is represented by the change in the angle θ, we can express Hooke's law for our system as

$$F = k\theta \tag{4.128}$$

integrating it to find an expression for the potential energy in the torsional spring

$$V = \frac{1}{2}k\theta^2 \tag{4.129}$$

Which has the unit Newton-meters/radian (Nm/rad), and for our system shown in fig. 4.28 this becomes

$$V = \frac{1}{2}k_1\theta_1^2 + \frac{1}{2}k_2(\theta_2 - \theta_1)^2 \tag{4.130}$$

4.6.5.1 Potential energy

We add it this (4.130) to our expression for the potential energy in our system (4.115)

$$V = \frac{gl_1m_1\theta_1^2}{2} - gl_1m_1 + \frac{gl_1m_2\theta_1^2}{2} - gl_1m_2 + \frac{gl_2m_2\theta_2^2}{2} - gl_2m_2 + \frac{k_1\theta_1^2}{2} + \frac{k_2\theta_1^2}{2} - k_2\theta_1\theta_2 + \frac{k_2\theta_2^2}{2} \tag{4.131}$$

4.6.5.2 Lagrangian

We expand on the Lagrangian (See 4.6.3.2) with small angle approximations (4.117) and add the potential energy including torsional springs (4.131), in addition to our previous expression for the kinetic energy (4.122). This becomes

$$\begin{aligned}\mathcal{L} = -\frac{gl_1m_1\theta_1^2}{2} + gl_1m_1 - \frac{gl_1m_2\theta_1^2}{2} + gl_1m_2 - \frac{gl_2m_2\theta_2^2}{2} + gl_2m_2 - \frac{k_1\theta_1^2}{2} - \frac{k_2\theta_1^2}{2} + k_2\theta_1\theta_2 - \\ \frac{k_2\theta_2^2}{2} + \frac{l_1^2m_1\dot{\theta}_1^2}{2} + \frac{l_1^2m_2\dot{\theta}_1^2}{2} + l_1l_2m_2\dot{\theta}_1\dot{\theta}_2 + \frac{l_2^2m_2\dot{\theta}_2^2}{2}\end{aligned} \tag{4.132}$$

Euler-Lagrange equations

The Euler-Lagrange equations (4.102, 4.103) derived from the small angle approximated expression for the Lagrangian including torsional springs (4.132) becomes (See appendix D.3.1 for the complete derivation).

$$gl_1m_1\theta_1 + gl_1m_2\theta_1 + k_1\theta_1 + k_2\theta_1 - k_2\theta_2 + l_1^2m_1\ddot{\theta}_1 + l_1^2m_2\ddot{\theta}_1 + l_1l_2m_2\ddot{\theta}_2 = 0 \tag{4.133}$$

$$gl_2m_2\theta_2 - k_2\theta_1 + k_2\theta_2 + l_1l_2m_2\ddot{\theta}_1 + l_2^2m_2\ddot{\theta}_2 = 0 \tag{4.134}$$

4.6.5.3 Hamiltonian

We expand on the Hamiltonian (See 4.6.3.4) with small angle approximations (4.123) and add the potential energy including torsional springs (4.131), in addition to our previous expression for the kinetic energy (4.122). This becomes

$$\mathcal{H} = \frac{gl_1m_1\theta_1^2}{2} - gl_1m_1 + \frac{gl_1m_2\theta_1^2}{2} - gl_1m_2 + \frac{gl_2m_2\theta_2^2}{2} - gl_2m_2 + \frac{k_1\theta_1^2}{2} + \frac{k_2\theta_1^2}{2} - k_2\theta_1\theta_2 + \frac{k_2\theta_2^2}{2} + \frac{p_2^2}{2l_2^2m_2} + \frac{p_2^2}{2l_2^2m_1} - \frac{p_1p_2}{l_1l_2m_1} + \frac{p_1^2}{2l_1^2m_1} \tag{4.135}$$

Hamilton's canonical equations

The Hamilton's canonical equations (4.90,4.91) are only partially affected by the torsional springs, as the expressions for $\dot{\theta}_\alpha$ (4.124, 4.125) remain unchanged. This is because they are derived with respect to the generalized momentum, which in turn is affected by the angular velocity $\dot{\theta}$.

The torsional springs only has one unknown parameter - the angle θ. Therefore, we get new expressions for $\dot{p}_\alpha$

$$\dot{p}_1 = -\frac{\partial \mathcal{H}}{\partial \theta_1} = -gl_1m_1\theta_1 - gl_1m_2\theta_1 - k_1\theta_1 - k_2\theta_1 + k_2\theta_2 \tag{4.136}$$

$$\dot{p}_2 = -\frac{\partial \mathcal{H}}{\partial \theta_2} = -gl_2m_2\theta_2 + k_2\theta_1 - k_2\theta_2 \tag{4.137}$$

4.6.6 Applied Kalman filter

To contribute to the Kalman filter, an expression for the angular acceleration of the inverted pendulum with torsional springs shown in fig. 4.28 was derived from the mathematical model. Since the Kalman filter is based on a linear approach to the system model, the linearization of the pendulum is therefore not needed due to the equations above.

4.6.6.1 Acceleration equations

Solving the Euler-Lagrange equations (4.133, 4.134) with respect to $\ddot{\theta}_1$ and $\ddot{\theta}_2$, yields the expressions for the angular acceleration of the system (4.138, 4.139)

$$\ddot{\theta}_1 = -\frac{g\theta_1}{l_1} - \frac{gm_2\theta_1}{l_1 m_1} + \frac{gm_2\theta_2}{l_1 m_1} - \frac{k_1\theta_1}{l_1^2 m_1} - \frac{k_2\theta_1}{l_1 l_2 m_1} + \frac{k_2\theta_2}{l_1 l_2 m_1} - \frac{k_2\theta_1}{l_1^2 m_1} + \frac{k_2\theta_2}{l_1^2 m_1} \tag{4.138}$$

$$\ddot{\theta}_2 = \frac{g\theta_1}{l_2} - \frac{g\theta_2}{l_2} + \frac{gm_2\theta_1}{l_2 m_1} - \frac{gm_2\theta_2}{l_2 m_1} + \frac{k_1\theta_1}{l_1 l_2 m_1} + \frac{k_2\theta_1}{l_2^2 m_2} - \frac{k_2\theta_2}{l_2^2 m_2} + \frac{k_2\theta_1}{l_2^2 m_1} - \frac{k_2\theta_2}{l_2^2 m_1} + \frac{k_2\theta_1}{l_1 l_2 m_1} - \frac{k_2\theta_2}{l_1 l_2 m_1} \tag{4.139}$$

4.6.6.2 Application in Kalman filter

We can take this parameters into the state space equation described in section 4.5.2. Luckily, we only need the linear Kalman filter since the pendulum system is linearized with θ for small angle approximation. Our state variables becomes (4.140)

$$\mathbf{x} = \begin{pmatrix} \dot{x_1} = \dot{\theta}_1 = x_2 \\ \dot{x_2} = \ddot{\theta}_1 = (4.138) \\ \dot{x_3} = \dot{\theta}_2 = x_4 \\ \dot{x_4} = \ddot{\theta}_2 = (4.139) \end{pmatrix} \tag{4.140}$$

To make this more readable we will call the new values from the mathematical model $\mathcal{D}, \mathcal{E}, \mathcal{F}$ and $\mathcal{G}$ where their new values are seen in (4.141)

$$\begin{pmatrix} \mathcal{D} = -\frac{g}{l_1} - \frac{gm_2}{l_1 m_1} - \frac{k_1}{l_1^2 m_1} - \frac{k_2}{l_1 l_2 m_1} - \frac{k_2}{l_1^2 m_1} \\ \mathcal{E} = \frac{gm_2}{l_1 m_1} + \frac{k_2}{l_1 l_2 m_1} + \frac{k_2}{l_1^2 m_1} \\ \mathcal{F} = \frac{g}{l_2} - \frac{gm_2}{l_2 m_1} + \frac{k_2}{l_2^2 m_2} + \frac{k_2}{l_2^2 m_1} + \frac{k_2}{l_1 l_2 m_1} \\ \mathcal{G} = \frac{g}{l_2} - \frac{gm_2}{l_2 m_1} - \frac{k_2}{l_2^2 m_2} - \frac{k_2}{l_2^2 m_1} - \frac{k_2}{l_1 l_2 m_1} \end{pmatrix} \tag{4.141}$$

the $\mathbf{A}$ matrix can now be written on the new form with the equations above seen in (4.142)

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \mathcal{D} & 0 & \mathcal{E} & 0 \\ 0 & 0 & 0 & 1 \\ \mathcal{F} & 0 & \mathcal{G} & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{4.142}$$

We could then derive the $\mathbf{B}$ matrix, but since only the initial values are needed to get the filter started, we can set a start value greater than zero instead of managing the control matrix. The control values for the initial torque are defined with $\frac{T_c}{m_1 l_1}$ for the lower pendulum and $\frac{T_c}{m_2 l_2}$ for the upper. The reason for doing this is that the applied torque is non measurable and changes over time. Since we have both the angular acceleration and angular velocity from the first Kalman filter, the output equation can be made to be (4.143)

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{4.143}$$

Where we use the measured values from the first Kalman filter to be the measured values to the filter. This will hold the filter more manageable instead of applying the previous system equations (See 4.5.2).

Before starting to implement the Kalman filter, we need to convert the state space model into discrete time we can manipulate the equation shown (4.144)

$$\boldsymbol{x} = A\mathbf{x} + B\mathbf{u} \tag{4.144}$$

to be in discrete time with (4.145)

$$\frac{\boldsymbol{x}_n - \boldsymbol{x}_{n-1}}{\Delta t} = A\mathbf{x}_{n-1} + B\mathbf{u}_{n-1} \tag{4.145}$$

and solving the matrices for computing the prediction at time n with (4.146) [62].

$$\boldsymbol{x}_n = (I + \Delta t A)\boldsymbol{x}_{n-1} + \Delta t B\mathbf{u}_{n-1} \tag{4.146}$$

Where I is the identity matrix for the system and uses the Q matrix from the Kalman section (4.50). Since there is much information about the system, the process noise needs to be adjusted during the testing as we will see in the next section.

Chapter 5

Test & Simulations

Verifying the correct models and simulations is important for the development of the ML and filter fusion algorithms. Tests such as the F-test and MRMR-test have been implemented to see their statistical properties, but also tests like comparing the fatigue and detrending the predictions to make the output more reliable.

Simulations on the mathematical model have yielded promising values for the spring stiffness and showed a working differential equation for the inverted double pendulum. Calibration for the motion and strain sensors have been tested up against the historical data. This provided the correct output from the OrcaFlex simulations, while keeping the correct placements for the sensors intact.

5.1 OrcaFlex-model

There are two main areas in terms of testing the OrcaFlex model, sensor gathering, and the model itself. Tests on the model was done with a visual approach, calculations stating if the original values are intact or simply to check whether or not the program are able to run without errors. One example of the visual test was the implementation of historical mean wave directions. We ran a simulation with the changes in place, and then opened it in the OrcaFlex software. A quick look at the global axis revealed that our principal wave direction matched the historical mean wave direction. The sensor gathering focuses more on comparisons to historical data, as the goal is to produce a model outputting realistic data.

5.1.1 OrcaFlex model preservation

It was very important to maintain the correct core values from the provided model, while we changed it to suit our needs. One example of this, was when we changed some of the line component setups to increase the accuracy of our sensor data. Calculations regarding this topic can be found in appendix B.1.1 and B.3.1.

5.1.2 O2-System, overview

Throughout the process, all simulations and changes to the model have been written in a table (See appendix B.3). This overview (table B.13) shows what kind of changes each iteration had, and which simulations have been completed. This was important for the ML training, as it gives a good indication on which changes the OrcaFlex data need, for it to match with historical sensor data.

5.1.2.1 SMU buoy elements

The changes to the gathering of SMU data, by creating new buoy elements (See 4.2.2.2) provided more accurate motion for the z-acceleration as seen in fig. 5.2, compared to the previous iteration seen in fig. 5.1. In the previous iteration it shows that the z-acceleration remained at the same values most of the time. This is not how the actual sensor data behaves, a more similar result is provided with the new iteration. It is important to note that we are not looking for direct correlations between historical and OrcaFlex, but rather similar amplitude. More on this topic can be seen in section (See 4.2.2.2).

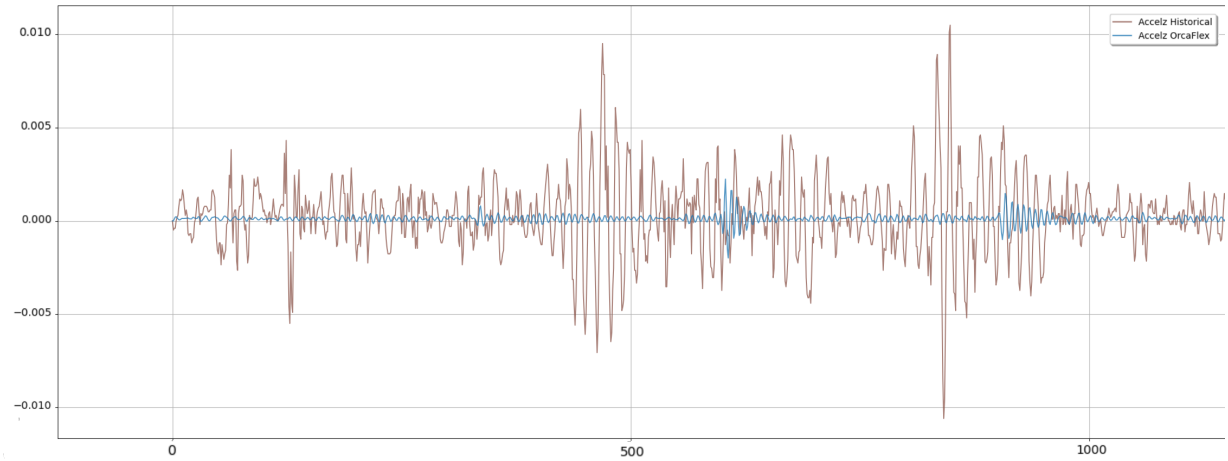


Figure 5.1: Without SMU3 buoy changes, OrcaFlex (blue) against historical (brown) data.

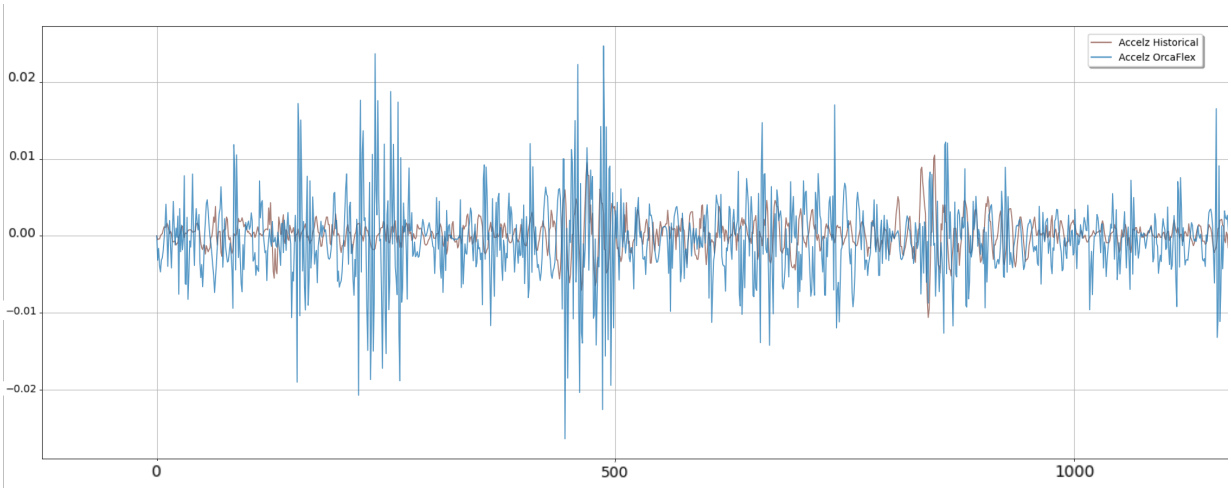


Figure 5.2: SMU3 buoy changes, OrcaFlex (blue) against historical (brown) data.x

5.1.3 Simulations

When running the simulations for the ML model, time sets of data with three hours length were used, more on this topic (See 4.2.5, 4.2.6). The ML model are in need of longer simulations than just one time set, preferably a whole day, and even consecutive days. This is done by setting up multiple simulations with coherent historical weather data. In the early stages of the CNN training, we provided data from short simulations, but as training progressed so did the length. Towards the end of the training process, the simulations stretch across whole days, with or without the RFJ on. Simulations had days with historical sensor data available and simulations without this data. We also set up simulations across a whole year, as the conditions out on sea changes quite a lot based on the seasons.

5.1.4 Sensor calibration

When testing the sensor output from OrcaFlex simulations, we are looking for similar behaviour in the signals and similar magnitude in the response. Our simulations are run with statistical wave data (See 2.6.2) and it is not possible, nor of particular interest, to replicate the historical signals beyond these parameters.

The plots shown in fig. 5.3 is a comparison of the historical sensor data (See 2.6.1) from Jan 15th 2020 and simulation data from OrcaFlex with historical wave data (See 2.6.2) from Jan 15th 2020.

A more comprehensive collection of calibration plots can be viewed in the appendix (See B.1).

5.1.4.1 Strain sensors

When measuring strain, it is important to take into account the size of the deformations, which are in microstrain, or μm/m. The comparison plots in fig. 5.3a show that the historical and simulated data varies within the same range, which should be considered a very good correlation after being repeated in other simulations.

The physical sensors are subject to many possible sources of error, especially during their installation. Small inaccuracies in the physical installation can affect the measurements in microstrain to an extent where we cannot expect perfect correlation.
In addition, the difference between simulation and reality must be recognized. The complexity of the forces acting upon the system are far beyond the scope of this project, and with the approximations we have made, such a correlation in magnitude must be considered a success, and a promising model to train the ML model on.

5.1.4.2 Motion sensors

It is of interest to look at the inertial behaviour of the SMU sensors placed above and below the RFJ, as they will exhibit different behaviour. The two SMU sensors placed on the BOP and LMRP as shown in fig. 4.6 are both part of a rigid body and will show similar behaviour.
We will therefore present the calibration plots for the SMU1 in and SMU3 sensors for the acceleration and rotational velocity parameters.

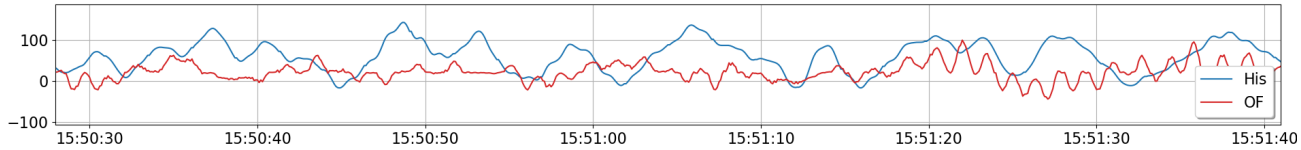
Acceleration

As we elaborated on in the previous chapter (See 4.2.4.1), the acceleration in xy-direction is the true lateral acceleration and the acceleration in z-direction is the gravitational component proportional to the riser angle to the vertical direction.

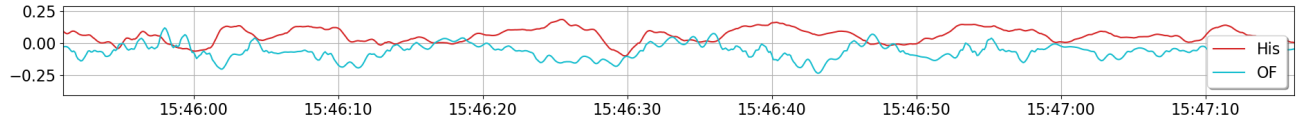
The correlation in magnitude is very good for both the SMU1 sensor and the SMU3 sensor. The acceleration of the SMU3 sensor is larger than the SMU1 sensor by a factor of 10, which is due to the fact that it is placed above the RFJ, and experiences the vibrations of the MR to a much larger extent than the sensors placed on the massive stack.

Rotational velocity

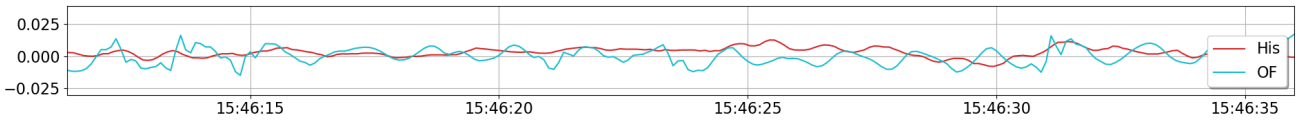
The same conclusion can be drawn for the rotational velocity. The correlation in magnitude for both SMU1 and SMU3 are very good.



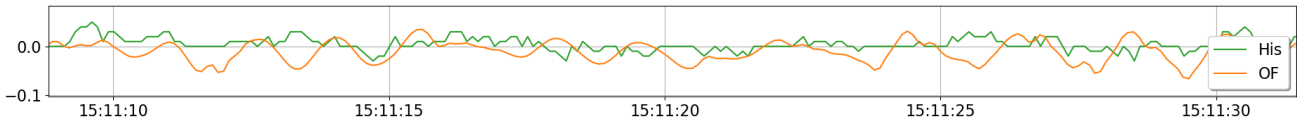
(a) Strain at the WH connector (μm/m)



(b) Acceleration above the RFJ (m/s^2)



(c) Acceleration at the lower BOP (m/s^2)



(d) Rotational velocity at the lower BOP (deg/s)

Figure 5.3: Comparison of historical sensor data and simulated OrcaFlex data

5.2 Testing the fusion filters

There are a bunch of filters from the design section that eventually are after the same goal of estimating the correct angle. The use of some known statistical and mathematical tests for selecting the correct filter will be beneficial, instead of manually rating every filter by appearance and performance.

The data collected for sampling are taken from 10,000 samples at 12:00 Dec 29th, when the RFJ was on, and at 12:00 Dec 31st when the RFJ was off. The statistical numbers can vary under different circumstances, but are good indicators to tell if we are moving in the right direction.

5.2.1 F-Test

We will start off with a statistical test known as the F-test. This statistical test is based upon (5.1) to estimate the equality within the variance region. This means that we can take the standard deviation of both the generated input signal and the known output and square them to yield their different variances.

$$F = \frac{\sigma_1^2}{\sigma_2^2} \tag{5.1}$$

Where σ_1^2 is the known output signals variance, in our example the BM measured at the WH and σ_2^2 represents the unknown variance in the output from the fusion filters. The variance is found with (5.2)

$$Var(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{y})^2 \tag{5.2}$$

Combined with the first equation, the statistical test calculates the ratio of the two signals to see if they are equal within their variance area.

The data used to compare against the BM are taken from dates when the RFJ was both on and off. This is important for choosing the right parameters for the ML model later on.

Matlab had an inbuilt function for finding the ratio between the signals and a visualization of the F-test can be seen in fig. 5.4

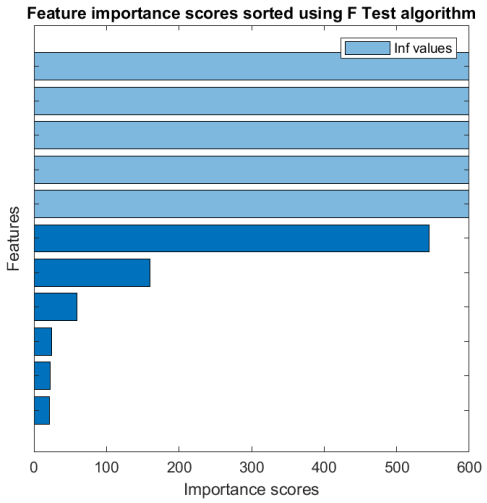


Figure 5.4: F-test algortihm

The F-test does not tell us if a signal should be a parameter for the ML model directly, but the values that fall under the category of "not infinite" should be thought of as not important. We can see their results more clearly from fig. 5.4 and in table 5.1 when the RFJ is off and 5.2 when the RFJ is on. Note that the number presented is just a ratio. The lower the number, the less matching variances they have.

Table 5.1: F-test RFJ off

Features	F-test SMU1	F-test SMU2	F-test SMU3
Complementary	inf	inf	inf
EKF	inf	inf	inf
KF	inf	inf	inf
KF$\dot{\theta}$	100	183	111
KF$\ddot{\theta}$	inf	124	inf
Integration	640	467	548
Madgwick	inf	214	inf
Mahony	inf	683	inf
Original Inclination	inf	inf	inf
Tilt	inf	inf	inf

Table 5.2: F-test RFJ on

Features	F-test SMU1	F-test SMU2	F-test SMU3
Complementary	421	inf	inf
EKF	inf	inf	inf
KF	inf	inf	inf
KF$\dot{\theta}$	7	19	inf
KF$\ddot{\theta}$	643	710	655
Integration	34	207	inf
Madgwick	99	inf	inf
Mahony	inf	inf	inf
Original Inclination	inf	inf	inf
Tilt	inf	inf	inf

5.2.2 MRMR-Test

The MRMR-test stands for minimum redundancy maximum relevance test. This algorithm is often used in gene research, but can be used to find correlation in datasets [63]. The test does not take into consideration the magnitude or size of the dataset, and must be transformed to a number between one and zero for the best possible outcome. This is called the mutual information between two discrete time series, seen in (5.3)

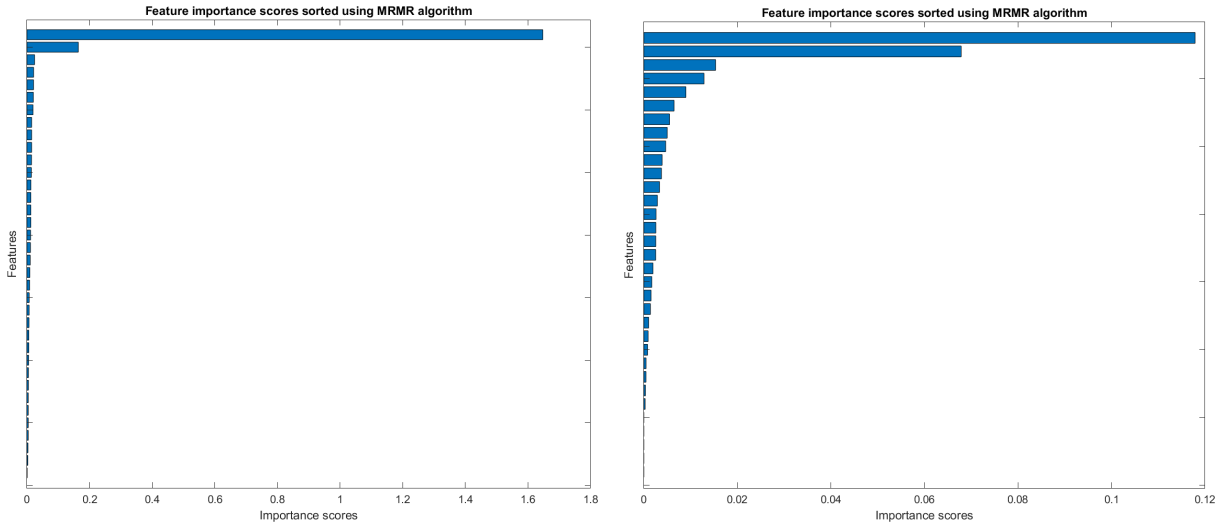
$$I(x,y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left[\frac{p(x,y)}{p(x)p(y)} \right] \tag{5.3}$$

Where p equals the marginal probability density and x, y is the different time series. Matlab [64] uses this equation together with the MRMR algorithm shown in (5.4,5.5),

$$V_s = \frac{1}{|S|} \sum_{x \in S} I(x,y) \tag{5.4}$$

$$W_s = \frac{1}{|S|^2} \sum_{x,z \in S} I(x,z) \tag{5.5}$$

To find the optimal an set of S that maximizes the relevance V_S with respect to S and y, and minimize the redundancy W_S with respect to S. A visualization of the MRMR-test is shown in fig. 5.5.



(a) RFJ off

(b) RFJ on

Figure 5.5: MRMR test.Feature impotance scores

The scores for RFJ off and on can be seen in table 5.3 and 5.4. Note that this is also a score based on the highest value, where a ratio of more than one defines a near identical fit.

Table 5.3: MRMR-test RFJ off

Features	SMU1	SMU2	SMU3
Complementary	0.0061	0.007	0.0048
EKF	0.0213	0.0151	0.0106
KF	0.0086	0.0126	0.0127
KF$\dot{\theta}$	0.0009	0.0042	0.0244
KF$\ddot{\theta}$	0.0073	0.0094	0.0150
Integration	0.0126	0.0206	0.0054
Madgwick	0.0113	0.0061	0.0150
Mahony	0.0042	0.0041	0.0030
Original Inclination	0.0117	0.0146	0.0195
Tilt	0.0141	0.0212	1.6475

Table 5.4: MRMR-test RFJ on

Features	SMU1	SMU2	SMU3
Complementary	0.481	0.007	0.007
EKF	0.012	0.017	0.010
KF	0.018	0.061	0.018
KF$\dot{\theta}$	$<$0.0000	$<$0.0000	0.014
KF$\ddot{\theta}$	0.005	0.011	0.004
Integration	0.579	0.007	0.055
Madgwick	0.003	0.036	0.015
Mahony	0.030	0.015	0.012
Original Inclination	0.009	0.016	0.026
Tilt	0.012	0.018	0.628

These values are not a solution to the parameter choosing for the ML model, but can be combined with the other data from the previous section to give a good indicator of which values that can help optimize the model.

5.3 Mathematical model

When testing the Hamiltonian ODEs that was found earlier in the different iterations (See 4.6.3, 4.6.4, 4.6.5), an evaluation was made of the Hamiltonian ODEs before creating a test environment in Python.

The four first order ODEs found in the first iteration without small angle approximation (4.109, 4.110, 4.111, 4.112) were considered too complex to continue working with, considering that the plan was to add two torsional springs and potentially a damper in the future iterations of the system.

The ODEs found in the derivation with small angle approximation and torsional springs (4.124, 4.125, 4.136, 4.137) were on the other hand considered very good, with a simplicity that makes it easier to perform tests on the system.

5.3.1 Solving differential equations

We implement the four first order ODEs we found using the Hamiltonian canonical equations (4.124, 4.125, 4.136, 4.137).

$$\dot{p}_1 = -\frac{\partial \mathcal{H}}{\partial \theta_1} = -gl_1m_1\theta_1 - gl_1m_2\theta_1 - k_1\theta_1 - k_2\theta_1 + k_2\theta_2$$

$$\dot{p}_2 = -\frac{\partial \mathcal{H}}{\partial \theta_2} = -gl_2m_2\theta_2 + k_2\theta_1 - k_2\theta_2$$

$$\dot{\theta}_1 = \frac{\partial \mathcal{H}}{\partial p_1} = -\frac{p_2}{l_1l_2m_1} + \frac{p_1}{l_1^2m_1}$$

$$\dot{\theta}_2 = \frac{\partial \mathcal{H}}{\partial p_2} = \frac{p_2}{l_2^2m_2} + \frac{p_2}{l_2^2m_1} - \frac{p_1}{l_1l_2m_1}$$

We solve these ODEs by defining the start values for our variables, the duration of our test and sampling rate per time-step. The variables are returned in the particular order p_1, p_2, θ_1, θ_2.

These variables (p, θ) represent the generalized coordinates and momentum of our system, and can be regarded as its total energy.

5.3.2 Defining constants

To find the values for the generalized coordinates and momentum, we must define the constants that represent the material and geometric properties of our system.

We have chosen to simplify the masses by introducing two concentrated mass points m_1 and m_2 and neglect the kinetic energy due to inertial forces (See 8.3).
Because of this, we are interested in the accumulated mass for the length of each rod in the pendulum. The length ℓ_1 in our system as illustrated in fig. 4.28 and fig. 2.5, begins at the WH and includes the BOP, LMRP, and the LFJ below its axis of rotation, while for ℓ_2 it includes the LFJ above its axis of rotation, the MRA and one standard riser joint.

The mass m_1 is the accumulated submerged weight[kg] (filled with sea water), equal to its weight minus the buoyancy force [65, p.57-59], from the WH along the length ℓ_1 to the rotational axis of the LFJ (including the mass of the RFJ), while the mass m_2 is the accumulated submerged weight[kg] (filled with sea water) from the rotational axis of the LFJ (including the mass of the RFJ) to the end of the first standard riser joint connected to the MRA.

TFMC provides an interface data sheet for each operation [66], where we have found the values for the lengths and masses for our system, outlined in table 5.5.

Table 5.5: Mathematical model constants

Parameter	Description	Value	Unit	Reference
m_1	Mass	222385	kg	[66, p.22]
m_2	Mass	24963	kg	[66, p.22]
ℓ_1	Length	11.436	m	[66, p.20-21]
ℓ_2	Length	25.601	m	[66, p.20-21]
g	Gravitational acceleration	-9.80665	m/s^2	
k_1	Spring stiffness	$182 \cdot 10^7$	Nm/rad	(5.9)
k_2	Spring stiffness	$455 \cdot 10^6$	Nm/rad	(5.10)
E	Young's modulus	$205 \cdot 10^9$	Pa	[4, p.22]
$d_{1,inner}$	Inner diameter	1.131	m	[4, p.22]
$d_{1,outer}$	Outer diameter	1.162	m	[4, p.22]
$d_{2,inner}$	Inner diameter	0.48	m	[4, p.22]
$d_{2,outer}$	Outer diameter	0.56	m	[4, p.22]

5.3.2.1 Torsion spring stiffness

We can assume that all bending for each of the rods (ℓ_1 and ℓ_2 in fig. 4.28) will happen at the point where the distance to the force is at its maximum. The bottom of the MRA for ℓ_2 and the cross-section just above the WH for ℓ_1.

To find the spring constant we use the moment distribution method, which states that the rotational stiffness of a member is proportional to its material and geometric stiffness. [67]

$$k = \frac{EI}{L} \tag{5.6}$$

Where the k is equal to the product of Young's modulus (E) and the second moment of inertia (I). For a hollow circular cross-section the second moment of inertia is

$$I = \frac{\pi(d_{\text{outer}}^4 - d_{\text{inner}}^4)}{64} \tag{5.7}$$

expanding our expression for the spring stiffness to

$$k = \frac{EI}{L} = \frac{\pi E(d_{\text{outer}}^4 - d_{\text{inner}}^4)}{64L} \tag{5.8}$$

The length L is the length of the beam which we expect to bend due to the loads, and in our case this is the circular cross section at the WH connector and the circular cross section at the MRA, for the lower and upper rod of the pendulum respectively. These lengths are $\pm$ 1m and we approximate them both to $L = 1\text{m}$ in this iteration.

The diameters listed in table 5.5 represent the cross-sections immediately above the WH and at the bottom of the MRA, for d_1 and d_2 respectively. By substituting these values in to our formula for the torsional spring stiffness (5.8) we acquire the values for k_1 and k_2 which can be used to solve our differential equations.

$$k_1 = \frac{\pi \cdot (205 \cdot 10^9\text{Pa}) \cdot ((1.162\text{m})^4 - (1.131\text{m})^4)}{64 \cdot 1\text{m}} = 182 \cdot 10^7\text{Nm/rad} \tag{5.9}$$

$$k_2 = \frac{\pi \cdot (205 \cdot 10^9\text{Pa}) \cdot ((0.56\text{m})^4 - (0.48\text{m})^4)}{64 \cdot 1\text{m}} = 455 \cdot 10^6\text{Nm/rad} \tag{5.10}$$

5.3.3 Simulations

The four first order Hamiltonian ODEs (4.124, 4.125, 4.136, 4.137) are now ready to be tested. A program for testing ODEs in Python was created, where we input the four first order ODEs, the constant values and the order in which to return our variables.

Table 5.6: ODE input variables

Type	Value	Unit
Test duration	15	sec
Sampling rate	1000	Hz

Table 5.7: ODE initial conditions

Type	Value	Unit
Generalized coordinate (θ_1)	0.005	deg
Generalized coordinate (θ_2)	0.01	deg
Generalized momentum (p_1)	0	$kg \cdot m/s$
Generalized momentum (p_2)	0	$kg \cdot m/s$

5.3.3.1 Generalized momentum

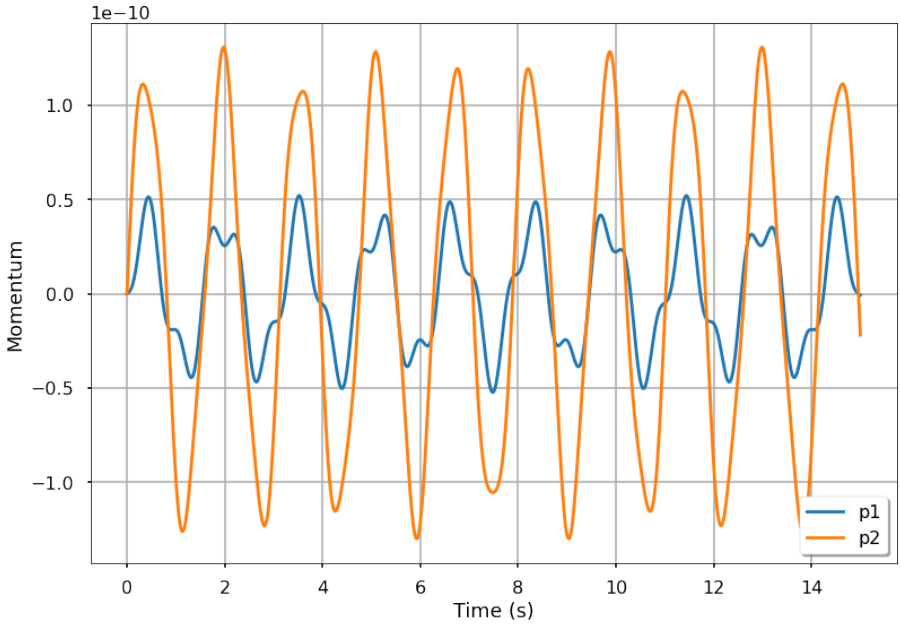


Figure 5.6: Inverted double pendulum generalized momentum

The behaviour of the four first order Hamiltonian ODEs are as one would expect of an inverted double pendulum with its stiffness, and it confirms the validity of the canonical equations.

The generalized momentum in fig. 5.6 show that the upper pendulum represented by ℓ_2 and m_2 in fig. 4.28, has a velocity that is much larger than that of the lower pendulum.
We look at the momentum at $t = 2$ in fig. 5.6.

$$p_1 = m_1 v_1 \rightarrow v_1 = \frac{p_1}{m_1} = \frac{0.35kg \cdot m/s}{222385kg} = 1.57 \cdot 10^{-6} m/s \tag{5.11}$$

$$p_2 = m_2 v_2 \rightarrow v_2 = \frac{p_2}{m_2} = \frac{1.25kg \cdot m/s}{24963kg} = 5 \cdot 10^{-5} m/s \tag{5.12}$$

The assumption holds true (5.11,5.12), and to test it further we decrease the stiffness of the torsional spring k_2 with a factor of 10 ($k_2 = 4.55 \cdot 10^5$) and as expected see a very large increase in the velocity (5.13) of the upper pendulum, shown in fig. 5.7.

$$v_2 = \frac{p_2}{m_2} = \frac{4.00kg \cdot m/s}{24963kg} = 1.6 \cdot 10^{-4} m/s \tag{5.13}$$

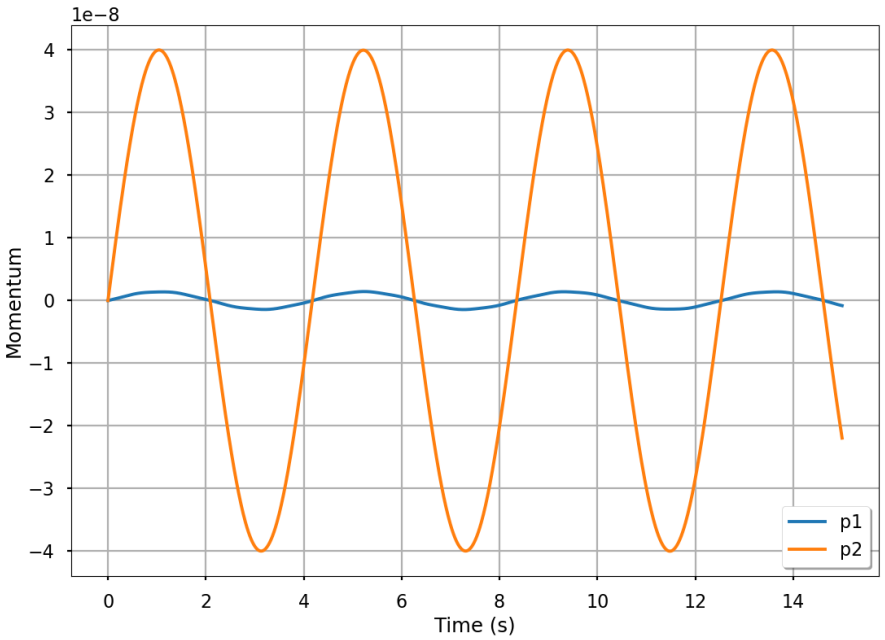


Figure 5.7: Inverted double pendulum generalized momentum. Decreased stiffness

5.3.3.2 Generalized coordinates

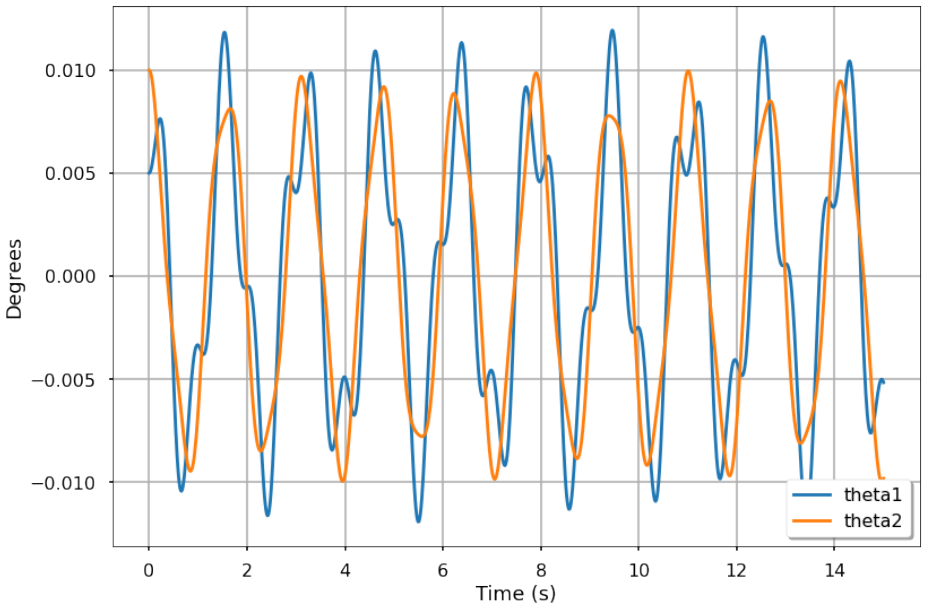


Figure 5.8: Inverted double pendulum generalized coordinates

The generalized coordinates visualized in fig. 5.8 show that the angles of the two rods in the pendulum (θ_1, θ_2) shown in fig. 4.28 follow the movement of each other with a small time delay, as expected.

To test it further, we increase the stiffness of the spring k_2 with a factor of 10 ($k_2 = 4.55 \cdot 10^7$), and as expected see a behaviour of a large rigid body of with a slight delay due to time shown in fig. 5.9.

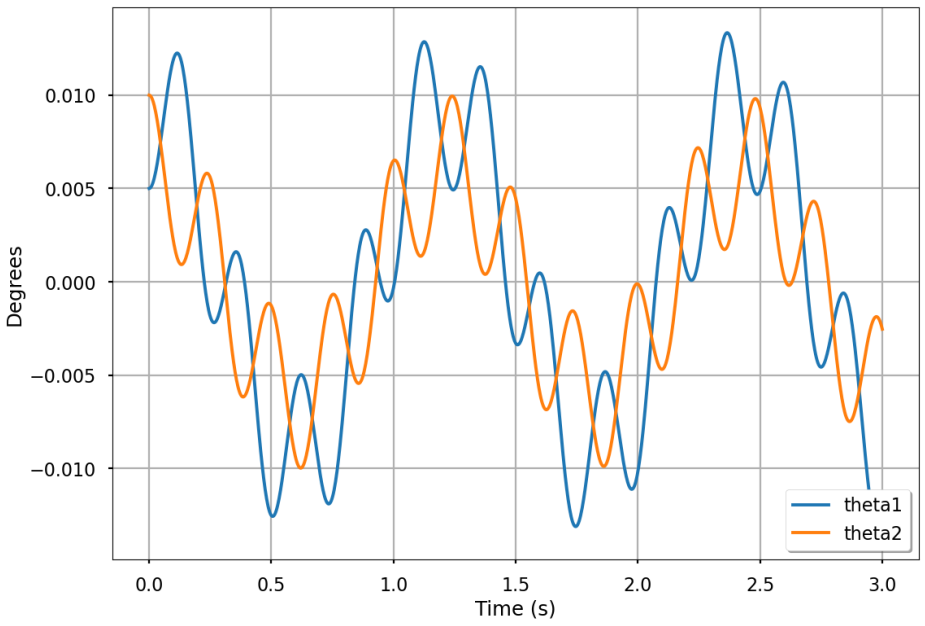


Figure 5.9: Inverted double pendulum generalized coordinates. Increased stiffness

5.4 Training & Testing the machine learning models

As previously mentioned, we decided to explore three different neural network types. Our initial tests were purely trained on historical data. In this phase we attempted to use two outputs of the network where the two outputs would be BMx and BMy. As inputs we used various data that seemed to have a decent correlation.

In the later tests we predicted only one axis at a time. Because of the MRMR and F-Tests analysis, we quickly went away from using BM from the COB1 sensor as input data to the network. Instead we relied heavily on the acceleration and rotational velocity data from the SMUs.

In the last phase of the project we relied on the inclination from the Kalman filter (See 4.5.2). It is also in the later phases of the project that we moved to training on OrcaFlex data. Validation in that phase is still done using historical data.

We used multiple methods to validate our predictions. Along with directly comparing the real and predicted BM through plots, the error function R^2 was used as a general estimator to compare the similarities, which can be seen in (5.14) [64].

$$\mathrm{R}^2 = 1 - \left(\frac{\sum_{i=0}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{N} (y_i - \bar{y})^2} \right) \tag{5.14}$$

We used our damage assessment component to compare the damage rates and accumulated fatigue calculated by the real and predicted BMs, which is the most important form of validation we did on our predictions.

Detrending the data

We noticed that the BM trends appeared over long periods of time, seen in fig. 5.10.

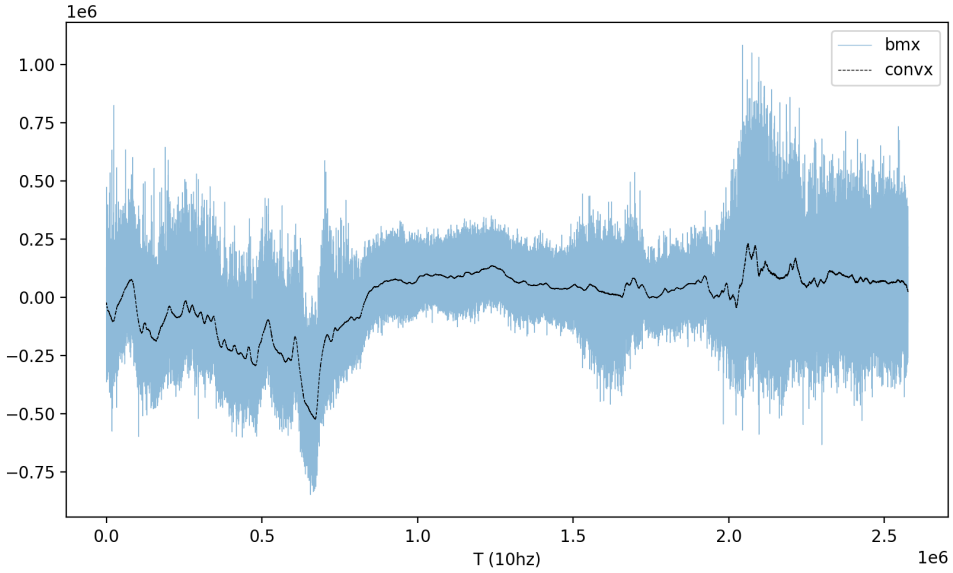


Figure 5.10: Long-term trend on historical BMx data. convx being the average mean.

This was a struggle during our early ML models, as they failed to predict the long-term changes in the system. We then explored the effects detrending had on the fatigue. This could be concluded to be insignificant, seen in fig. 5.12. Thus we started detrending the BMs before used in training, to hopefully increase the accuracy of the predictions.

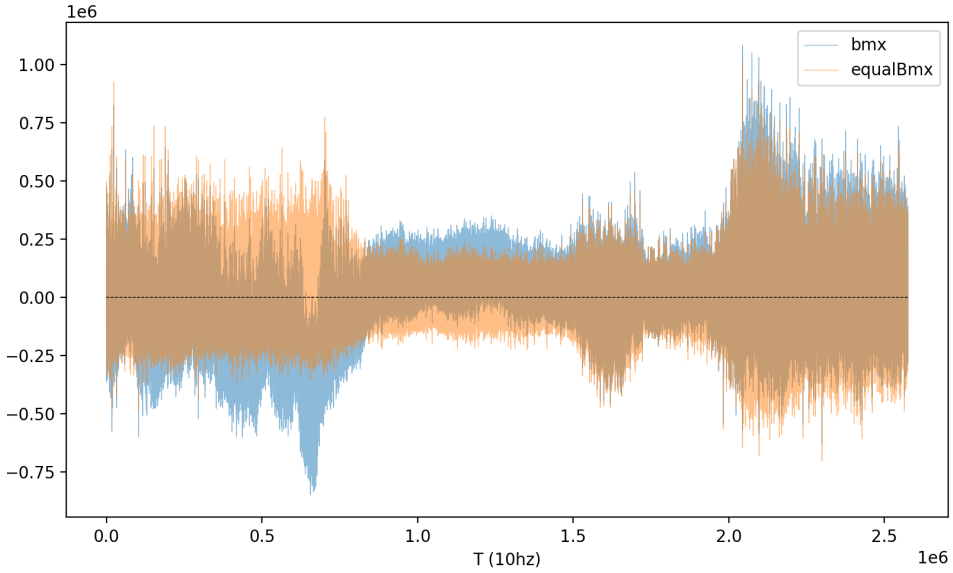


Figure 5.11: Detrended vs normal BMx

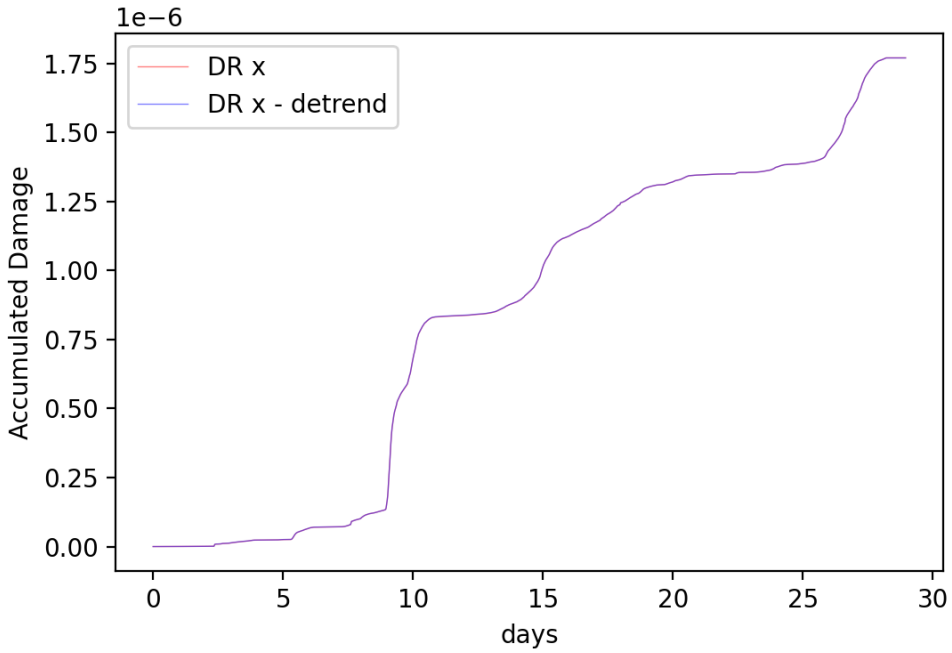


Figure 5.12: Fatige from detrended BMx vs normal BMx

Frequency adjustment and directional adjustment

As we got poor results when training on simulated data, we noticed was that some of the directions were inverted compared to historical data. Our ML model reflected this well, because its prediction of the trend was always inverted. In the latest versions of the OrcaFlex model we inverted the BM in both directions, in addition to inclination and rotational velocity in the x-direction.

We also noticed that the frequency of the simulated signals were significantly larger. This could suggest that the OrcaFlex data is too sensitive compared to the historical data. In a case where we had a ANN model that has no kind of history or memory, this should not make much difference, as long as the correlation between the input data and output data remains the same. However, we are using models with history or memory. This implies that our models are training on data that is constantly and frequently jumping up and down from sample to sample, and then validating on data where the data is steadily moving up and down in a wavelike motion.

We applied a Butterworth filter followed up by detrending for very low frequencies. We applied this to the BM and inclination from all the SMU sensors. The parameters for the filtering was tuned compared to the historical data. After the frequency changes, we compared the new BM against the inclination to verify that the correlation was not removed in the process.

In the analysis above we only looked at correlation in data. A full analysis on the physical aspects compared with the simulation was later done (See 4.2.3)

Scaling

We also normalize our input and output to be between -1 and 1. This is done by multiplication, so the predicted BM can easily be scaled with a constant to get it back up to the same range as the original BM.

Chapter 6

Results

The main objective of this project was to find an indirect method of measuring the BM. The fusion filters showed promising results, even though the mathematical model should have yielded better results than the fusion filters alone, but could not contribute to the corrected angles due to little information about the system.

One of the three ML models we trained proved successful when trained on simulated data, while another did well on historical data. The use of Kalman filters also proved to be beneficial regarding prediction accuracy.

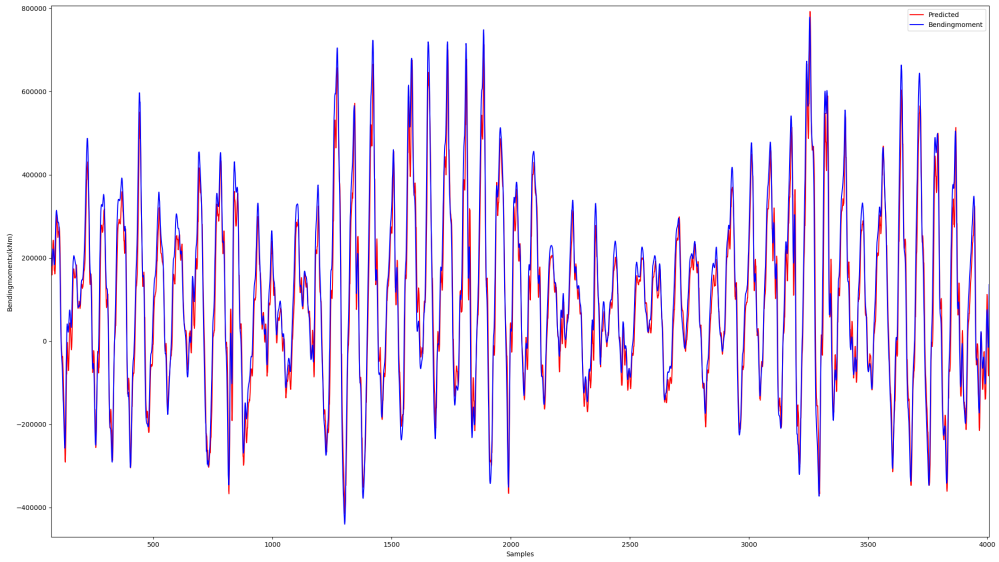
6.1 Fusion filter results

The analysed and processed data plays a crucial role in the ML model and needs to be compared with what we are trying to estimate. The analysis from section (5.2) will be used to see their correlation to the BM.

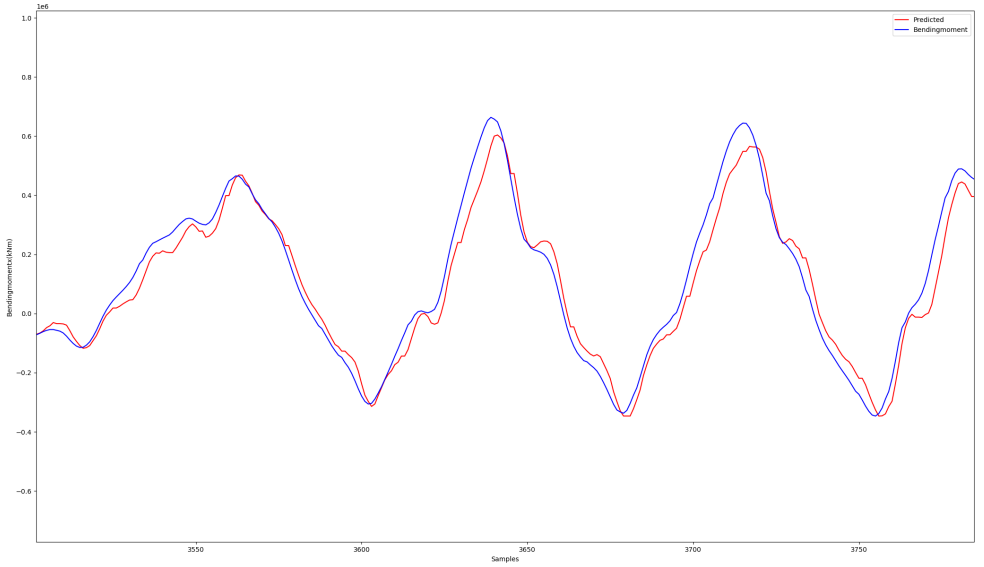
From the previous chapter, it was clarified that the Tilt fusion filter had the best score compared to the BM for when the RFJ was on and off. There is of course a difference between the inclination and the BM, but they are directly correlated since both are affected by the difference in angles.

6.1.1 R^2 score with RFJ off

We can compare the inclination measurements from the Tilt fusion filter with fig. 6.1 to see the correlation in time and use the familiar R^2 score from section 5.4 to test our filtered data. In table 6.1 we can see their score against each other.



(a)



(b)

Figure 6.1: Tilt measurements versus bending moment. y-axis is bending moment in Nm, x-axis is time samples

Table 6.1: R^2 score RFJ off

Features	SMU1	SMU2	SMU3
Complementary	0.33	0.30	0.31
EKF	0.85	0.24	0.68
KF	0.82	0.56	0.89
KF_$\dot{\theta}$	0.02	0.03	0.03
KF_$\ddot{\theta}$	0.16	0.02	0.08
Integration	0.08	0.09	0.11
Madgwick	0.84	0.03	0.84
Mahony	0.16	0.12	0.15
Original Inclination	0.38	0.62	0.90
Tilt	0.96	0.69	0.96

Every filter was scaled up with a factor of 2.3 million for best fit. This parameter could be found mathematically with the moment of inertia formula for mass times length. From the scores listed in table 6.1, the fusion filter had R^2 scores upwards of 0.96 with the BM.

In fig. 6.2 we can see the score from the Tilt fusion filter from SMU3.

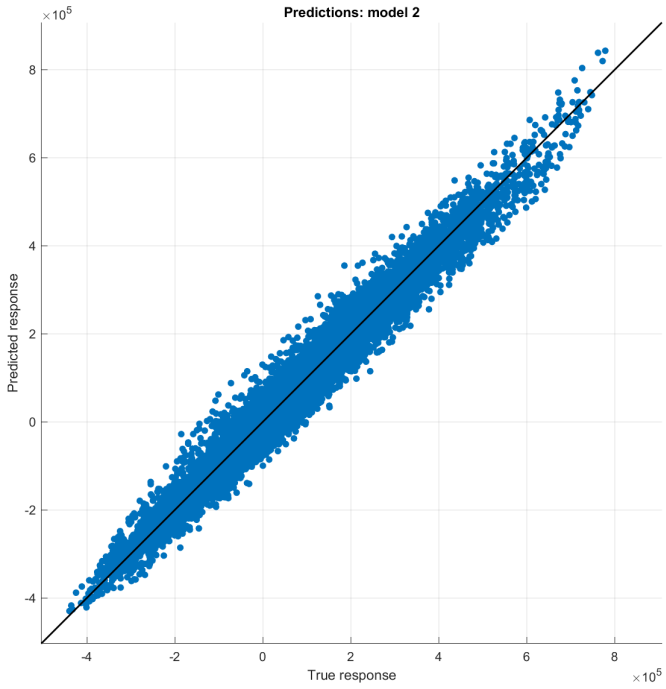


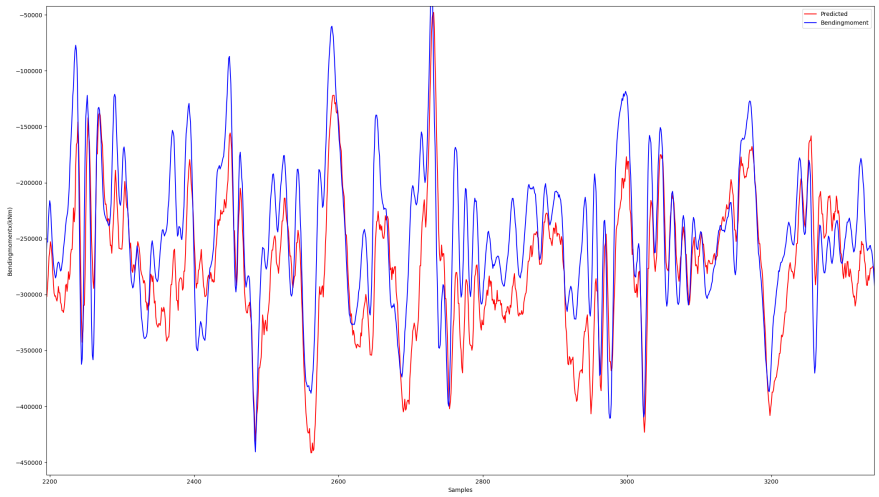
Figure 6.2: R^2 score from Tilt 3 filter, RFJ off

We will now look at the more complex part of the problem – when the system experiences a more non-linear trend.

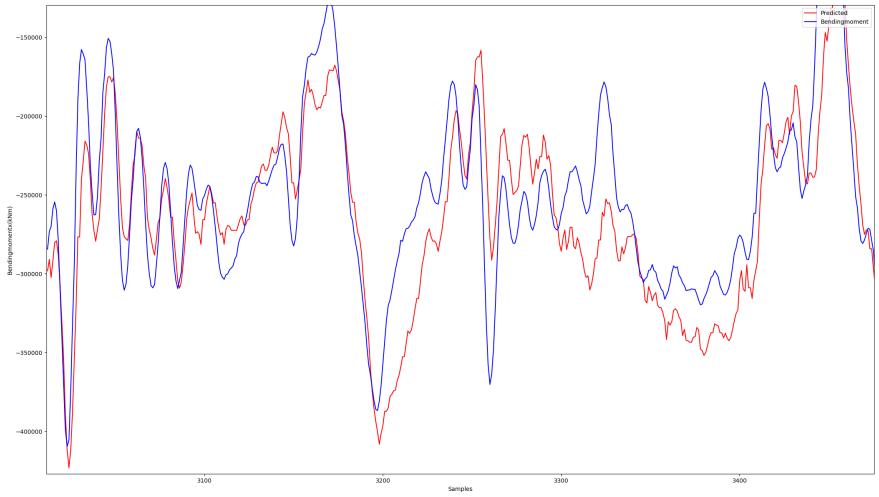
6.1.2 R^2 score with RFJ on

As discussed many times in this project, the enabled RFJ puts the system under a much more non-linear motion, therefore the scores will drop dramatically in this test. In the further work section 8.2 an even better approximation with the Lagrangian mechanics are found.

In fig. 6.3 we can see the correlation from the Tilt filter to the BM.



(a)



(b)

Figure 6.3: Tilt measurements and bending moment. y-axis is bending moment i Nm, x-axis is time samples

We will use the same ranking system as last time to test the filtered data. In table 6.2 their scores are presented.

Table 6.2: R^2 score RFJ on

Features	SMU1	SMU2	SMU3
Complementary	0.08	0.20	0.23
EKF	0.44	0.34	0.53
KF	0.63	0.70	0.64
KF$\dot{\theta}$	0.00	0.00	0.07
KF$\ddot{\theta}$	0.13	0.15	0.10
Integration	0.01	0.02	0.08
Madgwick	0.00	0.10	0.43
Mahony	0.38	0.46	0.51
Original Inclination	0.26	0.39	0.57
Tilt	0.42	0.56	0.72

Every filter was scaled up with a factor of 1.15 million for best fit. Note that the difference in BM are halved from when the RFJ was off. This parameter could also be found mathematically with the moment of inertia formula for mass times length, but also needs an expression for the damping in the elastomer in the RFJ. We can see the R^2 score for the scaled up Tilt filter signal in fig. 6.4.

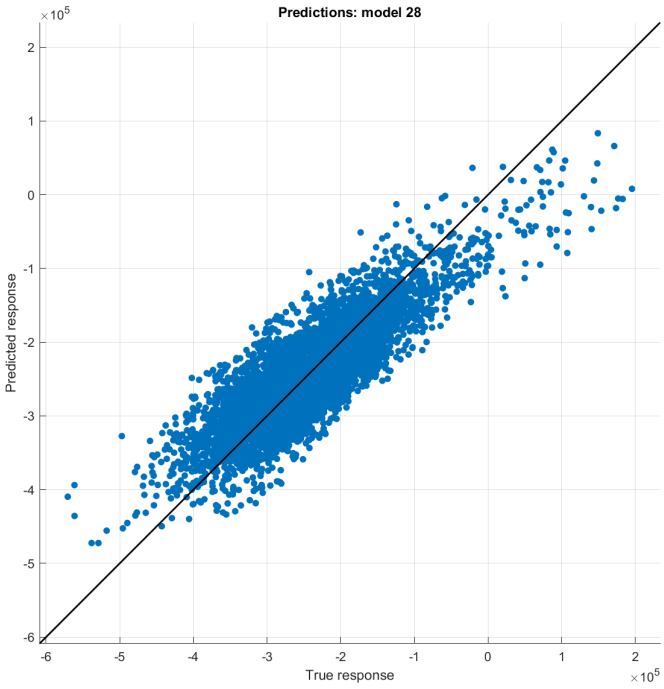


Figure 6.4: R^2 score from Tilt 3 filter, RFJ on

We can clearly see that the Tilt fusion filter has the best correlation with the BM, but the linear Kalman filter has the best overall performance. This is promising for the implementation of a process model from the Euler-Lagrange equations.

6.2 Mathematical model

The mathematical model is based on the equations of motion derived in section (4.6.6). To summarize, the Lagrangian process model was derived to be the heart of the system, while the sensor readings was the input to make a fusion between the process model and the signals at hand.

6.2.1 Kalman results

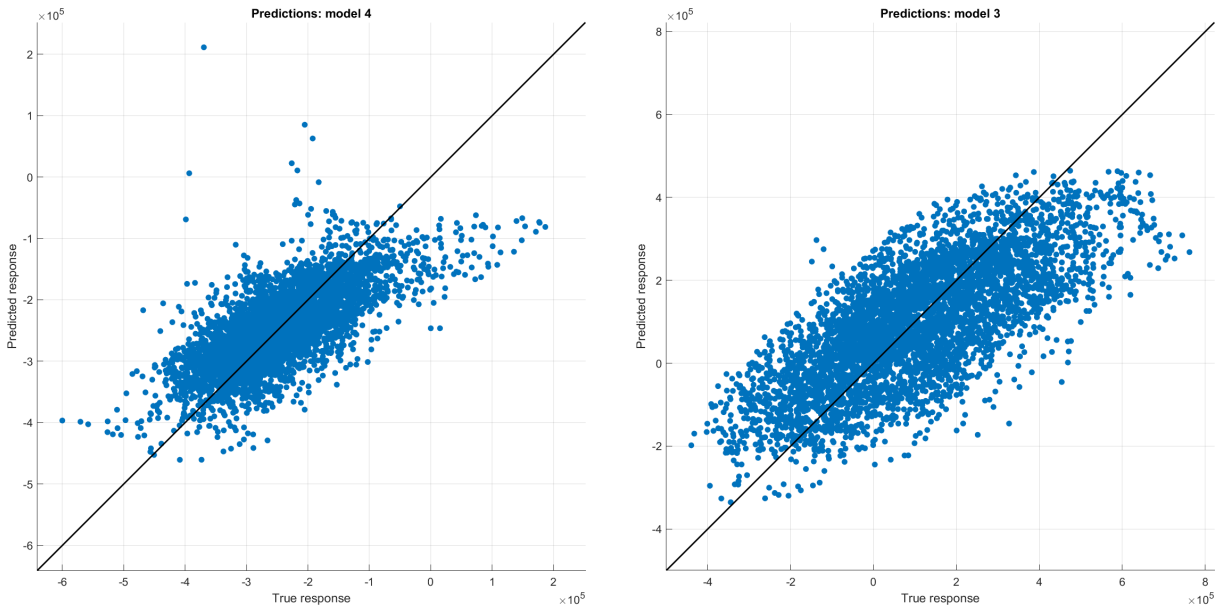
With little information about the system, we can only approximate process noise, weights, lengths etc. The system derived in section 4.6.6 gave results shown in table 6.3

Table 6.3: R-score Mathematical model

Features	SMU1	SMU2	SMU3
Kalman with process RFJ on	0.10	-	0.51
Kalman with process RFJ off	0.15	-	0.63

Note that the system derived is for a double pendulum and the SMU2 sensor is not needed to estimate the new angles.

We can see the scores in fig. 6.5a when the RFJ was on and fig. 6.5b when the RFJ was off.



(a) RFJ on

(b) RFJ off

Figure 6.5: Lagrangian process

The result here are not good, and much more information about the system is needed for this to give a great result. We will see in the further work section (See 8.2) how adding a non-linear damper to the system will massively help improve the correlation between the estimations.

6.3 Machine learning model

This section will cover the results from our ML models read about in section 4.4 and 5.4.

6.3.1 Feed-Forward results

The results from the FNN were as expected. The original training of the network gave a R^2 score of 0.72 when predicting the BM in the x-direction. The trained model was tested over one day. A R^2 score of 0.72 is decent considering the fact that FNNs are more generalized ANNs.

It is important to note that when validating on other days of the same operation the R^2 score went below 0.57. This means that the model is fitted to a specific day and specific sea state, and struggles when presented with different conditions.

When the network was trained on simulated data from OrcaFlex and tested against historical data, there was a discrepancy in the R^2 scores compared to when trained on historical data. The score goes bellow -3.2. The accumulated fatigue and damage rate can be seen in fig. 6.6 and 6.7. The FNN predicts quite linear when looking at the fatigue over several days and it struggles to follow the trend of spikes.

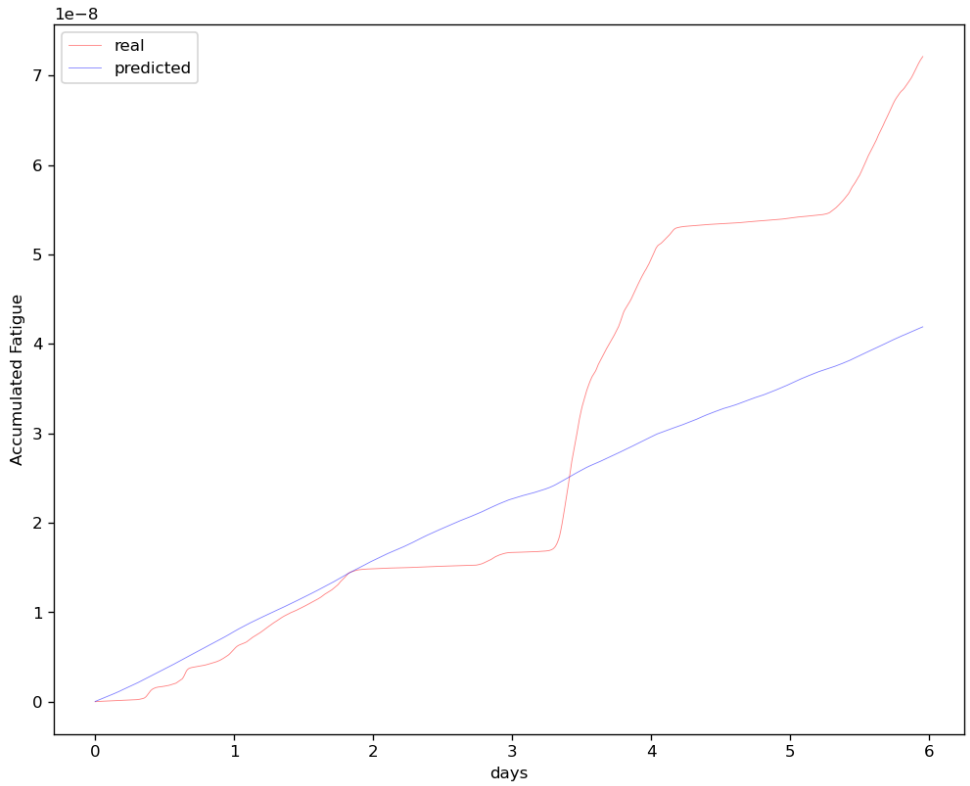


Figure 6.6: Comparison between historical(red), and predicted(blue) fatigue over 6 days.

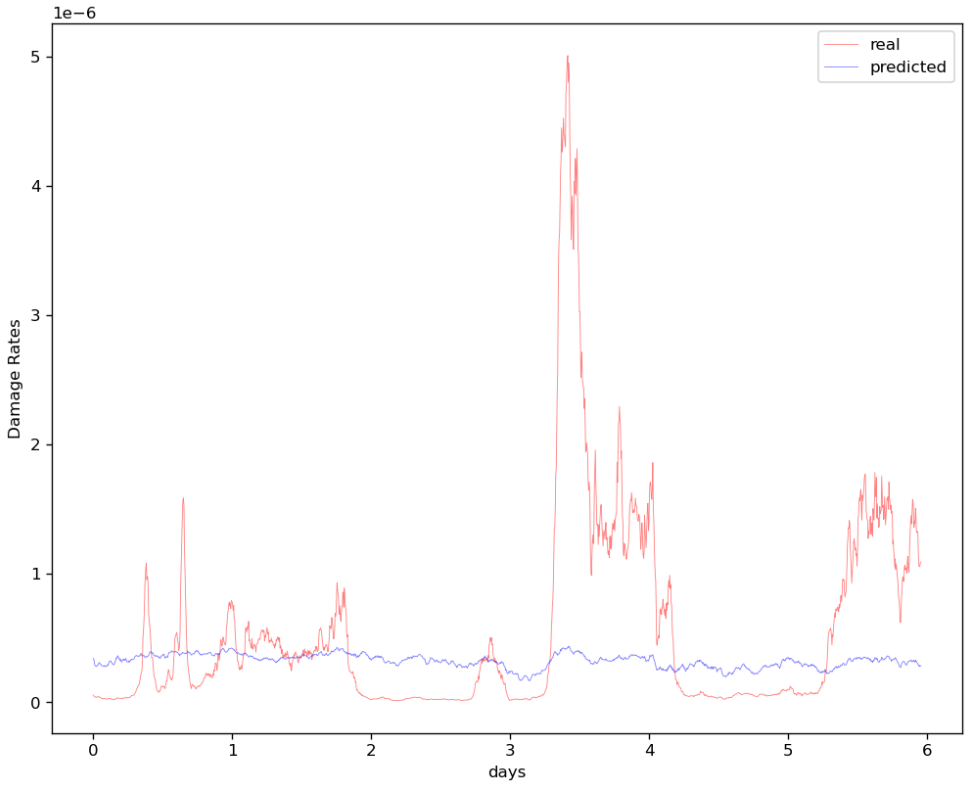


Figure 6.7: Comparison between historical(red), and predicted(blue) damage rates over 6 days.

As seen in fig. 6.7 the FNN struggles to follow the trend and sharp spikes. This might be a result of overfitting to certain conditions on the downscaled data. When training the FNN it tended to have one or two good epoch, then skyrocket or plummet. This is pointed out as "*with multivariable linear regression in high-dimensional spaces it is possible that some dimensions that is actually irrelevant appears by chance to be useful, resulting in overfitting.*" [2] In the results we gathered, the FNN tends to overfit quite often.

It is important to note that the group decided to create a FNN to have a more generalized network to test up against.

6.3.2 RNN results

RNN cell

In the first design, the RNN cell had a lot of issues with exploding gradients (See 4.4.2). We added a ReLU activation function on the data that came out of the hidden layer. By doing so, the gradients stabilized for lower learning rates. R^2 scores up to 0.9 can be achieved in certain weather conditions when the RFJ is enabled, and the model is trained on historical acceleration and rotational velocity. However, it quickly got overfitted to whatever data it was training on.

The same happened when we moved to training on OrcaFlex data. Validations on historical data resulted in a predicted straight line. This was during the time where we had not done the frequency changes as explained in 5.4 which might explain this behaviour.

RNN deeper

By adding a few linear layers after the RNN cell the results improved. With a deeper network we could apply Tanh activations between the layers. We were also able to put dropout between layers to reduce the overfitting. Doing so we then saw better results when validating on the historical data. It was also while testing this network that we did the frequency adjustments which improved the performance even more.

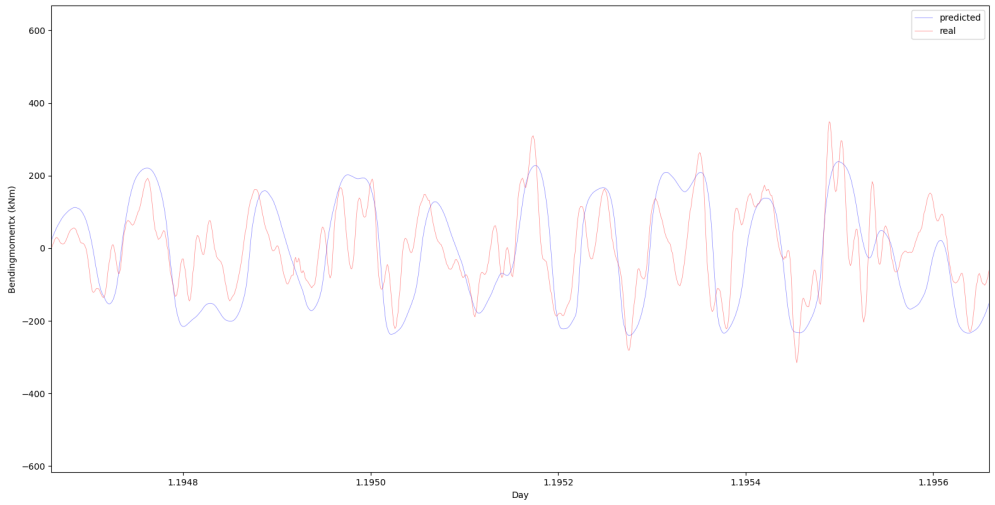


Figure 6.8: RNN Deep network trained on OrcaFlex data, validated on historical data. y-axis is bending moment (kNm)

Seen in fig. 6.8 the model started to follow the mid-range frequencies to some degree. This is the first time we saw some kind of result on the RNN model trained on OrcaFlex data. However, the results were far from sufficient. It totally missed the lower range frequencies and the higher range frequencies. A validation of this model over six days with historical data can be seen in appendix fig. C.4.

RNN residual

The residual network ended up as a faster learning version of the the deeper RNN network. When it comes down to the results the performance was much the same.

LSTM

With the LSTM the performance improved. As seen in fig. 6.9 the predictions is following the historical data to some degree. It is unable to predict the larger amplitudes of the signal. For this reason the accumulated fatigue from the signal is unusable as it is the largest amplitudes that gives the highest damage rate. This is proved in fig. 6.10 where we can see an accurate prediction until we reach the higher amplitudes of the signal, where the accumulated fatigue starts to drift off.

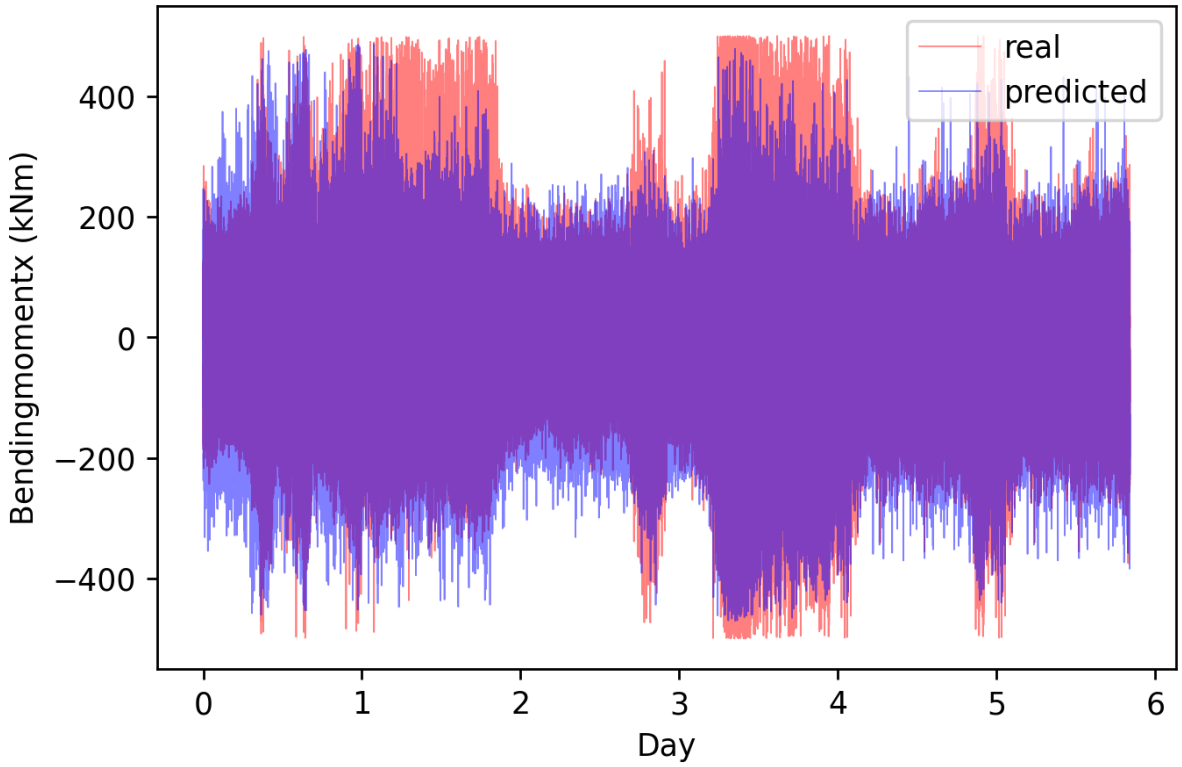


Figure 6.9: LSTM network trained on OrcaFlex data, validated on six days of historical data

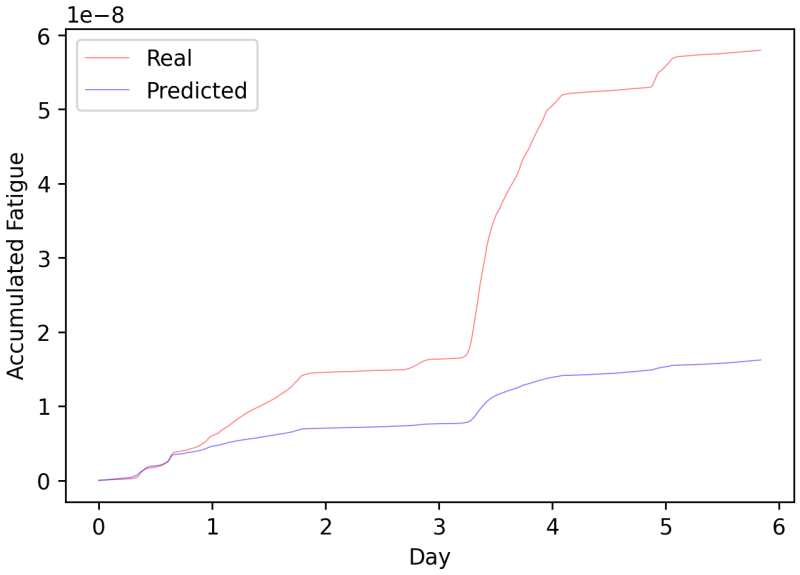


Figure 6.10: LSTM network trained on OrcaFlex data, validated on six days of historical data R^2 score : -0.35. For day 1 before the drift starts, the R^2 score is 0.96

Summary

The RNN has trouble training on OrcaFlex data and then predicting on historical data. However, when training and predicting on historical data the RNN predictions got R^2 scores between 0.8 and 0.95. In fig. 6.11 we can see that the model predicts the higher frequencies well.

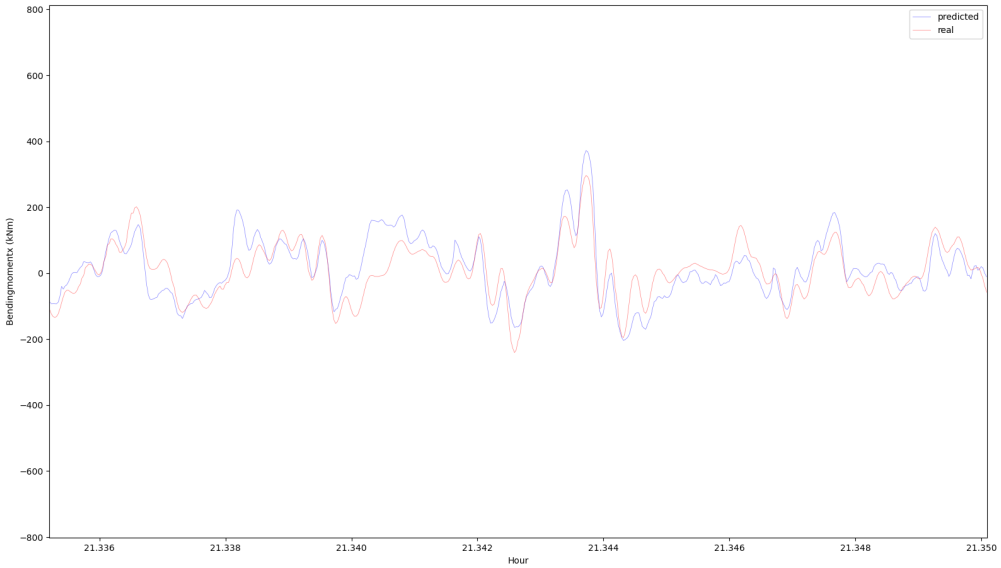


Figure 6.11: RNN trained on historical data and predicting unseen historical data. y-axis i bending moment (kNm), x-axis is time

6.3.3 CNN results

We first tested a small network which consisted of two conv-1d layers, and two fully connected linear layers. The input to this network was the acceleration and rotational velocity from all three SMU sensors. The approach was discarded as it lacked any valuable results when it came to training on OrcaFlex data; most predictions ended up as spikes. A problem with overfitting appeared when trained on historical data. Various attempts at activation functions were tried without much luck. We therefore moved on to a more advanced design (See 4.4.3.1).

The results gained by the new design is indeed promising, as it was only trained by simulated OrcaFlex data. The model was able to predict the BMx over several days with a promising accuracy, both with the RFJ enabled and disabled.

Six days of historical sensor data gathered from the Askeladd operation was used to verify the model. The network is specifically trained to predict the BMx, using the inclination and rotational velocity from our Kalman filters as the input, note that the only data used in the Kalman filter was gathered from SMU1. All data used in the model are scaled down and normalized beforehand. The scaling can easily be reversed as this is done by simple multiplication. The BMs was also detrended beforehand, as discussed in (5.4).
The network require a historical buffer size of 64 seconds. After that it can simply move the buffer one time step forward for each new sensor reading; in other words we use a sliding window with the step size equal to rate of data at 0.1 seconds (10hz). The results are stable over the periods we have tested it. The comparison between the historical and predicted BMx over the six day period can be seen in fig. 6.12.
The model in question was to our surprise trained using unbalanced data. It was however preprocessed in an unusual way, further discussed in chapter 7.

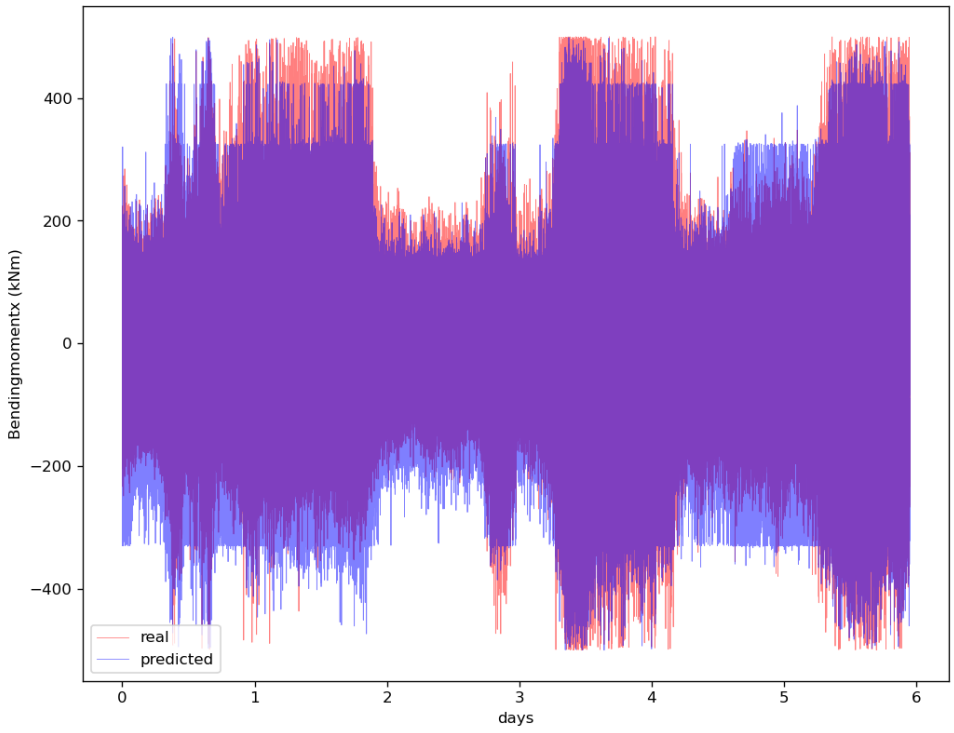


Figure 6.12: Predicted and historical BMx comparison over 6 days, where the model is purely trained on simulated data.

Note that the RFJ was enabled and disabled multiple times during the first day. It was fully enabled between day 2 and 3, which is where this model performs best, meaning the model is able to predict the BMx of a non-linear system to a certain degree of accuracy.
To further verify these results, we calculated the accumulated fatigue and damage rates based on the predicted BMx. We then did the same for the historical data during the same days and compared the two, the results can be seen in fig. 6.13 and 6.14.

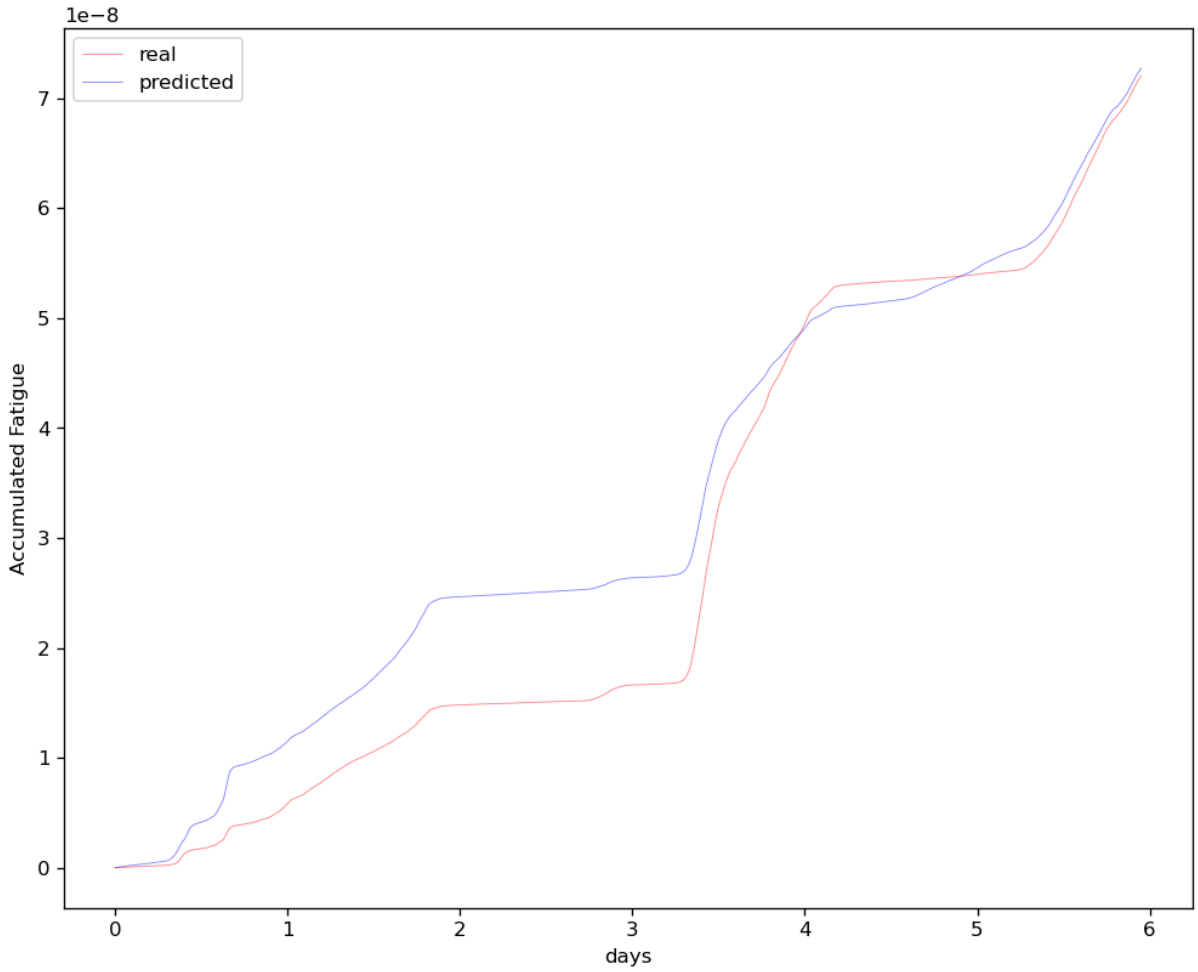


Figure 6.13: Fatigue compared, 6 days. $R^2 = 0.93$

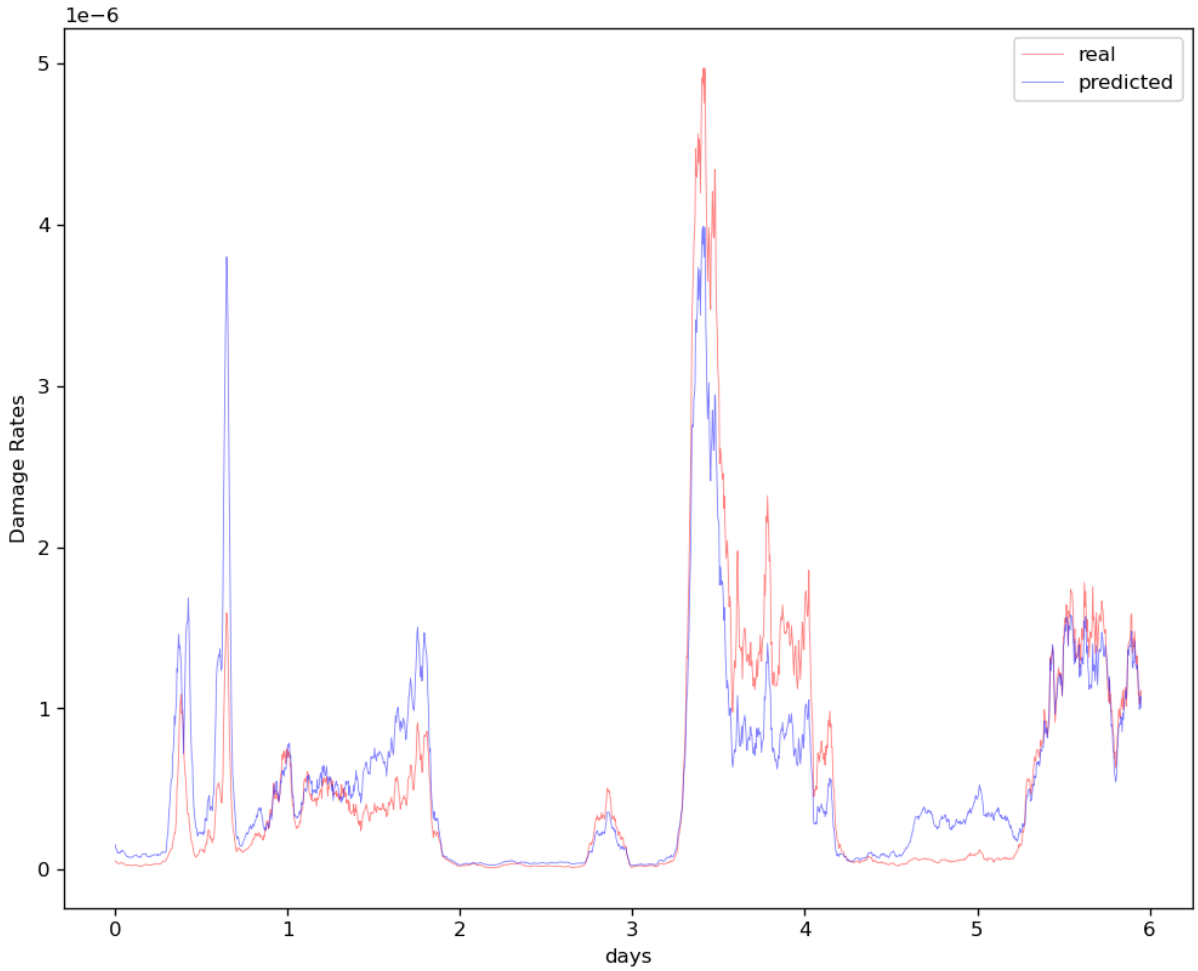


Figure 6.14: Damage rates compared, 6 days.

Chapter 7

Discussion

Our results yields high correlation with regards to BM, and can be seen as a promising way to approach the problem.

Higher R^2 scores between the BM and the original inclination measurements were found, and the Kalman filter showed its versatility in finding a better inclination measurement than the preprocessed data. The most surprising result was the simple filter fusion of just the accelerometer, giving R^2 scores upwards of 0.96 when the system experienced a more linear trend. Note that these scores were taken over a much shorter time period than the ML models and will vary under different conditions.

The linear Kalman filter showed the greatest overall score with respect to fusion filters, and proves yet again that the best linear estimator is the optimal way to go.

The Kalman filter could be expanded with the equations derived in the further work section to obtain a linear equation of the system at small angles (See 8.2). A favourable approach from this study would suggest a ML model that is set to find the spring, damper, and process noise matrix in the future. At the same time, it should be looking at a non-linear approach to the observation covariance matrix, where the normal distribution can change under different velocities and accelerations.

There are also numerous papers done regarding the accuracy of MEMS, such as [68] and [69] that specifies the limitation of the sensors. If the cost for sensors are limitless, the use of ring laser gyroscopes and more accurate accelerometers are preferable. If there are no limitation to energy consumption, a higher frequency accelerometer combined with a vibrational accelerometer would help identify the vibrations experienced by the system, while simultaneously improving the angular acceleration data that could be used in the proposed Kalman filter.

We also saw that undersampling the data helped reduce overfitting to some degree when training on historical data. However, there is room for improvement on the clustering methods. This approach did not help to reduce overfitting when training on OrcaFlex data, as variance lays in different simulation setups. An easier solution would be to build datasets on smaller simulations with varying environment configurations. Undersampling with classification could be beneficial if more accurate weather data is used in the simulations.

A regular FNN had difficulty with learning the non-linear trends of the system, but could be used with the proposed mathematical model in another study to explore its possibilities.

Removing some of the higher frequencies in the training data from OrcaFlex seemed to give better results when validating on historical data. The untouched simulated data should be mathematically correct, but if the correlation between the input and the output is too good, the ML model will adjust itself to rely on the wrong data. On historical data, the inclination from the Kalman filter

did not provide a sufficient correlation when the RFJ was enabled. However, a trained model would still rely on that correlation which no longer exists within the data. By modifying the frequencies, we removed some of the reliability in the process and forced it to rely more on the input data.

A major bug had managed to go undetected in the script where simulated data was merged to form whole days. The problem was fixed, and a network was trained to test the new findings. The bug might have been a blessing in disguise after all, as it sorted the merged signals in a specific pattern, which we mistakenly took as a bi-product of the OrcaFlex simulation. It was treated as noise, and filtered to share characteristics with the historical data. The altered output was a signal where the periods were consecutively intertwined, beginning with the first data point from all eight simulations, continuing with the second and so on. The proposed CNN managed to extract useful information when trained on data affected by the bug, reflected by our results. Further research would be interesting regarding this accidental method of dataset creation, as it might be a reason why the network managed to predict the non-linear behaviour so well.

The data produced from the OrcaFlex model might have some varying degrees of inaccuracies. The indirect methods used for extracting parameters (See 4.2.4), might be prone to inaccuracies (See 4.2.2). Even after the quality checks on preservation of the original model, there could be errors in these calculations. The axis in the model might be incorrect compared to those that WAMS used [19]. The TFMC provided OrcaFlex model could have inaccurate parameters implemented, missing parameters or build errors.

To summarize, the Kalman filter has proven to be effective in finding new inclination measurements, both with and without a process model. The fusion between a mathematical model and more accurate motion sensors can help to predict the BM with higher accuracy.

We argue that training a CNN on inclination and rotational velocity from simulated operations can be used as an indirect method to find the BM of a subsea WH when the RFJ is enabled.

The CNN and RNN have both proven effective in their own ways, and a combination of the two could be beneficial to explore in future research. The same goes for the simulated data affected by a bug that changed the data structure used in training. The accidental method have to our surprise yielded better results than the intended version when used to train ML models to predict non-linear behaviour over time.

Chapter 8

Further work

In a project such as this, time will always be a constraint. There are several avenues which we did not explore, either due to prioritizing, or because it was discovered too late.
Therefore, we have outlined the most important roads to explore for future projects wanting to build on the foundation of our research.

8.1 Damping

The RFJ has a non-linear behaviour due to the elastomer which can be modelled as a damping force. For our system, shown in fig. 4.28, the damper force can be expressed as

$$F = b\dot{\theta} \tag{8.1}$$

Integrating to find the energy dissipation due to viscous damping

$$R = \frac{1}{2}b(\dot{\theta}_2 - \dot{\theta}_1)^2 \tag{8.2}$$

We expand the Euler-Lagrange equations to include the damper force (8.2)

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_1}\right) - \frac{\partial T}{\partial \theta_1} + \frac{\partial R}{\partial \dot{\theta}_1} + \frac{\partial V}{\partial \theta_1} = \sum F_q \tag{8.3}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta}_2}\right) - \frac{\partial T}{\partial \theta_2} + \frac{\partial R}{\partial \dot{\theta}_2} + \frac{\partial V}{\partial \theta_2} = \sum F_q \tag{8.4}$$

where $\sum F_q$ is the sum of forces corresponding to the loads at each coordinate [70, p.3]

The Euler-Lagrange equations (8.3,8.4) derived from our original Lagrangian (4.132) and the energy dissipation due to viscous damping (8.2) [70, p.3], becomes (8.5, 8.6) (See appendix D.4 for the complete derivation)

$$-b\dot{\theta}_1 + b\dot{\theta}_2 + gl_1m_1\theta_1 + gl_1m_2\theta_1 + k_1\theta_1 + k_2\theta_1 - k_2\theta_2 + l_1^2m_1\ddot{\theta}_1 + l_1^2m_2\ddot{\theta}_1 + l_1l_2m_2\ddot{\theta}_2 = \sum F_q \tag{8.5}$$

$$b\dot{\theta}_1 - b\dot{\theta}_2 + gl_2m_2\theta_2 - k_2\theta_1 + k_2\theta_2 - l_1l_2m_2\ddot{\theta}_1 + l_2^2m_2\ddot{\theta}_2 = \sum F_q \tag{8.6}$$

8.2 Kalman fusion

Using the Euler-Lagrange equations including the damper term (8.5,8.6) we can solve for the angular acceleration $\ddot{\theta}$ and expand on the expression for the Kalman filter (4.138, 4.139)

$$\ddot{\theta}_1 = \frac{bl_1\dot{\theta}_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{bl_1\dot{\theta}_2}{l_1^2l_2m_1 + 2l_1^2l_2m_2} + \frac{bl_2\dot{\theta}_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{bl_2\dot{\theta}_2}{l_1^2l_2m_1} + 2l_1^2l_2m_2 - \frac{gl_1l_2m_1\theta_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{gl_1l_2m_2\theta_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} + \frac{gl_1l_2m_2\theta_2}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{k_1l_2\theta_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{k_2l_1\theta_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} + \frac{k_2l_1\theta_2}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{k_2l_2\theta_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} + \frac{k_2l_2\theta_2}{l_1^2l_2m_1 + 2l_1^2l_2m_2} \tag{8.7}$$

$$\ddot{\theta}_2 = \frac{bl_1^2\dot{\theta}_1}{l_1^2l_2^2m_1 + 2l_1^2l_2^2m_2} - \frac{bl_1^2\dot{\theta}_2}{l_1^2l_2^2m_1 + 2l_1^2l_2^2m_2} + \frac{bl_1\dot{\theta}_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{bl_1\dot{\theta}_2}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{b\dot{\theta}_1}{l_2^2m_2} + \frac{b\dot{\theta}_2}{l_2^2m_2} - \frac{gl_1^2m_1\theta_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{gl_1^2m_2\theta_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} + \frac{gl_1^2m_2\theta_2}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{g\theta_2}{l_2} - \frac{k_1l_1\theta_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} - \frac{k_2l_1^2\theta_1}{l_1^2l_2^2m_1} + 2l_1^2l_2^2m_2 + \frac{k_2l_1^2\theta_2}{l_1^2l_2^2m_1 + 2l_1^2l_2^2m_2} - \frac{k_2l_1\theta_1}{l_1^2l_2m_1 + 2l_1^2l_2m_2} + \frac{k_2l_1\theta_2}{l_1^2l_2m_1 + 2l_1^2l_2m_2} + \frac{k_2\theta_1}{l_2^2m_2} - \frac{k_2\theta_2}{l_2^2m_2} \tag{8.8}$$

Using the same state variables as found in section 4.6.6.2, we can find the new state vectors to become (8.9)

$$\begin{pmatrix} \mathcal{O} = -\frac{gl_1l_2m_1}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{gl_1l_2m_2}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{k_1l_2}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{k_2l_1}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{k_2l_2}{l_1^2l_2m_1+2l_1^2l_2m_2} \\ \mathcal{P} = \frac{bl_1}{l_1^2l_2m_1+2l_1^2l_2m_2} + \frac{bl_2}{l_1^2l_2m_1+2l_1^2l_2m_2} \\ \mathcal{Q} = \frac{gl_1l_2m_2}{l_1^2l_2m_1+2l_1^2l_2m_2} + \frac{k_2l_1}{l_1^2l_2m_1+2l_1^2l_2m_2} + \frac{k_2l_2}{l_1^2l_2m_1+2l_1^2l_2m_2} \\ \mathcal{R} = -\frac{bl_1}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{bl_2}{l_1^2l_2m_1} \\ \mathcal{S} = -\frac{gl_1^2m_1}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{gl_1^2m_2}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{k_2l_1^2}{l_1^2l_2^2m_1+2l_1^2l_2^2m_2} - \frac{k_2l_1}{l_1^2l_2m_1+2l_1^2l_2m_2} + \frac{k_2}{l_2^2m_2} \\ \mathcal{T} = \frac{bl_1^2}{l_1^2l_2^2m_1+2l_1^2l_2^2m_2} + \frac{bl_1}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{b}{l_2^2m_2} \\ \mathcal{U} = \frac{gl_1^2m_2}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{g}{l_2} + \frac{k_2l_1^2}{l_1^2l_2^2m_1+2l_1^2l_2^2m_2} + \frac{k_2l_1}{l_1^2l_2m_1+2l_1^2l_2m_2} - \frac{k_2\theta_2}{l_2^2m_2} \\ \mathcal{V} = -\frac{bl_1^2}{l_1^2l_2^2m_1+2l_1^2l_2^2m_2} - \frac{bl_1}{l_1^2l_2m_1+2l_1^2l_2m_2} + \frac{b}{l_2^2m_2} \end{pmatrix} \tag{8.9}$$

The state variables could be set into the state space equation as seen in (8.10)

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \mathcal{O} & \mathcal{P} & \mathcal{Q} & \mathcal{R} \\ 0 & 0 & 0 & 1 \\ \mathcal{S} & \mathcal{T} & \mathcal{U} & \mathcal{V} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{8.10}$$

Using the same process noise, Kalman gain and estimator from section 4.5.2, the R^2 score for the mathematical model can be seen in fig. 8.1

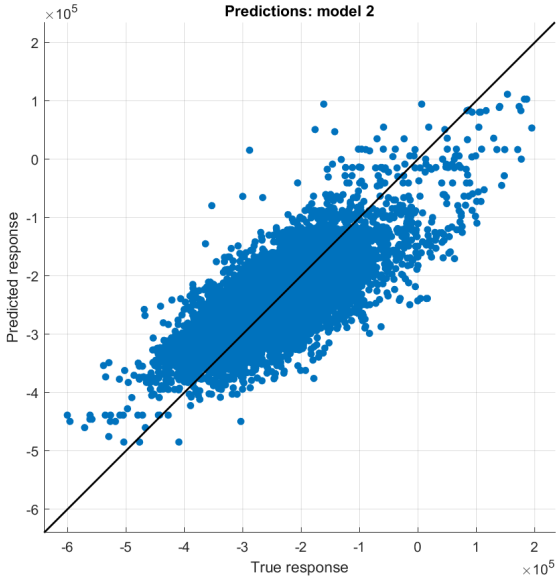


Figure 8.1: R^2 score from damped system with Kalman filter, RFJ on

That yields a R^2 score of 0.71. The mathematical model is built on the inclination measurements from the Kalman filter and has lifted the score with 0.07. Note that the damper coefficient and process noise matrix are just tested, and the system may be non-repeatable over different periods of time. A network that could alter the spring and damper coefficients with the process noise matrix, is a proposed method that should yield better results than just the inclination measurements alone.

8.3 Mass moment of inertia

Instead of simplifying with concentrated mass points, we could express its impact on the systems behaviour more accurately by calculating its moment of inertia. This approach would distribute the mass in a more realistic way as it also takes the rotational energy into account.

Using the polar moment of inertia we express the kinetic energy of in our inverted double pendulum shown in fig. 4.28 with more complexity (where $I_{com} = mr^2$)

$$T = \frac{1}{2}mv^2 + \frac{1}{2}I_{com}\omega^2 \tag{8.11}$$

8.4 CRNN with LSTM

The suggested CNN model (See 6.3.3) is not perfect, and further work can be done to improve its accuracy. With access to only a few days of data where the RFJ is enabled, our test space was limited to roughly 15% of the variation in wave height, and energy during the validation process (2.6.2). The model can serve as a valuable stepping stone to further explore the use of 1D-CNNs in fatigue related tasks.
We believe that a Convolutional RNN with LSTM cells should be explored next. As mentioned in 6.3.2 and 6.3.3 both networks bring different advantages to the table. The RNN and its capability to follow the higher frequencies, and the CNN being able to discern features and predict shifts in amplitude over time.

8.5 Hamiltonian neural networks

In a paper about unsupervised Hamiltonian Neural Network (HNN), Greydanus et Al. suggests that *"Instead of crafting the Hamiltonian by hand, we propose parameterizing it with a neural network and then learning it directly from data."* [58].
We recommend exploring Hamiltonian neural networks to expand upon the simplified model introduced in this project.

8.6 Weather data

There is an option to buy even more accurate weather data, that could be directly implemented into the O2-system, and by that into the OrcaFlex environment. This would improve the accuracy on the simulations produced by a full-scale model, but it is not easy to predict how much. There is also an option to discard the full-scale model, and instead try to create a scaled-down model. This approach would implement forces gathered from historical sensor data, instead of creating a simulated environment. We cannot give a clear recommendation on which of these two solutions to try at this given time. The detailed hindcast solution can be easily implemented, while the scaled-down model will provide some challenges.

8.7 Scaled-down model

Our initial plan was to create a scaled-down model of the Askeladd J1 operation based of the full-scaled model, by cutting the model at the COB1 location – in the middle of the MRA – and applying a time-series of BMs and tension at the cutting point.
This could increase the accuracy of the model, as the environment is very simplified compared to reality. The solution we wanted to try first was to randomly select batches of forces from historical COB1 data. However, cutting the drill pipe would not result in correct model output.
Unfortunately we did not find a solution within the projects time-frame, and decided to focus on improving the full-scale model.

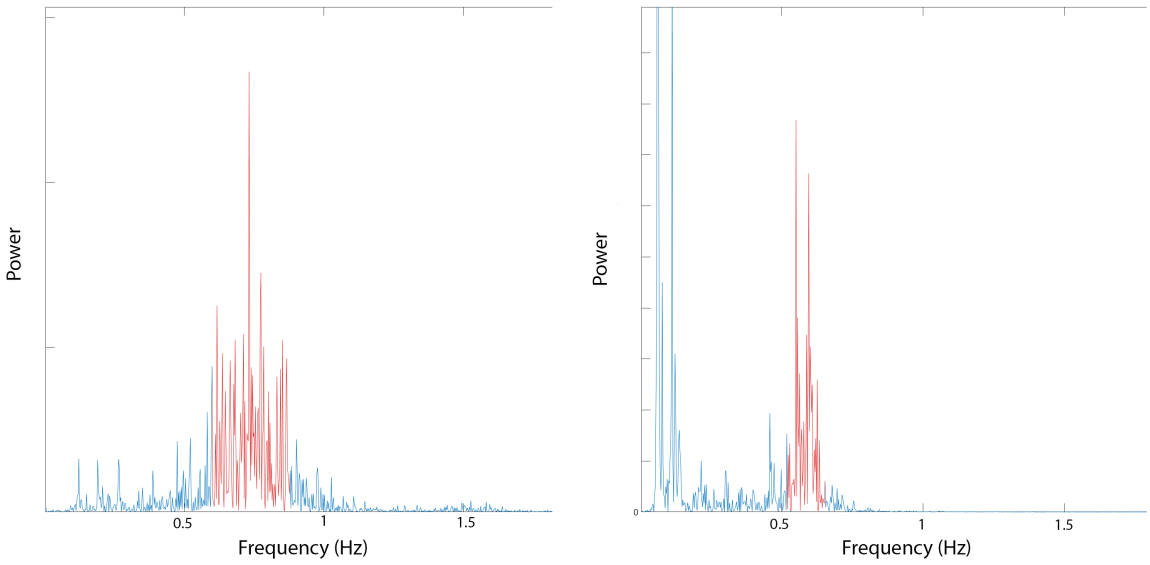
8.8 Final OrcaFlex iteration

An updated version of the OrcaFlex model was created, and simulations run by this iteration should be directly comparable to data from WAMS. There might be some imprecise input to our ML models due to the confusion regarding the axis in WAMS, the schematics of the model and OrcaFlex. Fig. 4.12 is our final understanding of this relation. In addition, the topics discussed in sections (4.2.3.1, 4.2.3 and 4.2.3.2)

8.9 Spectral density analysis

Towards the end of the project, we began testing with spectral density analysis as seen in fig. 8.2, and their normal distribution graphs can be seen in fig. B.14. Due to limited time, there were no changes implemented from these tests into the final OrcaFlex model. There was only time to compare the SMU2 accelerations. However, we got some interesting results, and further improvements to the model should be investigated. Spectral density graphs can be set up to be extracted directly from the O2-system, in the RES.

By comparing the two normal distribution plots in fig. B.14, its clear that the acceleration from the OrcaFlex data is more spread out, which is because the OrcaFlex model is less stiff than the actual operation. Differences in the model itself, or the seabed, may be the culprit. Comparing fig. 8.2a and fig. 8.2b confirms our suspicion. The red highlights should be equally distributed over the same frequency. This can be tweaked by changing the different stiffness parameters in OrcaFlex.



(a) Historical data

(b) OrcaFlex data

Figure 8.2: Spectral density graph

Chapter 9

Conclusion

It is reasonable to assume that indirect methods for measuring bending moments through reliable motion sensors to a large extent may replace strain sensors. The overarching goal is to extend the lifespan of subsea wellheads, through accurate and reliable measurements of bending moments to be used in fatigue calculations.

The findings presented in this report indicate that a convolutional neural network trained on simulated data from OrcaFlex performs remarkably well in predicting bending moments, when validated against historical inclination and rotational velocity processed through a Kalman filter. These results can to a large extent be taken as a proof of concept that an indirect method can be found when training on an OrcaFlex model.
In contrast, recurrent neural networks did not yield good results when trained on simulated data. It did, however, show very good correlation when trained on and validated against historical data. Long short-term memory trained on inclination and rotational velocity, gave especially good predictions.
The simplest network available, a feed-forward network, was unable to make any predictions of note, struggling when presented with conditions different to what it was trained on – a classic case of overfitting.

These results have in common that they have been validated against data which has been preprocessed by a Kalman filter. It has significantly increased the quality of the results from the artificial neural networks, while at the same time producing results of note on its own. The correlation found between Tilt and inclination outperforms the method currently used, and can be universally applied to all operations with accelerometers. The angular accelerations of the system has been described in the form of Euler-Lagrange equations in our simplified mathematical model. These have been fed into a Kalman filter and showed a promising ability to predict bending moments. A purely mathematical model can potentially predict the fatigue of future operations, a highly sought after tool in the industry.

The relatively small amount of data with non-linear behaviour that was available for validation, somewhat limit the certainty of our results. Repeating the predictions seen when training on simulated data would be of interest, particularly in different environmental conditions. Furthermore, the OrcaFlex model have limitations in both the modelling itself, and the forces applied to it. Future projects would benefit from modelling the system with even greater accuracy.

Due to their performance in predicting signals of different frequencies, the combination of convolutional and recurrent neural networks, is an avenue that should be of interest for future projects. Additionally, an artificial neural network using Hamilton's canonical equations of motion could yield interesting results.

Increasing the lifespan of wellheads will continue to be a top priority in the industry. As a result, reliable and accurate fatigue calculations will be high in demand. The indirect method proposed in this report may serve as a valuable asset for future operations. Building on the foundation laid by this project, a clear recommendation is to continue exploring convolutional neural networks trained on simulations.

Chapter 10

Individual technical contribution

10.1 Sverre Weum Berge

Table 10.1: Technical overview for Sverre Weum Berge

What	When	Where
Physical model	28.02 - 25.05	Memory stick
O2-system tutorial summary	18.02 - 03.03	Memory stick
Setup, running and extraction from OrcaFlex simulations	29.04 - 08.05	5.1.3
Small manual, for choosing simulations	08.05 - 09.05	Memory stick
Spectral density analysis	29.04 - 05.05	8.9
Auxiliary figures, visual explanations	Whole period	Memory stick
OrcaFlex/O2-system design		
Implementing sensors in OrcaFlex	25.02 - 03.03	4.2.2
Changing the COB sensors into more accurate positions	27.04 - 03.05	4.2.2.1
Changes to be able to gather sensor data	04.03 - 11.03	4.2.1
Implementing an irregular sea state	04.03 - 11.03	4.2.6
Transferring all changes from OrcaFlex to the O2-system	11.03 - 05.04	Memory stick
Quality checking calculations for preservation of the original model, and implementation from OrcaFlex to O2-system.	05.04 - 18.04	5.1.1
Updating our model to be compatible with OrcaFlex 11.2	18.04 - 29.04	Memory stick
Changes to sensors due to axis calibration	29.04 - 05.05	4.2.2.1
New solution to gathering SMU data	29.04 - 05.05	4.2.2.2
Cardinal directions, OrcaFlex and historical relation	29.04 - 05.05	4.2.5
Implementing historical mean wave directions	29.04 - 05.05	4.2.5
Design for easy setup OrcaFlex simulations	20.04 - 29.04	5.1.3
SMU changes to orientation and position	06.05 - 12.05	4.2.3

Table 10.2: Total hours for Sverre Weum Berge

Date	Fri	Sat	Sun	Mon	Tue	Wed	Thur	Total
31.12 - 06.01					2	3,5	2	7,5
07.01 - 13.01	2			2,5		9	8	21,5
14.01 - 20.01	4			5,5		6	10	25,5
21.01 - 27.01	7			7		9	10	33
28.01 - 03.02	3		4,5	9,5	5	6		28
04.02 - 10.02				7,5	3	8,5	9	28
11.02 - 17.02	6			9,5	1	9	8	33,5
18.02 - 24.02				6		7,5	8	21,5
25.02 - 03.03	7,5			5			11,5	24
04.03 - 10.03	7,5			11	2,5	3,5	7	31,5
11.03 - 17.03	8			11,5		4	1	24,5
18.03 - 24.03							7,5	7,5
25.03 - 31.03	1			14	13	7	10	45
01.04 - 07.04	10,5	2	8,5	11,5		8	7	47,5
15.04 - 21.04				1	7	7,5	7,5	23
22.04 - 28.04	5			8	9	4,5	8,5	35
29.04 - 05.05	12			11,5	8	11	7,5	50
06.05 - 12.05	7,5			8,5	12	12,5	7,5	48
13.05 - 19.05	8	2	6	14,5	6,5	13	13	63
20.05 - 26.05	11	14	12	3				40
Total	100	18	31	147	69	129,5	143	637,5

10.2 Runar Bergum Eckholdt

Table 10.3: Technical overview for Runar Bergum Eckholdt

What	When	Where
Architectural planning	04.02-24.02	4.3 Memory stick
Classifier Designed Coded parts of it Integrated sub-components	 11.02-24.02 05.04-28.04	 4.3.1.1 Memory stick
Recurrent Neural Networks Designed & test Preparing data Analysing results Coding	 March-May	 4.4.2 6.3.2 Memory stick
ANN Manager Assisted with design Programmed some functions	 11.03-24.03 29.04-05.05	 4.3.2 Memory stick
Script to merge simulation files (MergeFiles.py)	19.04-05.05	Memory stick
DataPreparationTool Designed Integration of the sub-components	 04.02-03.03	 4.3.1 Memory stick
Orcaflex data analysis Correlation analysis Frequency adjustments	 May	 Memory stick
Kalman filter Assisted the electrical engineer to put the theory to code	May	4.5.3
Webpage Second iteration Changed it to php	 04.03-10.03	 Memory stick

Table 10.4: Total hours for Runar Bergum Eckholdt

Date	Fri	Sat	Sun	Mon	Tue	Wed	Thur	Total
31.12 - 06.01						1,5	0,5	2
07.01 - 13.01	1,5			4,5		6	5,5	17,5
14.01 - 20.01	2,5		3	8	3	8	9	33,5
21.01 - 27.01	7			8		8	10,5	33,5
28.01 - 03.02	6	4,5	7	10	4	3		34,5
04.02 - 10.02				6	1	9	8	24
11.02 - 17.02	8		4	9,5	1	7,5	10	40
18.02 - 24.02	7			9	1,5	11	8	36,5
25.02 - 03.03	5		3	9,5			7,5	25
04.03 - 10.03	5			6,5	5,5	8	7,5	32,5
11.03 - 17.03	7		5	7,5	6,5	9	9,5	44,5
18.03 - 24.03	7,5			2	2		8,5	20
25.03 - 31.03	3	4	5	8,5	11,5	13	8,5	53,5
01.04 - 07.04	7,5	5	5	8		12	12	49,5
08.04 - 14.04	7,5	2	5	10	7	3		34,5
15.04 - 21.04				9	7	8	8	32
22.04 - 28.04	8			7,5	10	9	11	45,5
29.04 - 05.05	10	4		9	10	12,5	9,5	55
06.05 - 12.05	10	7	11,5	13	13	9	10	73,5
13.05 - 19.05	11	5	9	12	7,5	9,5	14,5	68,5
20.05 - 26.05	14,5	14	12	3				43,5
Total	128	45,5	69,5	160,5	90,5	147	158	799

10.3 Markus Øvereng Leander

Table 10.5: Technical overview for Markus Øvereng Leander

What	When	Where
DataReader Design Programmed	11.02 - 24.02	4.3.1 Memory stick
AnnManager Design Programmed core	25.02 - 24.03 19.04 - 05.05	4.3.2 Memory stick
FFNN Design Programmed Result analysis	19.04 - 12.05 14.05-16.05	4.4.1,6.3.1 Memory stick
Weather Script Programmed	29.04	Memory stick
Webpage Programmed	25.02 - 13.03	Memory stick
TimeConverter Script Programmed	03.04	Memory stick

Table 10.6: Total hours for Markus Øvereng Leander

Date	Fri	Sat	Sun	Mon	Tue	Wed	Thur	Total
31.12 - 06.01						1,5	3	4,5
07.01 - 13.01	2			3		5,5	5	15,5
14.01 - 20.01	5		0,5	7,5	2,5	5,5	9	30
21.01 - 27.01	6,5			9		10	6	31,5
28.01 - 03.02	5	5	2	8,5	6,5	4		31
04.02 - 10.02				7	1	7	7,5	22,5
11.02 - 17.02	7			11,5	1	5	8,5	33
18.02 - 24.02	5			7		7	7	26
25.02 - 03.03	4			7	3		7,5	21,5
04.03 - 10.03	9			9,5	6	6,5	7,5	38,5
11.03 - 17.03	6,5	1	3	10,5	3	5	8	37
18.03 - 24.03	6,5							6,5
25.03 - 31.03		2	3	14	14	12	7,5	52,5
01.04 - 07.04	5,5	8	7	9		5		34,5
08.04 - 14.04				5,5	4			9,5
15.04 - 21.04			1	4	8	8,5	8	29,5
22.04 - 28.04	6			9	8	7	8	38
29.04 - 05.05	8			7,5	9	11,5	8	44
06.05 - 12.05	8,5		3	13	14	10	9	57,5
13.05 - 19.05	6	7	4,5	12,5	7	10,5	12	59,5
20.05 - 26.05	13	14	12	3				42
Total	103,5	37	36	158	87	121,5	121,5	664,5

10.4 Sondre Tiller Løver

Table 10.7: Technical overview for Sondre Tiller Løver

What	When	Where
Orientation Orientation matrix for turning the measurements into global axis.	18.02 - 24.02	4.1.1
Testing Took measurements from the IMU sensors to make a measurement noise matrix for the Kalman filter.	25.02 - 03.03	4.1.2
Fusion filters Kalman Complementary Tilt Integration	18.02 - 09.05	4.5 4.5.1 4.5.2 4.5.4
Ordinary filters Made different filters for further analysis (Low-pass, High-pass).	04.03 - 10.03	Presentation 2 documentation Memory stick
Statistical tests Tested their statistical values with F-test. Tested their similarity using the MRMR test.	07.04 - 13.04	5.2.1 5.2.2
State space Made a state space model for the Lagrangian motion model	14.04 - 20.04	4.6.6 8.1
Mathematical model Started the work of a mathematical model for the system. Found the highest yielding fusion algorithm for estimating the angle on the stack-up. Created new inclination measurements based on Lagrangian motion model. Created a better way of finding the inclinations with the contribution of a damper.	11.03 - 11.05	6.1 6.2.1 Presentation 2 documentation Memory stick
OrcaFlex Made script for comparing the spectral density in OrcaFlex and historical data	28.04 - 04.05	8.9 Memory stick

Table 10.8: Total hours for Sondre Tiller Løver

Date	Fri	Sat	Sun	Mon	Tue	Wed	Thur	Total
07.01 - 13.01	1,5			3,5		6,5	6,5	18
14.01 - 20.01	1,5		5	4	5		3	18,5
21.01 - 27.01	7			4,5		6	7	24,5
28.01 - 03.02	4,5	2,5	5	10,5	8,5	6	1	38
04.02 - 10.02	2		1	6,25	3,25	7,5	7,5	27,5
11.02 - 17.02		4		7		7	9,5	27,5
18.02 - 24.02	5			6,5		8	8	27,5
25.02 - 03.03	4		9	11		6	6	36
04.03 - 10.03	6		3	13		9,5	12	43,5
11.03 - 17.03	5	8	6	11		8	4	42
25.03 - 31.03			8	14	13,5	8	7,5	51
01.04 - 07.04	5,5	7	7	10,5	6	8,5	10	54,5
08.04 - 14.04	4			1,5	2	1,5		9
15.04 - 21.04					7	6	7	20
22.04 - 28.04	5			4,5	9	6	9	33,5
29.04 - 05.05	7			11,5	9	9	10,5	47
06.05 - 12.05	5	4	7,5	15	12	14	13	70,5
13.05 - 19.05	11	7	12	10	5	14,5	11	70,5
20.05 - 26.05	13	14	12	3				37
Total	87	46,5	75,5	147,25	80,25	132	132,5	701

10.5 Jørgen Winther Søbstad

Table 10.9: Technical overview for Jørgen Winther Søbstad

What	When	Where
Plotter Design Code	11.02 - 17.02	Memory stick
Data labeler Design Code	22.02 - 28.02	4.3.1.1 Memory stick
Dataset generator Design Code	03.03 - 22.04	4.3.1.3 Memory stick
Webpage Work on css structure	11.03 - 12.03	Memory stick
Convolutional Neural Networks Design & test Generating datasets Misc scripting Analyze results Code	25.04 - 09.05	4.4.3 4.4.3.1 6.3.3 Memory stick
Damage assessment Code Experiments regarding detrended data	27.04 - 28.04	4.3.1.4 5.4 Memory stick
ANN Manager Added plotter functionality Network info saver	02.05 - 03.05	4.3.2 Memory stick

Table 10.10: Total hours for Jørgen Winther Søbstad

Date	Fri	Sat	Sun	Mon	Tue	Wed	Thur	Total
31.12 - 06.01						1,5	4	5,5
07.01 - 13.01	3			5,75		6	7	21,75
14.01 - 20.01	5			8,5	3	8	8	32,5
21.01 - 27.01	11	1,5		10		10,5	10	43
28.01 - 03.02	6,5	4	7,5	11	12	4		45
04.02 - 10.02	3			6,75	1	7,5	10,5	28,75
11.02 - 17.02	6	6		7,75	1	6	9	35,75
18.02 - 24.02	5			9	6	8,5	10	38,5
25.02 - 03.03	4			8			8	20
04.03 - 10.03	6			7	4	10	10	37
11.03 - 17.03	9			9	6,5	10,5	10	45
18.03 - 24.03	9			2	4			15
25.03 - 31.03		4	4	5	12	14	8,5	47,5
01.04 - 07.04	7,5	7,5	5	7,25		12	10	49,25
08.04 - 14.04	8			9	3	5	5	30
15.04 - 21.04	5				8	9	7	29
22.04 - 28.04	6,5			9,25	10	8	11,5	45,25
29.04 - 05.05			2	11,5	10	13	11	47,5
06.05 - 12.05	10	7	10	14	13,5	7	10	71,5
13.05 - 19.05	6	7	6	8	6	9	15,25	57,25
20.05 - 26.05	13	14	12	3				42
Total	123,5	51	46,5	151,75	100	149,5	164,75	787

10.6 Martin Sørensen

Table 10.11: Technical overview for Martin Sørensen

What	When	Where
Mathematical model Theoretical research Defining the system Calculations Test environment Running simulations Evaluating simulations Formulas for Kalman-filter	04.04 - 14.05 (125 hours)	4.6 4.6.2 4.6.3 4.6.4 4.6.5 4.6.6.1 5.3 5.3.1 5.3.2 5.3.3 8.1 8.2 8.3 Appendix D
OrcaFlex-design Created method for the initial sensor placement in OrcaFlex Create both full and down-scaled simplified model of the stack-up Created method for extracting all sensor parameters for OrcaFlex Calibrated the sensors by testing and reviewing methods Researched and defined statistical weather data parameters	07.02 - 01.04 (90 hours)	2.5.3 2.6.2 4.2.2 4.2.4.1 4.2.4.2 4.2.4.3 5.1.4 Appendix B.1
OrcaFlex-simulations Creating and running simulations for ML-training	18.04 - 12.05 (20 hours)	B.3 Memory stick
Python-scripts Converting OrcaFlex simulations to CSV-format Mathematical operations on dataframes with sensor parameters Plotting historical data against simulate data Solving equations with Scipy Ordinary differential equation program	03.03 - 12.05 (35 hours)	4.2.4.4 5.3.3 Memory stick
Fatigue calculation Calculating method for transforming bending moment	05.05 - 06.05 (5 hours)	Memory stick
Theoretical research Reading relevant research papers and forwarding relevant papers to other group members	13.01 - 25.02 (25 hours)	Memory stick
Presentation for TFMC Holding a 10 minute technical presentation on behalf of the group for TFMC department	13.02 - 15.02 (10 hours)	Memory stick

Table 10.12: Total hours for Martin Sørensen

Date	Fri	Sat	Sun	Mon	Tue	Wed	Thur	Total
31.12 - 06.01					6	4	5,5	15,5
07.01 - 13.01	2,5			7	1	8,5	11,5	30,5
14.01 - 20.01	1,5			9	1	5	12	28,5
21.01 - 27.01	10		2	9		2,5	10	33,5
28.01 - 03.02	11		5,5	10,5	8,5	4	1	40,5
04.02 - 10.02	4			9	5	7,5	10,5	36
11.02 - 17.02	8,5		3,5	9,5	7,5	2,5	9	40,5
18.02 - 24.02	9,5	7,5		7,5	1,5	6,5	8	40,5
25.02 - 03.03	8			7			15	30
04.03 - 10.03	8,5	4,5	4	8	10,5	6,5	10	52
11.03 - 17.03	7,5			6,5	10	9,5	5	38,5
18.03 - 24.03				2			5	7
25.03 - 31.03	4			9	13	9	7,5	42,5
01.04 - 07.04	7,5	8	9,5	10	3	8	9	55
08.04 - 14.04	4,5			0,5				5
15.04 - 21.04				2	8	8,5	8	26,5
22.04 - 28.04	5			11	7	9	12	44
29.04 - 05.05	7	2	4	11	4	9,5	13	50,5
06.05 - 12.05	8	9	6	10	10,5	11,5	14	69
13.05 - 19.05	6	2,5	6,5	9	12	9,5	13	58,5
20.05 - 26.05	11	14	12	3				40
Total	124	47,5	53	150,5	108,5	121,5	179	784

References

[1] TechnipFMC. Method document for operations with wams 2.0 rev a. 2021.

[2] Petter Russell, Stuart Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, 4 edition, 2021.

[3] S.O.H. Madgwick, R. Vaidyanathan, and A.J.L. Harrison. An Efficient Orientation Filter for Inertial Measurement Units (IMUs) and Magnetic Angular Rate and Gravity (MARG) Sensor Arrays. April 2010.

[4] TechnipFMC. Wellhead fatigue status report for askeladd j-1 (rpt60173609 rev b). 2021.

[5] Wamsconfigurator. http://kbg1demo01/WamsConfigurator/. Accessed: 10.03.2022.

[6] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda. https://developer.nvidia.com/cuda-zone, 2020.

[7] Guttorm Grytøyr, Fredy Coral, Halvor Borgen Lindstad, and Massimilliano Russo. Wellhead fatigue damage based on indirect measurements. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 56529, page V05BT04A015. American Society of Mechanical Engineers, 2015.

[8] Srikonda Rohit, Russo Massimiliano, Haakonsen Rune, and Periyasamy Peri. Real-time wellhead bending moment measurement using motion reference unit (mru) sensors and machine learning. In *Conference on Ocean, Offshore and Arctic Engineering*, volume OMAE2018, 2018.

[9] Halvor Snersrud Gustad. Bruk av kunstige nevrale nettverk til å predikere bøyemomenter til stigerør. Master's thesis, NTNU, 2019.

[10] Kenneth S Rubin. *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.

[11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[12] The pandas development team. pandas-dev/pandas: Pandas, feb 2020.

[13] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[14] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine

learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdfPyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[16] S. Bennett, J Skelton, and K Lunn. *Schaum's outlines of UML*. McGraw-Hill, New York NY, 2 edition, 2004.

[17] Statistical description of wave parameters. http://www.coastalwiki.org/wiki/Statistical_description_of_wave_parameters. Accessed: 04.03.2022.

[18] Yong Bai and Wei-Liang Jin. Marine structural design (second edition). chapter Chapter 5 - Wave Loads for Ship Design and Classification, pages 73–93. Butterworth-Heinemann, Oxford, second edition edition, 2016.

[19] ECI Technologies. COORDINATE SYSTEMS AND THE CONSEQUENCES THAT MISTAKES CAN HAVE ON PROJECTS. page 1, February 2022. (Accessed: 22.05.2022).

[20] Robert Grover Brown and Patrick Y C Hwang. *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions; 3rd ed.* Wiley, New York, NY, 1997.

[21] Leonhard Euler. Novi Commentarii academiae scientiarum Petropolitanae. volume 20, pages 189–207, 1776.

[22] Yan-Bin Jia. Quaternions and Rotations *. September 2013. Accessed: 04.04.2022.

[23] Jonathan Strickland. "What is a gimbal – and what does it have to do with NASA?". 2008. Accessed: 29.03.2022.

[24] TechnipFMC. Method document for fatigue calculation. 2019.

[25] National Aeronautics and Space Administatration. Earth fact sheet. 2022. Accessed: 29.04.2022.

[26] Bosch. Smart sensor combining accelerometer, gyroscope, magnetometer and orientation software. 2022. Accessed: 12.04.2022.

[27] TechnipFMC. Report, qualification test, subsea - completion and workover, well access management system qualification summary, hardware (rpt60170964 rev a). 2021.

[28] Cybernetics. Dws - deepwater strain sensor (dws02a-000pqs001). 2020.

[29] Orcina. Orcaflex viv toolbox validation. pages 13–15, Nov 2018.

[30] Total position and motion results. https://www.orcina.com/webhelp/OrcaFlex/Content/html/Vesselresults.htm#TotalPositionResults. Accessed: 14.03.2022.

[31] Environment: Data for torsethaugen spectrum. http://www.orcina.com/webhelp/OrcaFlex/Content/html/Environment,DataforTorsethaugenspectrum.htm. Accessed: 12.05.2022.

[32] Knut Torsethaugen, Sverre K. Haver, and Stavanger Norway. Simplified double peak spectral model for ocean waves. 2004.

[33] Environment: Wave data. https://www.orcina.com/webhelp/OrcaFlex/Content/html/Environment,Wavedata.htm. Accessed: 28.03.2022.

[34] Environment: Data for random waves. https://www.orcina.com/webhelp/OrcaFlex/Content/html/Environment,Dataforrandomwaves.htm#WaveSpectrumDiscretisation. Accessed: 30.03.2022.

[35] Robert C Martin. *Clean Architecture*. Pearson, 2018.

[36] Charles Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th international conference on Machine Learning (ICML-03)*, pages 147–153, 2003.

[37] V. Alto. Understanding the OLS method for Simple Linear Regression. 2019.

[38] Yi Yao, Timothy Sullivan IV, Feng Yan, Jiaqi Gong, and Lin Li. Balancing data for generalizable machine learning to predict glass-forming ability of ternary alloys. *Scripta Materialia*, 209:114366, 2022.

[39] Haidong Li, Jiongcheng Li, Xiaoming Guan, Binghao Liang, Yuting Lai, and Xinglong Luo. Research on overfitting of deep learning. In *2019 15th International Conference on Computational Intelligence and Security (CIS)*, pages 78–81, 2019.

[40] Guido van Rossum and Talin. Introducing abstract base classes. https://www.python.org/dev/peps/pep-3119/, April 2007.

[41] Jukka Lehtosalo ukasz Langa, Ivan Levkivskyi. Protocols: Structural subtyping. https://peps.python.org/pep-0544/, March 2017.

[42] Sean Robertson. NLP FROM SCRATCH: CLASSIFYING NAMES WITH A CHARACTER-LEVEL RNN. 2022. (Accessed: 19.04.2022).

[43] Xin Liu, Jun Zhou, and Huimin Qian. Comparison and evaluation of activation functions in term of gradient instability in deep neural networks. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 3966–3971, 2019.

[44] Serkan Kiranyaz, Turker Ince, Osama Abdeljaber, Onur Avci, and Moncef Gabbouj. 1-d convolutional neural networks for signal processing applications. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8360–8364, 2019.

[45] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. volume 105, pages 2295–2329, 2017.

[46] Osama Abdeljaber, Onur Avci, Mustafa Serkan Kiranyaz, Boualem Boashash, Henry Sodano, and Daniel J Inman. 1-d cnns for structural damage detection: Verification on a structural health monitoring benchmark data. *Neurocomputing*, 275:1308–1317, 2018.

[47] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj. Real-time patient-specific ecg classification by 1-d convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, 63(3):664–675, 2015.

[48] Wei Zhang, Chuanhao Li, Gaoliang Peng, Yuanhang Chen, and Zhujun Zhang. A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load. *Mechanical Systems and Signal Processing*, 100:439–453, 2018.

[49] Salim Malek, Farid Melgani, and Yakoub Bazi. One-dimensional convolutional neural networks for spectroscopic signal regression. *Journal of Chemometrics*, 32(5):e2977, 2018.

[50] S. Ekti Radin Charel, Eko Henfri Binugroho, M. Anfa'ur Rosyidi, R. Sanggar Dewanto, and Dadet Pramadihanto. Kalman filter for angle estimation using dual inertial measurement units on unicycle robot. In *2016 International Electronics Symposium (IES)*, pages 256–261, 2016.

[51] Kalman R. A new approach to linear filtering and prediction problems. *ASME- Journal of Basic Engineering*, 28:35–45, 1960.

[52] Gene F. Franklin, David J. Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall PTR, USA, 8th edition, 2020.

[53] Simone A. Ludwig and Kaleb D. Burnham. Comparison of euler estimate using extended kalman filter, madgwick and mahony on quadcopter flight data. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1236–1241, 2018.

[54] Garcia Mario. AHRS: Attitude and Heading Reference Systems. 2022. Accessed: 16.05.2022.

[55] B. E. Bona and Robert J. Smay. Optimum reset of ship's inertial navigation system. *IEEE Transactions on Aerospace and Electronic Systems*, AES-2(4):409–414, 1966.

[56] M. Spiegel. *Theoretical Mechanics*. McGraw-Hill Book Company, 1967.

[57] Lindstrøm T. and K. Hveberg. *Flervariabel analyse med lineær algebra*, volume 2. Gyldendal, 2019.

[58] Jason Yosinski Sam Greydanus, Misko Dzamba. Hamiltonian neural networks. Draft.

[59] P Gutierrez. Classical mechanics the hamiltonian. Accessed: 22.04.2022, 2003.

[60] Eric Poisson. Advanced mechanics. 2008.

[61] F. von Herrath and S. Mandell. The double pendulum problem. 2000.

[62] Riki. COPTER ANGLE CONTROL (ABSOLUTE). 2022. Accessed: 16.05.2022.

[63] M. Radovic, M. Ghalwash, and N. Filipovic. Minimum redundancy maximum relevance feature selection approach for temporal gene expression data. Number 6. BMC Bioinformatics 18, 2017.

[64] MATLAB. *version 9.12.0.1884302 (R2022a)*. The MathWorks Inc., Natick, Massachusetts, 2022.

[65] Donald F. Young. *Introduction to Fluid Mechanics*. Wiley, 2012.

[66] TechnipFMC. Interface data sheet (ids60172958 rev a). 2020.

[67] T. Bartlett Quimby. Continuous Beam Analysis. Accessed: 26.04.2022, 2014.

[68] Manon Kok, Jeroen D. Hol, and Thomas B. Schön. Using inertial sensors for position and orientation estimation. volume 11, pages 1–153, 2017.

[69] Kazusuke Maenaka. Mems inertial sensors and their applications. pages 71–73. 2008 5th International Conference on Networked Sensing Systems, 2008.

[70] California State University. Lagrange's equation. Accessed: 29.04.2022, Unpublished.

Appendix

The appendix is sectioned into five parts, sorted after importance. It is not necessary to read the appendix as it merely acts as a supplement to the report.

A Requirements and risk analysis

A.1 Project requirements

R-ID	Requirement description	Importance	Status	Source
RQ-PJ-1	The product will use an Artificial Neural Network (ANN) as an indirect method to find the bending moment on a subsea wellhead.	**HIGH**	**DONE**	**TFMC**
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-PJ-1-C	The accumulated fatigue calculated from predicted data should follow the trend and rough magnitude of the historical fatigue.	**must**	**DONE**	**09.05.2022**
T-PJ-1-C	The model should be able to predict the bending moment when the RFJ is enabeled.	**must**	**DONE**	**09.05.2022**
T-PJ-1-B	The prediction is done without the COB2 sensor.	**must**	**DONE**	**05.05.2022**
T-PJ-1-B	The prediction is done without any of the DWS sensors.	**optional**	**DONE**	**05.05.2022**
T-PJ-1-A	RQ-CE-1 must be approved.	**must**	**DONE**	**29.04.2022**

Figure A.1: Project Requirement 1

Testing Compare the predicted data generated from the ANN with both historical and simulated data.				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-PJ-1-A	See RQ-CE-1	**Success**	-	29.04.2022
T-PJ-1-B	RNN successfully trained on historical data. Predictions done using 1 day of historical, unseen data. The calculated fatigue are acceptable compared to the historical fatigue during the same time period, but falls short on other days.	**Success**	Runar	05.05.2022
T-PJ-1-C	CNN successfully trained on simulated data. Predictions done using 6 days of historical data with and without RFJ enabled. The calculated fatigue are acceptable compared to the historical fatigue during the same time period.	**Success**	Jørgen	09.05.2022

Figure A.2: Tests Project Requirement 1

A.2 Computer requirements

R-ID	Requirement description	Importance	Status	Source
RQ-CE-1	A Machine Learning environment with the focus on Artificial Neural Networks will be made.	HIGH	DONE	TFMC
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-CE-1-A	Neural Networks must be implemented using the PyTorch framework	must	DONE	29.04.2022
T-CE-1-B	The solution must be of modular design	must	DONE	29.04.2022
T-CE-1-A	The design must not limit the ability to use various algoritms	must	DONE	29.04.2022
T-CE-1-C	Implement custom functionality to compare and plot data in multiple ways	optional	DONE	03.05.2022
	The ANN should be ported from python to C++ for increased performance	optional	DISCARDED	-

Figure A.3: Computer Requirement 1

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-CE-1-A.1	Tested with RNN made using the PyTorch Framework.	Success	Runar	29.04.2022
T-CE-1-A.2	Tested with CNN made using the PyTorch Framework.	Success	Jørgen	29.04.2022
T-CE-1-B.1	Tested with RNN and CNN, it works as intended.	Success	Jørgen, Runar	29.04.2022
T-CE-1-C	Added funcitonallity is tested and validated to work as intended.	Success	Jørgen	03.05.2022
T-CE-1-A.3	Tested with FFNN made using the PyTorch Framework.	Success	Markus	10.05.2022
T-CE-1-B.2	Tested with FFNN, it works as intended.	Success	Markus	10.05.2022

Figure A.4: Tests Computer Requirement 1

R-ID	Requirement description	Importance	Status	Source
RQ-CE-2	The ANN will be trained by using historical data.	**MEDIUM**	**DONE**	**TFMC**
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-CE-2-A.1	The data must be passed through filters to remove noise.	**must**	**DONE**	**28.02.2022**
T-CE-2-B.2	The overall output are without significant loss of accuracy compared to the sensors today.	**must**	**DONE**	**12.05.2022**

Figure A.5: Computer Requirement 2

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-CE-2-A	Loaded historical sensor data and ran it though the preprocesser using a butterworth filter.	**Success**	Runar	28.02.2022
T-CE-2-B.1	Trained RNN on filtered historical data. The recurrent cell was unable to learn.	**Failed**	Runar	03.05.2022
T-CE-2-B.2	Trained deeper RNN on filtered historical data. The network was able to learn. R squared scores between 0.8-0.95	**Success**	Runar	12.05.2022

Figure A.6: Tests Computer Requirement 2

R-ID	Requirement description	Importance	Status	Source
RQ-CE-3	The ANN should be trained by using generated data from a realistic simulation made in Orcaflex under several sea states.	HIGH	DONE	TFMC
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-CE-3-B	The overall output are without significant loss of accuracy compared to the sensors today.	must	DONE	09.05.2022
T-CE-3-B	The data compared with filtered historical data and the simulation of the same historical data are within an acceptable threshold.	must	DONE	09.05.2022

Figure A.7: Computer Requirement 3

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-CE-3-A.1	Trained RNN on OrcaFlex data and validated on six days of historical data. Performance was not sufficient	Failed	Runar	04.05.2022
T-CE-3-B	Trained CNN on OrcaFlex data. The model was validated on 6 days of historical data. The results seem to be within the acceptable threshold, but we are awaiting the final word about this from our stakeholders.	Success	Jørgen	09.05.2022
T-CE-3-A.2	Trained deep RNN on OrcaFlex data and validated on six days of historical data. Performance was not sufficient	Failed	Runar	12.05.2022
T-CE-3-A.3	Trained residual RNN on OrcaFlex data and validated on six days of historical data. Performance was not sufficient	Failed	Runar	13.05.2022
T-CE-3-D	Trained LSTM on OrcaFlex data and validated on six days of historical data. Performance was not sufficient	Failed	Runar	16.05.2022
T-CE-3-C	Trained FFNN on OrcaFlex data. The model was validated on 6 days of historical data. The results are outside the acceptable threshold.	Failed	Markus	13.05.2022

Figure A.8: Tests Computer Requirement 3

R-ID	Requirement description	Importance	Status	Source
RQ-CE-4	A custom finite element space should be implemented in NGsolve.	MEDIUM	DISCARDED	TFMC
Related test	**Criteria**	**Necessity**	**Status**	**Date**
	Equations of fourth order derivative must be supported	must	DISCARDED	-

Figure A.9: Computer Requirement 4

R-ID	Requirement description	Importance	Status	Source
RQ-CE-1.1	A Data Preparation Tool(DPT) component must be made.	HIGH	DONE	WIBE
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-CE-1.1-A	The DPT must include a Classifier.	must	DONE	28.02.2022
T-CE-1.1-B	The DPT must include a DataLoader.	must	DONE	28.02.2022
T-CE-1.1-C	The DPT must include a Preprocesser.	must	DONE	28.02.2022

Figure A.10: Computer Requirement 1.1

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-CE.1.1-A	The function classify was tested with historical data, code returned values as expected	Success	Runar	28.02.2022
T-CE.1.1-B	The dataLoader was tested and it loaded data as expected	Success	Runar	28.02.2022
T-CE.1.1-C	The function preprocess was tested with historical data, code returned values as expected	Success	Runar	28.02.2022

Figure A.11: Tests Computer Requirement 1.1

R-ID	Requirement description	Importance	Status	Source
RQ-CE-1.1.1	The Classifier must contain Signal Analysis and Data Labeling.	**HIGH**	**DONE**	**WIBE**
Related tests	**Criteria**	**Necessity**	**Status**	**Date**
T-CE-1.1.1-A	An Oscillation Analyser must be implemented.	**must**	**DONE**	**23.02.2022**
T-CE-1.1.1-B	A Signal Analyser must be implemented.	**must**	**DONE**	**23.02.2022**
T-CE-1.1.1-C	A clustering algorithm must be implemented.	**must**	**DONE**	**25.02.2022**

Figure A.12: Computer Requirement 1.1.1

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-CE.1.1.1-A	The code excecuted as expected	**Success**	Runar	23.02.2022
T-CE.1.1.1-B	The code excecuted as expected	**Success**	Runar	23.02.2022
T-CE-1.1.1-C	The component has been tested with historical data and works as intended.	**Success**	Jørgen	25.02.2022

Figure A.13: Tests Computer Requirement 1.1.1

R-ID	Requirement description	Importance	Status	Source
RQ-CE-1.1.2	The Data Reader must be able to set a file path and read csv files	HIGH	DONE	WIBE
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-CE-1.1.2-A	The data reader must be able to use relative path.	must	DONE	21.02.2022
T-CE-1.1.2-A	The data reader must be able to set a path	must	DONE	21.02.2022
T-CE-1.1.2-A	The data reader must be able to read csv files	must	DONE	24.02.2022
T-CE-1.1.2-B	The data reader must be able to read from multiple csv files	optional	DONE	03.03.2022
	The data reader must be able to set multiple paths	optional	DISCARDED	-
	The data reader should be able to read given time intervals	optional	DISCARDED	-
	The data reader should be able to read xlsx files	optional	DISCARDED	-
	The data reader should be able to read xlsb files	optional	DISCARDED	-

Figure A.14: Computer Requirement 1.1.2

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-CE-1.1.2-A	The final component was included in a test script where all functions was tested and verified.	Success	Markus	24.02.2022
T-CE-1.1.2-B	Loaded inn SMU1, SMU2, SMU3, COB1, and COB2 data	Success	Runar	03.03.2022

Figure A.15: Tests Computer Requirement 1.1.2

R-ID	Requirement description	Importance	Status	Source
RQ-CE-1.2	A Dataset Generator(DSG) must be made.	**HIGH**	**DONE**	**WIBE**
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-CE-1.2-A	The DSG must be able to balance data based on labels	**must**	**DONE**	**10.03.2022**
T-CE-1.2-A	The DSG must be able to balance data based on custom settings	**must**	**DONE**	**10.03.2022**
T-CE-1.2-B	The DSG must include a custom Dataset which inherits from PyTorch default Dataset class.	**must**	**DONE**	**14.03.2022**
T-CE-1.2-C	The DSG must be able to return the Balanced datasets as Tensors which can be directly used by the Pytorch DataLoader.	**must**	**DONE**	**21.03.2022**

Figure A.16: Computer Requirement 1.2

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-CE-1.2-A.1	A balancer which balance data based on custom settings and labels are implemented and tested.	**Success**	Jørgen	03.03.2022
T-CE-1.2-A.2	The balancer is further tested for CNN and RNN. Further work should be done for the RNN datasets as they shouln't have skips in time.	**Success**	Jørgen	07.03.2022
T-CE-1.2-A.3	New funcitonality to monitor balance is tested and validated.	**Success**	Jørgen	10.03.2022
T-CE-1.2-B	Custom Dataset + Sampler is tested and validated.	**Success**	Jørgen	14.03.2022
T-CE-1.2-C.1	Multiple tweaks and fixes are tested and validated.	**Success**	Jørgen	21.03.2022
T-CE-1.2-C.2	Re-wrote to store indecies instead of dataFrames. Tested and validated. Resulted in big efficiency-boost regarding computation time.	**Success**	Jørgen	21.04.2022

Figure A.17: Tests Computer Requirement 1.2

R-ID	Requirement description	Importance	Status	Source
RQ-CE-1.3	An Artificial Neural Network(ANN) Manager must be implemented.	HIGH	DONE	WIBE
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-CE-1.3-A	The ANN manager must be able to store presets	must	DONE	16.03.2022
T-CE-1.3-A	The ANN manager must be able to load presets	must	DONE	15.03.2022
T-CE-1.3-B	The ANN manager must be able to load models	must	DONE	27.03.2022
T-CE-1.3-B	The ANN manager must be able to save models	must	DONE	25.04.2022
T-CE.1.3-C	The ANN manager must be able to train models	must	DONE	29.04.2022
T-CE.1.3-C	The ANN manager must be able to predict with different models	must	DONE	29.04.2022

Figure A.18: Computer Requirement 1.3

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-CE-1.3-A	Ran the Ann manager with a FFNN and it was able to store filePath, columnNames, input and output number.	Success	Markus	15.03.2022
	Ran the Ann manager with a FFNN and it was able to load filePath, columnNames, input and output number.	Success	Markus	15.03.2022
T-CE-1.3-B	The ANN manager was able to store the model as a .pth file	Success	Markus	25.03.2022
	The ANN manager was able to load the already trained model	Success	Markus	27.03.2022
T-CE.1.3-C	Used the ANN Manager to train an RNN network	Success	Runar	29.04.2022
	Used the ANN Manager to load a model and predict validation data	Success	Runar	29.04.2022

Figure A.19: Tests Computer Requirement 1.3

A.3 Electrical requirements

R-ID	Requirement description	Importance	Status	Source
RQ-EE-1	A complementary filter and a Kalman filter should be made.	HIGH	DONE	TFMC
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-EE-1A	Complementary filter	must	DONE	12.04.2022
T-EE-1B	Kalman filter	must	DONE	29.04.2022
T-EE-1C	The output must remove noise from the SMU data such that the processed signal is reliable in training the ANN.	must	DONE	05.05.2022
T-EE-1D	Implementing and testing other fusion filters	Optional	DONE	05.05.2022

Figure A.20: Electrical Requirement 1

Testing The limits and accuracy of the currently used sensors must be studied, verified and documented.				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-EE-1A	Made with a Lowpass and highpass filter from T-EE-2A-B	Success	Sondre	12.04.2022
T-EE-1B	The Linear and Extended Kalman filter is fully built and tested on historical data	Success	Sondre	29.04.2022
T-EE-1C	The fusion filters automatically smoothes the signal, and therefore not required for the different kinds of fusion filters. Filters such as the Complementary and mahony uses filters for choosing the right data.	Success	Sondre	05.05.2022
T-EE-1D	Filters such as Madgwick, Mahony, Tilt and integration are also tested and compared with the bendingmoment.	Success	Sondre	05.05.2022

Figure A.21: Tests Electrical Requirement 1

R-ID	Requirement description	Importance	Status	Source
RQ-EE-2	Various filters will be implemented in the signal processing.	HIGH	DONE	TFMC
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-EE-2A	Lowpass	must	DONE	28.02.2022
T-EE-2B	Highpass	must	DONE	28.02.2022
T-EE-2C	Average mean	must	DONE	25.03.2022
T-EE-2D	Linear regression	must	DONE	10.03.2022
T-EE-2E	SFT and FFT	must	DONE	10.03.2022

Figure A.22: Electrical Requirement 2

Testing *Signal to Noise ratio and Error rates must be documented and compared with the Accurate measurements.*				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-EE-2A	Made both with scipy package and with kaiser window	Success	Sondre	28.02.2022
T-EE-2B	Made both with scipy package and with kaiser window	Success	Sondre	28.02.2022
T-EE-2C-1	Tested to find the statistical mean and variance of the signals	Success	Sondre	25.03.2022
T-EE-2D	Uniform and Kernel filter	Success	Sondre	10.03.2022
T-EE-2E	Spectogram visualization and with a kaiser window.	Success	Sondre	10.03.2022
T-EE-2C-2	Tested with Kalman filter for filter fusion. Gave less results	Success	Sondre	03.05.2022

Figure A.23: Tests Electrical Requirement 2

R-ID	Requirement description	Importance	Status	Source
RQ-EE-3	A magnetometer may be implemented to calculate the offset of the BOP.	LOW	DONE	WIBE
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-EE-3A	Investigate if torsion is a relevant problem	Optional	DONE	22.02.2022

Figure A.24: Electrical Requirement 3

Testing *A calibration matrix needs to be developed and tested in Python/C for accurate measurements, and to eliminate noise in the original signal.*				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-EE-3A	The need of a torsion based model is concluded not neccesery, but is recommended for tesating more fusion filters	Success	Sondre	22.02.2022

Figure A.25: Tests Electrical Requirement 3

A.4 Mechanical requirements

R-ID	Requirement description	Importance	Status	Source
RQ-ME-1	The OrcaFlex-model must be adapted to our purpose, both in the CAD-software and the O2-script.	HIGH	DONE	TFMC
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-ME-1-D	The subsea motions sensors must be placed at the exact distance from the Wellhead Datum as in the specific operation	must	DONE	28.02.2022
T-ME-1-D	The strain must be measured at the exact location of each of the 7 strain sensors.	must	DONE	25.02.2022
T-ME-1-B	The OrcaFlex-model must be meshed at a fine enough level at important sections to achieve accurate results.	must	DONE	07.03.2022
T-ME-1-C	Irregular waves parameters must be used	must	DONE	09.03.2022
T-ME-1-A	Enviromental input must not contstrain the simulation in ways that makes it differ greatly from the actual conditions	must	DONE	09.03.2022

Figure A.26: Mechanical Requirement 1

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-ME-1-A.1	Running dynamic simulation with wave data from 04.01.2020	Success	Martin	04.03.2022
T-ME-1-A.2	Running dynamic simulation with wave data from 14.01.2020	Success	Sverre	04.03.2022
T-ME-1-A.3	Running dynamic simulation with wave data from 09.01.2020	Success	Martin	05.03.2022
T-ME-1-A.4	Running dynamic simulation with wave data from 09.01.2020	Success	Martin	05.03.2022
T-ME-1-A.5	Running dynamic simulation with wave data from 04.01.2020	Success	Martin	05.03.2022
T-ME-1-A.6	Running dynamic simulation with wave data from 30.12.2019	Success	Martin	05.03.2022
T-ME-1-A.7	Running dynamic simulation with wave data from 17.01.2020	Success	Martin	05.03.2022
T-ME-1-A.8	Running dynamic simulation with wave data from 18.01.2020	Success	Martin	06.03.2022
T-ME-1-A.9	Running dynamic simulation with wave data from 15.01.2020	Success	Martin	06.03.2022
T-ME-1-A.10	Running dynamic simulation with wave data from 31.12.2019	Success	Martin	06.03.2022
T-ME-1-A.11	Running dynamic simulation with wave data from 12.01.2020	Success	Martin	06.03.2022
T-ME-1-B.1	Running dynamic simulation with wave data from 17.01.2020	Success	Sverre	07.03.2022
T-ME-1-B.2	Running dynamic simulation with wave data from 17.01.2020	Success	Sverre	07.03.2022
T-ME-1-B.3	Running dynamic simulation with wave data from 17.01.2020	Success	Sverre	07.03.2022
T-ME-1-C.1	Running dynamic simulation with wave data from 15.01.2020	Success	Martin	08.03.2022
T-ME-1-B.4	Running dynamic simulation with wave data from 17.01.2020	Success	Sverre	08.03.2022
T-ME-1-B.5	Running dynamic simulation with wave data from 17.01.2020	Success	Sverre	09.03.2022
T-ME-1-C.2	Running dynamic simulation with wave data from 15.01.2020	Success	Martin	09.03.2022
T-ME-1-D.1	Placing strain sensors on OrcaFlex model	Success	Martin/Sverre	25.02.2022
T-ME-1-D.2	Placing motion sensors on OrcaFlex-model	Success	Martin/Sverre	25.02.2022
T-ME-1-D.3	Placing motion sensors on OrcaFlex-model (only in z-direction)	Success	Martin/Sverre	28.02.2022

Figure A.27: Tests Mechanical Requirement 1

R-ID	Requirement description	Importance	Status	Source
RQ-ME-2	A Python-script must be made to import and compare historcal data from the Deep Water Strain and Subsea Motion Unit sensors, with the OrcaFlex results.	MEDIUM	DONE	WIBE
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-ME-2-A	Must be able to import both historical data and OrcaFlex data	must	DONE	09.03.2022
T-ME-2-C	Must be able to plot strain, rotational velocity and acceleration against each other	must	DONE	09.03.2022
T-ME-2-A	Must sort all sensors in to seperate dataframes, making it easy to compare them.	must	DONE	09.03.2022
T-ME-2-B	Create a function to generate mean values for all parameters	optional	DONE	09.03.2022
T-ME-2-B	Create a function to calculcate the standard deviation of parameters	optional	DONE	12.03.2022
T-ME-2-C	Must be able to plot historical data against OrcaFlex data for each parameter	must	DONE	09.03.2022

Figure A.28: Mechanical Requirement 2

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-ME-2.A.1	Import simulation data from our file structure to pandas dataframe	Success	Martin	08.03.2022
T-ME-2.C.1	Plot simulated data for each sensor	Success	Martin	09.03.2022
T-ME-2.A.2	Import historical data from TFMCs file structure to pandas dataframe	Success	Martin	09.03.2022
T-ME-2.C.2	Plot historical data against simulated data for all parameters	Success	Martin	09.03.2022
T-ME-2-B.1	Define functions for calculating mean values for historical and simulated data	Success	Martin	10.03.2022
T-ME-2-B.2	Define functions for calculating standard deviations for historical and simulated data	Success	Martin	12.03.2022

Figure A.29: Tests Mechanical Requirement 2

R-ID	Requirement description	Importance	Status	Source
RQ-ME-3	The completed OrcaFlex model must replicate the magnitude of the strain from the two COB sensors, and the acceleration and rotational velocity from the three SMU sensors compared to historical operations.	HIGH	DONE	TFMC
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-ME-3-A	The acceleration measurements extracted must be identical to that of the historical data	must	DONE	02.03.2022
T-ME-3-B	The rotational velocity measurements extracted must be identical to that of the historical data	must	DONE	16.03.2022
T-ME-3-C	The strain measurements extracted must be identical to that of the historical data	must	DONE	09.03.2022
T-ME-3-D	The historical wave data must be imported to the OrcaFlex simulation	must	DONE	25.02.2022
T-ME-3-E	Implement historical mean wave direction, need to adress the relationship between the OrcaFlex model and historical cardinal direction.	must	DONE	03.05.2022
T-ME-3-E	Find the relationship between the global and local axis in the OrcaFlex model and histrocial setup.	must	DONE	29.04.2022

Figure A.30: Mechanical Requirement 3

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-ME-3-A.1	Plotting acceleration with GX/GY Acceleration against historical data	Success	Martin	28.02.2022
T-ME-3-B.1	Plotting rotational velocity with GX/GY Velocity against historical data	Success	Martin	28.02.2022
T-ME-3-C.1	Plotting strain with ZZ strain against historical data	Success	Martin	28.02.2022
T-ME-3-B.2	Plotting rotational velocity from linear velocity against historical data	Success	Martin	07.03.2022
T-ME-3-A.2	Plotting acceleration with acceleration rel g. against historical data	Success	Martin	02.03.2022
T-ME-3-B.3	Plotting rotational velocity from dynamic angle against historical data	Success	Martin	16.03.2022
T-ME-3-C.2	Plotting strain from bending stress against historical data	Success	Martin	09.03.2022
T-ME-3-D	Simulation results from several independent simulations matches historical data.	Success	Martin	16.03.2022
T-ME-3-E.1	Principal wave direction changes to the correct orientation, based on OrcaFlex's cardinal system.	Success	Sverre	03.05.2022
T-ME-3-E.2	Comparing the 3D view of the OrcaFlex model with figures based on correct historical setup.	Success	Sverre	29.04.2022

Figure A.31: Tests Mechanical Requirement 3

R-ID	Requirement description	Importance	Status	Source
RQ-ME-4	A downscaled model based on the OrcaFlex-model should be designed to improve the efficiency and accuracy when training the ML-model-	**MEDIUM**	**DISCARDED**	**TFMC**

Figure A.32: Mechanical Requirement 4

R-ID	Requirement description	Importance	Status	Source
RQ-ME-5	A Python-script must be made to convert data exported from OrcaFlex to a similar format as historical data	MEDIUM	DONE	WIBE
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-ME-5-A	Must convert all single parameter excel-files extracted from OrcaFlex to a combined csv-file for each sensor	must	DONE	03.03.2022
T-ME-5-B	Must convert elapsed time in the extracted OrcaFlex-results to the specific datetime of the historical operation period thats being plotted against	must	DONE	08.03.2022
T-ME-5-A	Must convert all parameters to correct unit and of similar magnitude as the historical data	must	DONE	04.03.2022
T-ME-5-A	Should be written in a way that minimizes the input when converting new files	optional	DONE	06.03.2022
T-ME-5-D	Must be altered to convert input from results extracted from the O2-system	must	DONE	26.04.2022
T-ME-5-D	Should be improved to alter data more efficiently	optional	DONE	05.05.2022
T-ME-5-C	Must correctly convert output from OrcaFlex in units and magnitude	must	DONE	15.03.2022

Figure A.33: Mechanical Requirement 5

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-ME-5-A.1	Converting OrcaFlex .xlsx-files to csv	Success	Martin	03.03.2022
T-ME-5-A.2	Combining csv-files per sensor	Success	Martin	04.03.2022
T-ME-5-A.3	Converting to identical format as historical data	Success	Martin	04.03.2022
T-ME-5-A.4	Simplifying script so that it needs less input	Success	Martin	06.03.2022
T-ME-5-C.1	Recalculating linear velocity to angular velocity in conversion	Success	Martin	07.03.2022
T-ME-5-B	Converting elapsed time to datetime	Success	Martin	08.03.2022
T-ME-5-C.2	Recalculating dynamic angle to angular velocity in conversion	Success	Martin	09.03.2022
T-ME-5-C.3	Recalculating bending stress to strain in conversion	Success	Martin	15.03.2022
T-ME-5-D.1	Extracted and converted data successfully	Success	Sverre	26.04.2022
T-ME-5-D.2	Data extraction time drastically improved	Success	Sverre	05.05.2022

Figure A.34: Tests Mechanical Requirement 5

R-ID	Requirement description	Importance	Status	Source
RQ-ME-6	A Python-script that randomly gathers COB1-data from a given number of periods from the Askeladd J-1 historical data must be made to create input for the downscaled model	**HIGH**	**DISCARDED**	**WIBE**

Figure A.35: Mechanical Requirement 6

R-ID	Requirement description	Importance	Status	Source
RQ-ME-7	Make the Askeladd-J1 model compatible with the O2 system	HIGH	DONE	TFMC
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-ME-7-A	The lines Conductor, BOP and LFJ needs their top end stiffness values removed. The UFJ line needs the bottom end stiffness values removed.	must	DONE	16.03.2022
T-ME-7-B	The parameter (IncludeSeabedFrictionInStatics) with either values "No" and "Yes", needs to be replaced with the parameter (StaticSeabedFrictionPolicy) with the value "None". Applies to all lines that should not have any seabed friction.	must	DONE	16.03.2022
T-ME-7-C	Both the parameters (LayAzimuth) and (AsLaidTension) needs to be removed or commented out. Applies to all lines that should not have any seabed friction.	must	DONE	16.03.2022
T-ME-7-D	Remove the parameter ConnectionDamping from all used line components.	must	DONE	16.03.2022
T-ME-7-E	Implement irregular seastate, add "WaveDirectionSpreadingExponent" to Torsethaugen.yml., create variables for wave number og spectral direction and spreadexp.	must	DONE	17.03.2022
T-ME-7-F	Change the O2-system's LCG file to be able to change the wave direction on each spesific simulation.	must	DONE	04.05.2022

Figure A.36: Mechanical Requirement 7

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-ME-7-A	Run the O2 program, without error	Success	Sverre	16.03.2022
T-ME-7-B	Run the O2 program, without error	Success	Sverre	16.03.2022
T-ME-7-C	Run the O2 program, without error	Success	Sverre	16.03.2022
T-ME-7-D	Run the O2 program, without error	Success	Sverre	16.03.2022
T-ME-7-E	Run the O2 program, without error	Success	Sverre/Martin	17.03.2022
T-ME-7-F	Each simulation representing a time set in the historical weather data, has correct mean wave direction.	Success	Sverre	04.05.2022

Figure A.37: Tests Mechanical Requirement 7

R-ID	**Requirement description**	**Importance**	**Status**	**Source**
RQ-ME-8	The O2 setup needs to be effective and clean	**MEDIUM**	**DONE**	**WIBE**
Related test	**Criteria**	**Necessity**	**Status**	**Date**
T-ME-8-A	Setup a clean and effective RES.yml file, to produce good and quick result extractions.	**optional**	**Done**	**05.05.2022**
T-ME-8-B	All changes should be implemented into the correct "Overview" file.	**optional**	**Done**	**13.05.2022**
T-ME-8-C	Keep the structure in the GOV and LCG files simular to the base files provided.	**optional**	**Done**	**06.05.2022**

Figure A.38: Mechanical Requirement 8

Testing				
T-ID	**Test details**	**Result**	**Who**	**Date**
T-ME-8-A	Extracts correct data, and sorts them into a system a script easily can gather. Extra extractions is easy and quickly implemented.	Success	Sverre	05.05.2022
T-ME-8-B	Provides the reader with a complete overview of all changes and simulations.	Success	Sverre/Martin	13.05.2022
T-ME-8-C	Both files are just as easily navigated as the original. Added some comments to help new users.	Success	Sverre	06.05.2022

Figure A.39: Tests Mechanical Requirement 8

A.5 Risk analysis

Category	Description	Probability	Consequence	Risk
Mechanical	Unable to calibrate OrcaFlex-model	2	4	Medium
Electrical	Unable to implement Kalman-filter	2	3	Medium
Non-Technical	Damaged equipment (hardware)	3	3	Medium
Non-Technical	Prolonged absence of a group member	1	4	Low
Non-Technical	Misinterpreting data	4	3	Medium
Mechanical	Incorrect OrcaFlex-design variables	2	3	Medium
Electrical	Unable to find a correlation between motion data and strain	2	4	Medium
Mechanical	Unable to create a scaled-down OrcaFlex-model	3	2	Medium
Non-Technical	Unable to access vital systems / Software failure	3	3	Medium
Electrical	Unable to differentiate between signal and noise	3	3	Medium
Electrical	Unable to implement complimentary filter	1	3	Low
Mechanical	Sensor placement in OrcaFlex is wrong	2	3	Medium
Mechanical	Unable to gather correct data from sensors in OrcaFlex	1	2	Low
Mechanical	Unable to find and implement environmental data	1	3	Low
Electrical	Unable to process signals in Python and applying digital filters	2	3	Medium
Non-Technical	Stolen Computers	1	4	Low
Non-Technical	Major changes in requirements during the project	3	2	Medium
Computer	Undiscovered major bugs/errors	3	5	High
Computer	No results from the ML training	2	3	Medium
Computer	Slow code after implementation	1	5	Medium
Computer	Github Website shutdown	1	3	Low
Non-Technical	Not beeing able to reach the desired goal in the given time	4	2	Medium
Computer	Major flaw in software architecture	1	4	Low
Non-Technical	Software versions incompability	2	2	Low
Computer	Undiscovered minor bugs and error	4	1	Low
Non-Technical	Unsatisfactory support from TFMC advisors	1	3	Low

Figure A.40: Risk analysis

B OrcaFlex-model

B.1 Sensor calibration plots

B.1.1 COB-sensors

COB1-sensor

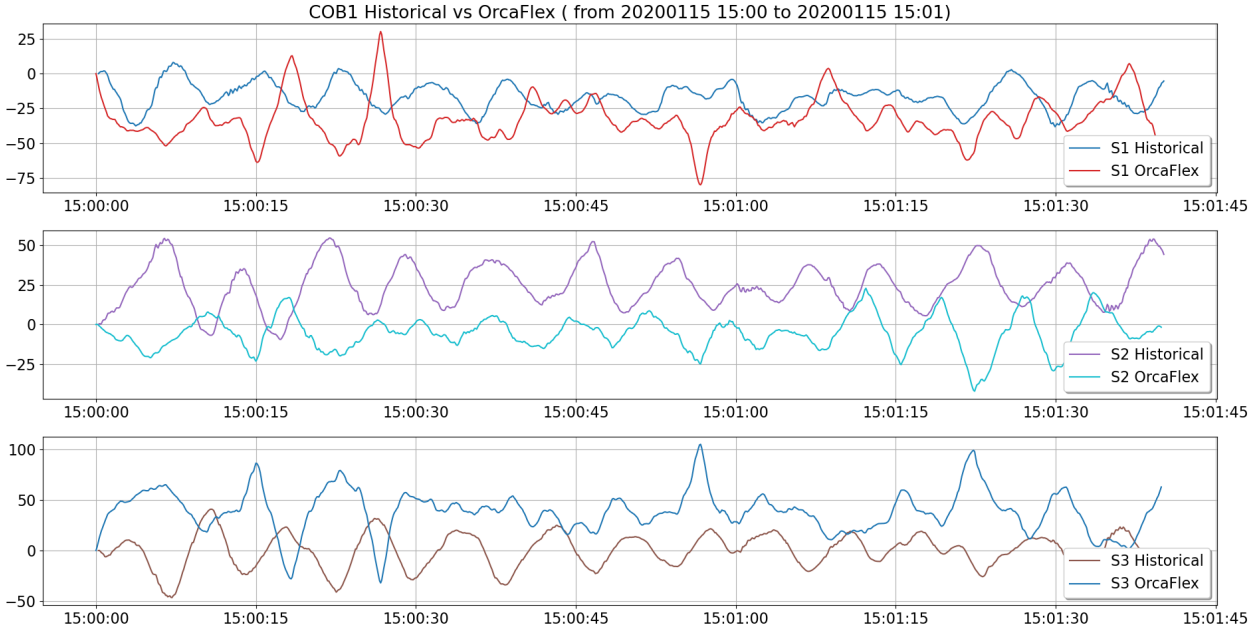


Figure B.1: COB1 Historical vs OrcaFlex in microstrain ($\mu m/m$)

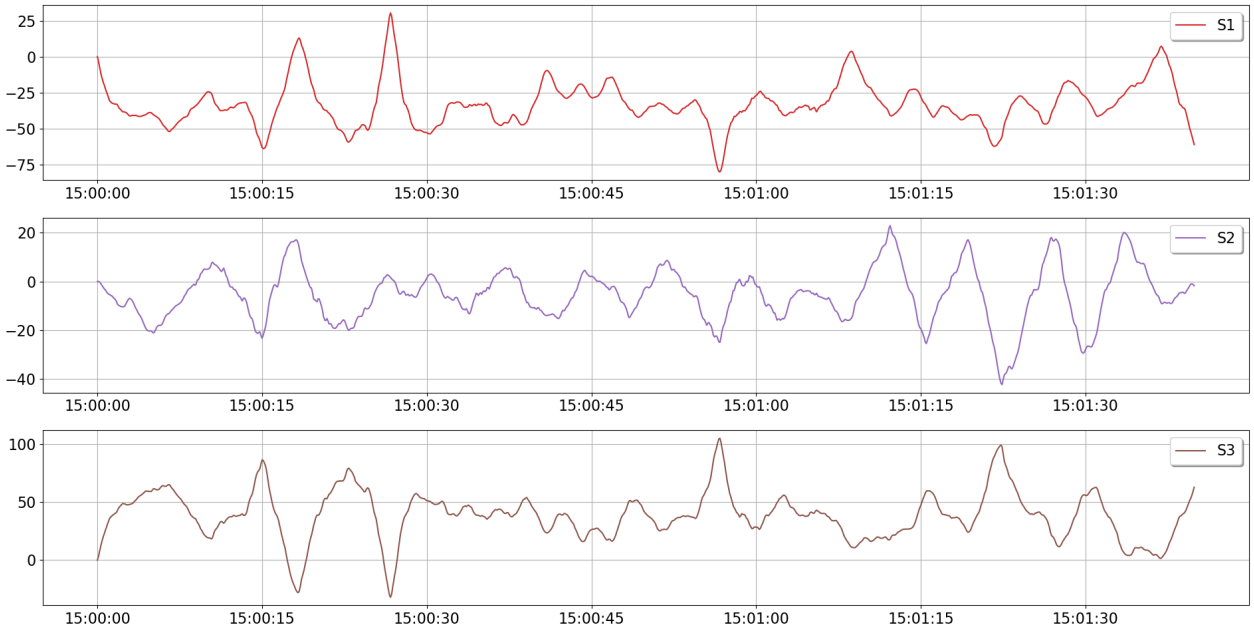


Figure B.2: COB1 OrcaFlex in microstrain ($\mu m/m$)

COB2-sensor

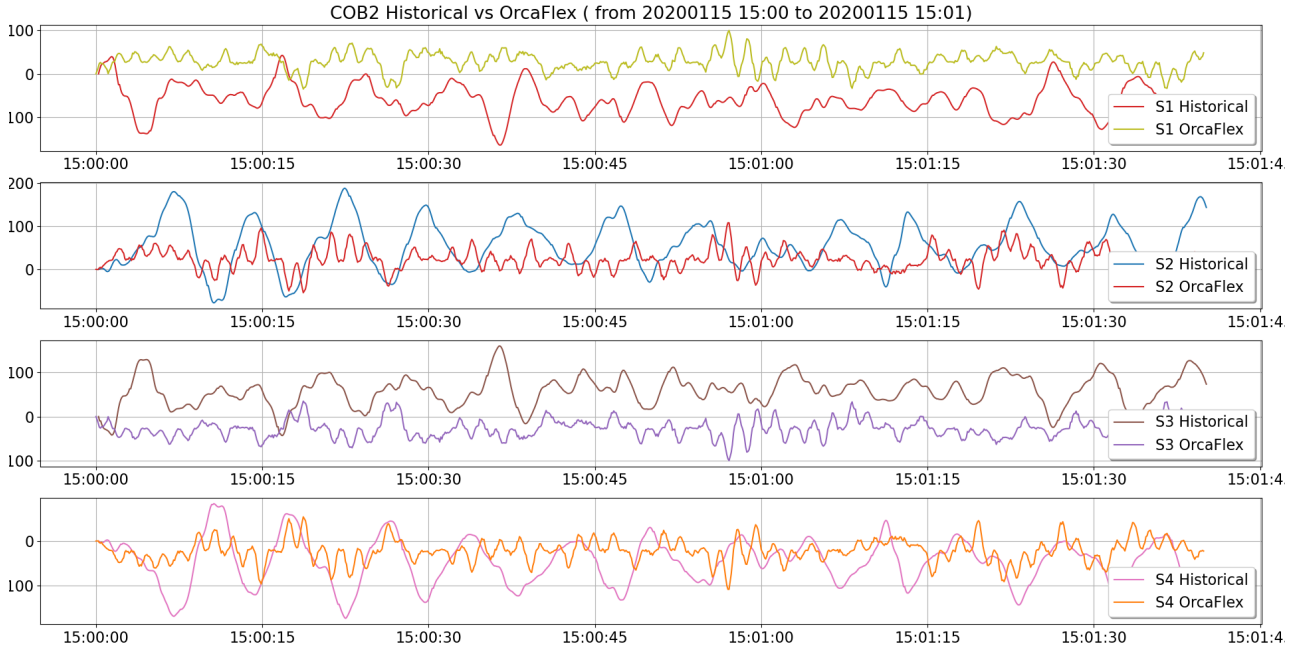


Figure B.3: COB2 Strain Historical vs OrcaFlex ($\mu m/m$)

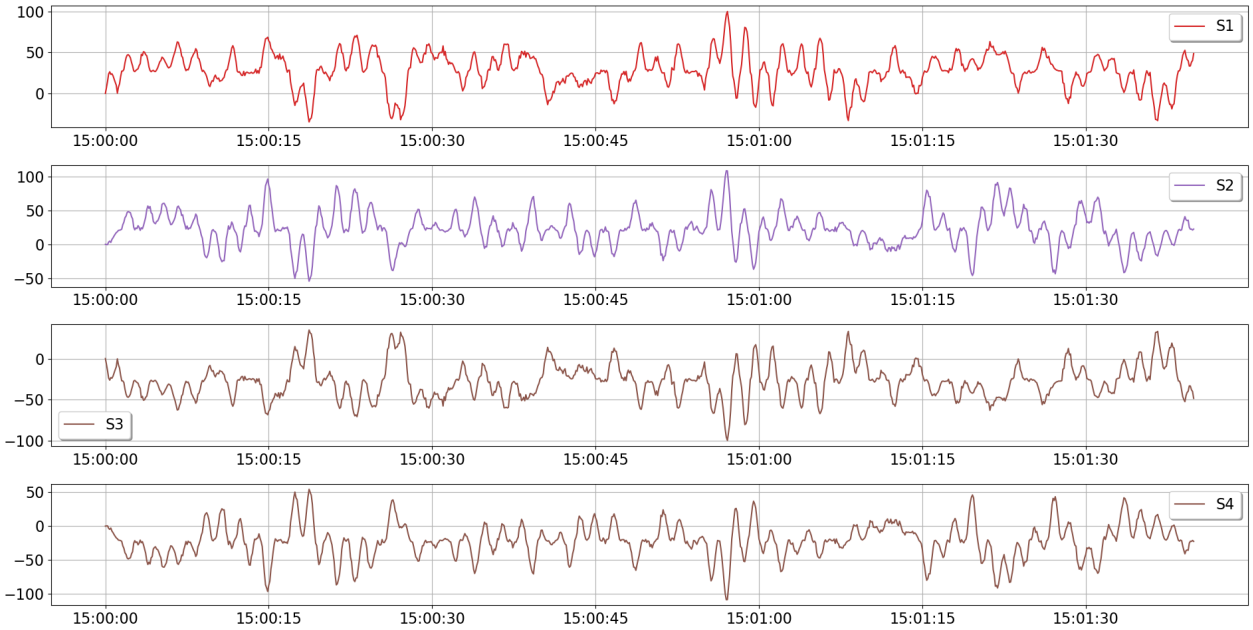


Figure B.4: COB1 OrcaFlex strain ($\mu m/m$)

B.1.2 SMU-sensors

SMU1-sensor

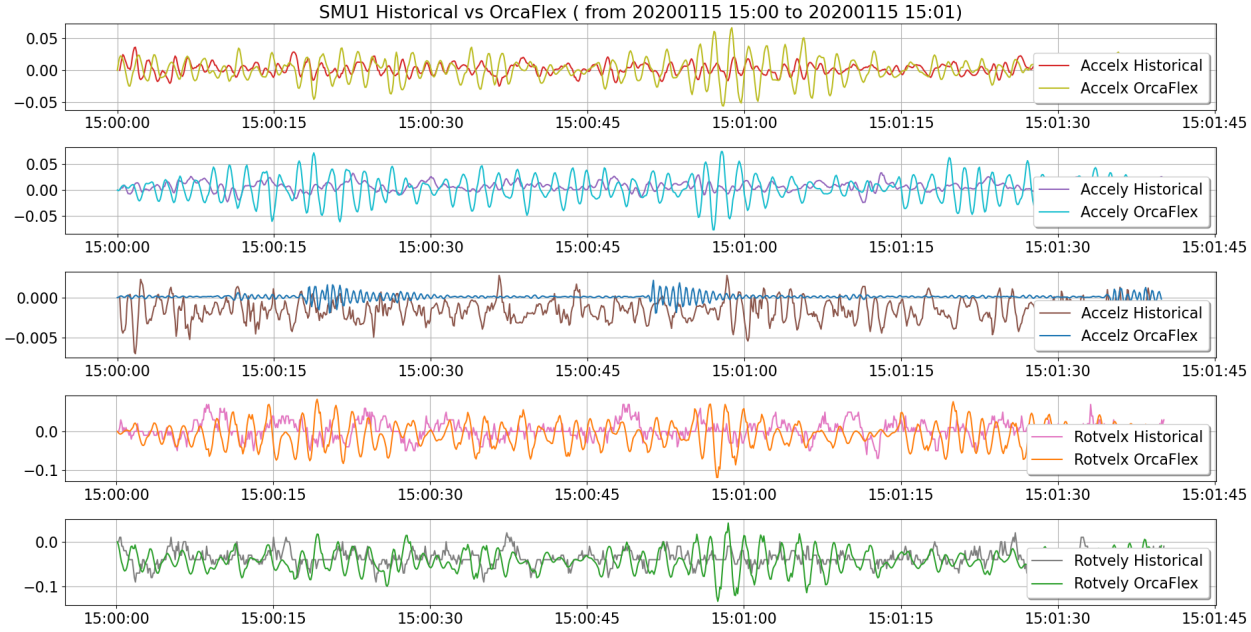


Figure B.5: SMU1 Historical vs OrcaFlex)

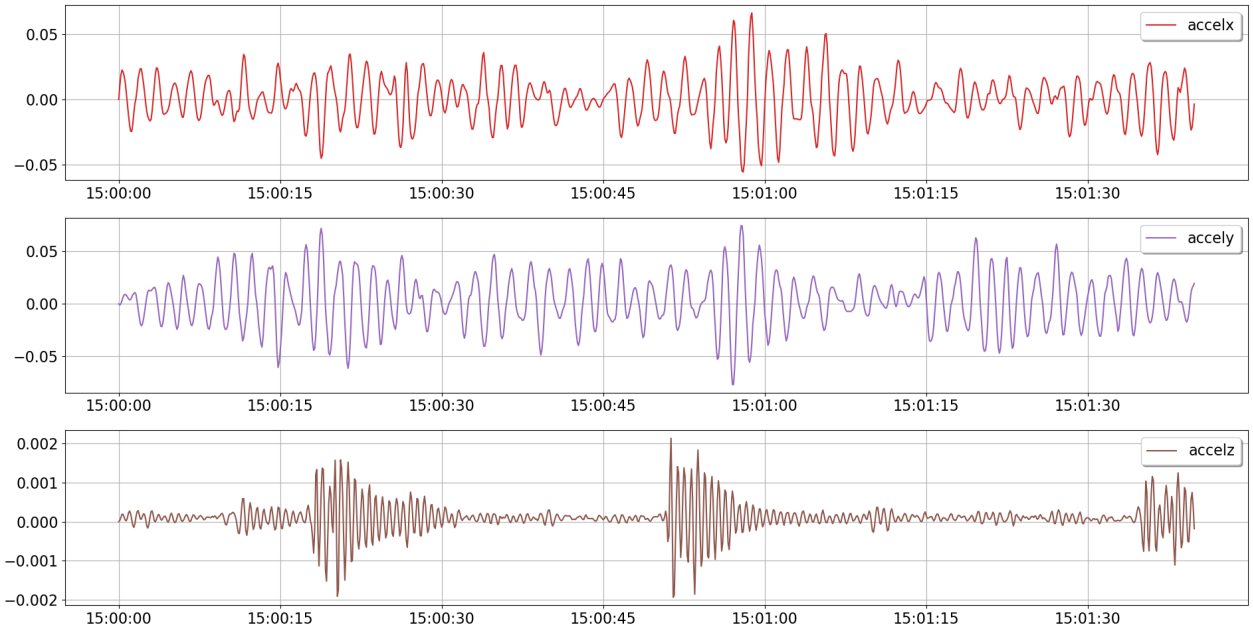


Figure B.6: SMU1 OrcaFlex acceleration (m/s^2)

Figure B.7: SMU1 OrcaFlex rotational velocity (deg/s)

SMU2-sensor

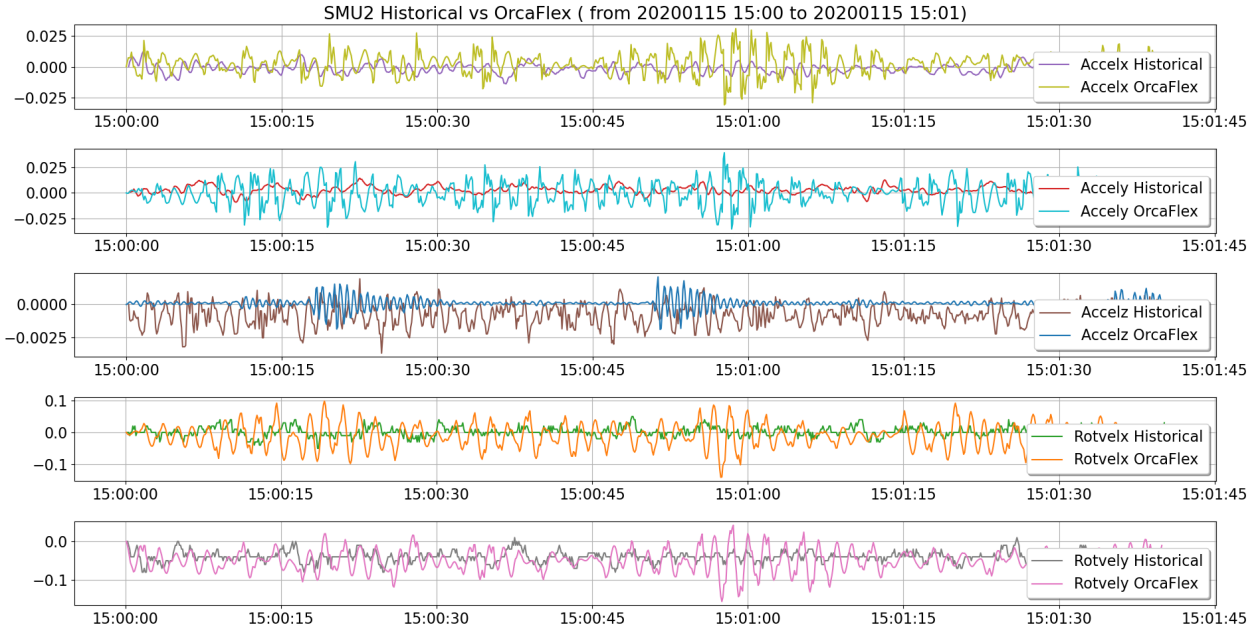


Figure B.8: SMU2 Historical vs OrcaFlex

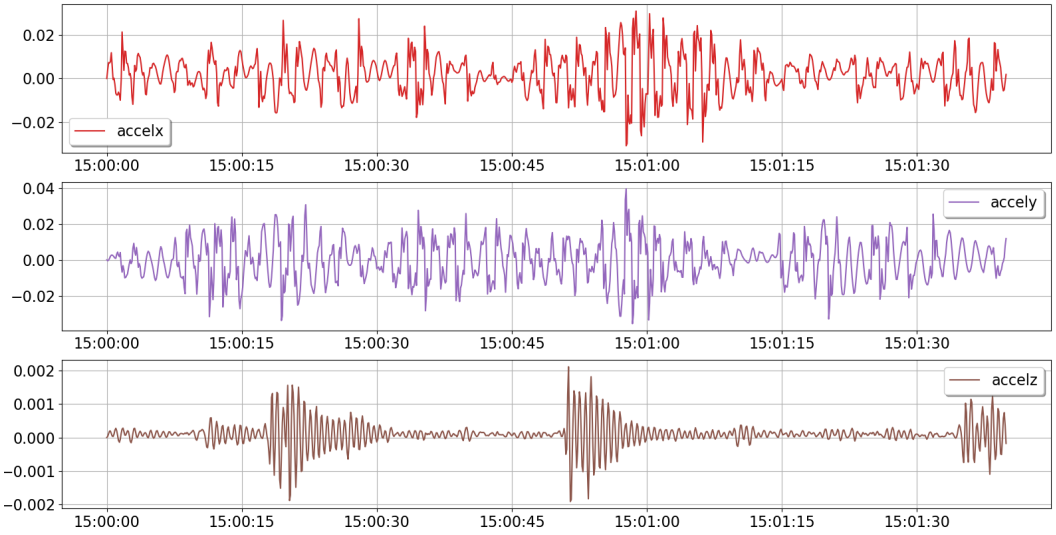


Figure B.9: SMU2 OrcaFlex acceleration (m/s^2)

Figure B.10: SMU2 OrcaFlex rotational velocity (deg/s)

SMU3-sensor

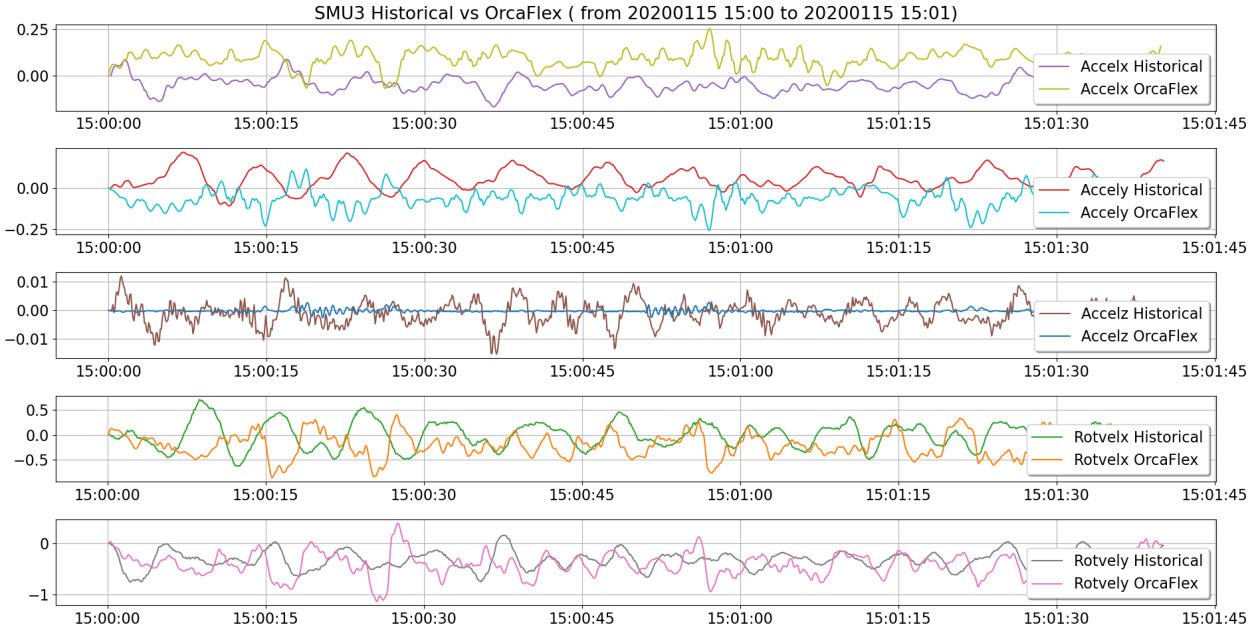


Figure B.11: SMU3 Historical vs OrcaFlex

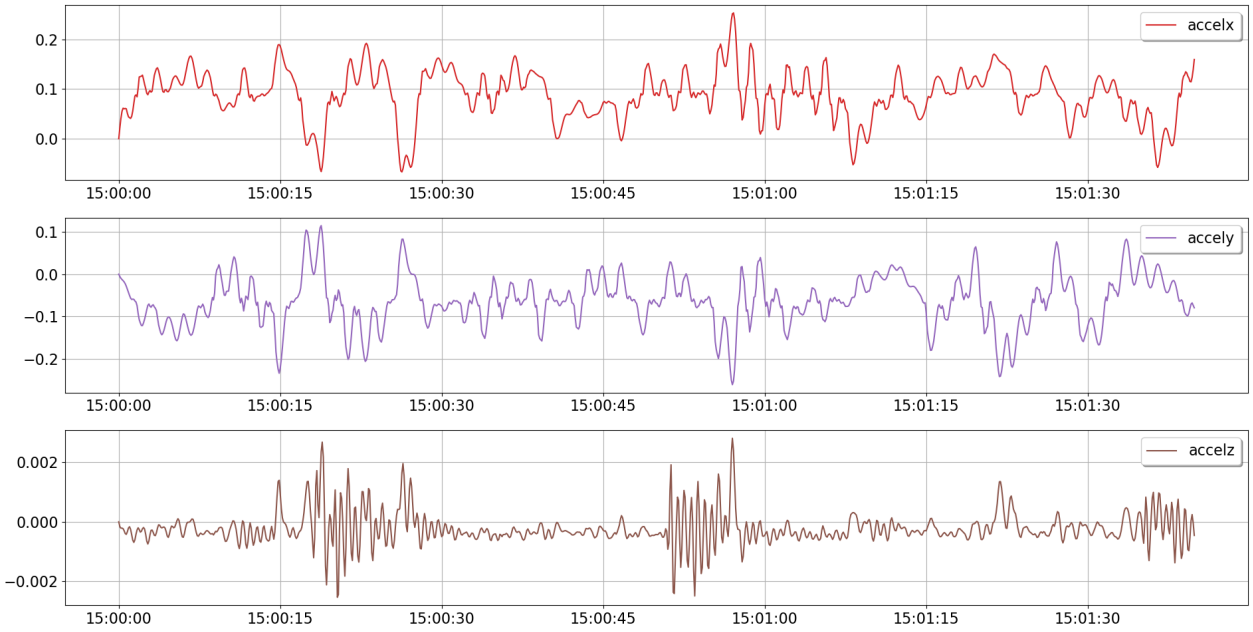


Figure B.12: SMU3 OrcaFlex acceleration (m/s^2)

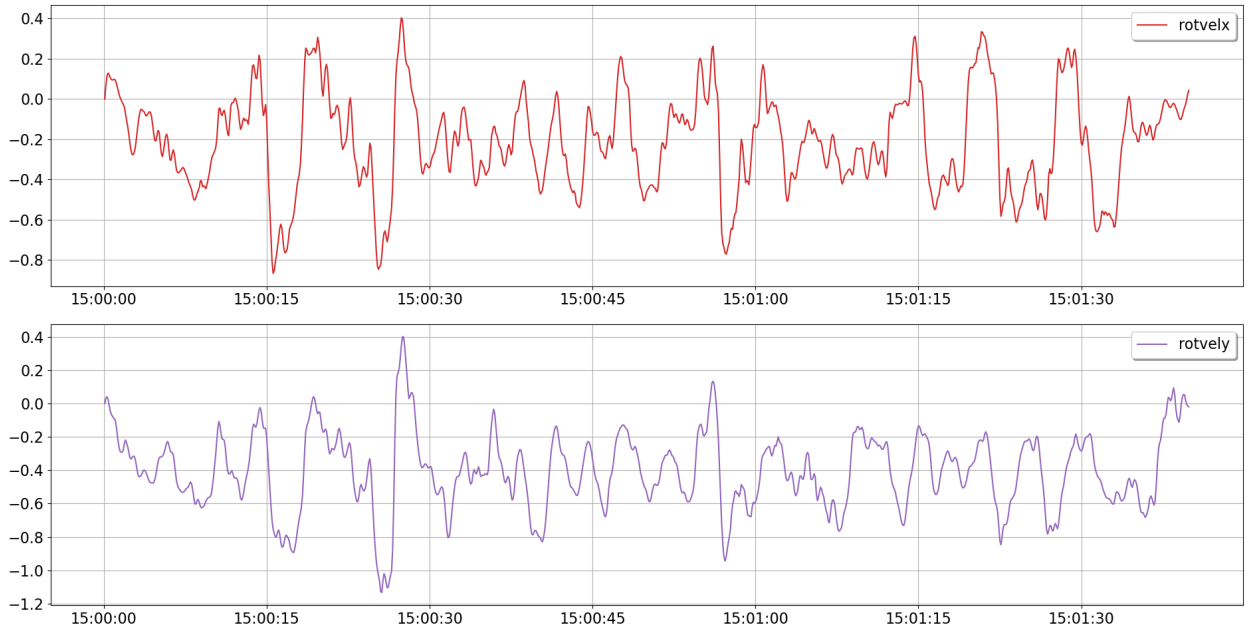


Figure B.13: SMU3 OrcaFlex rotational velocity (deg/s)

B.1.3 Spectral density analysis

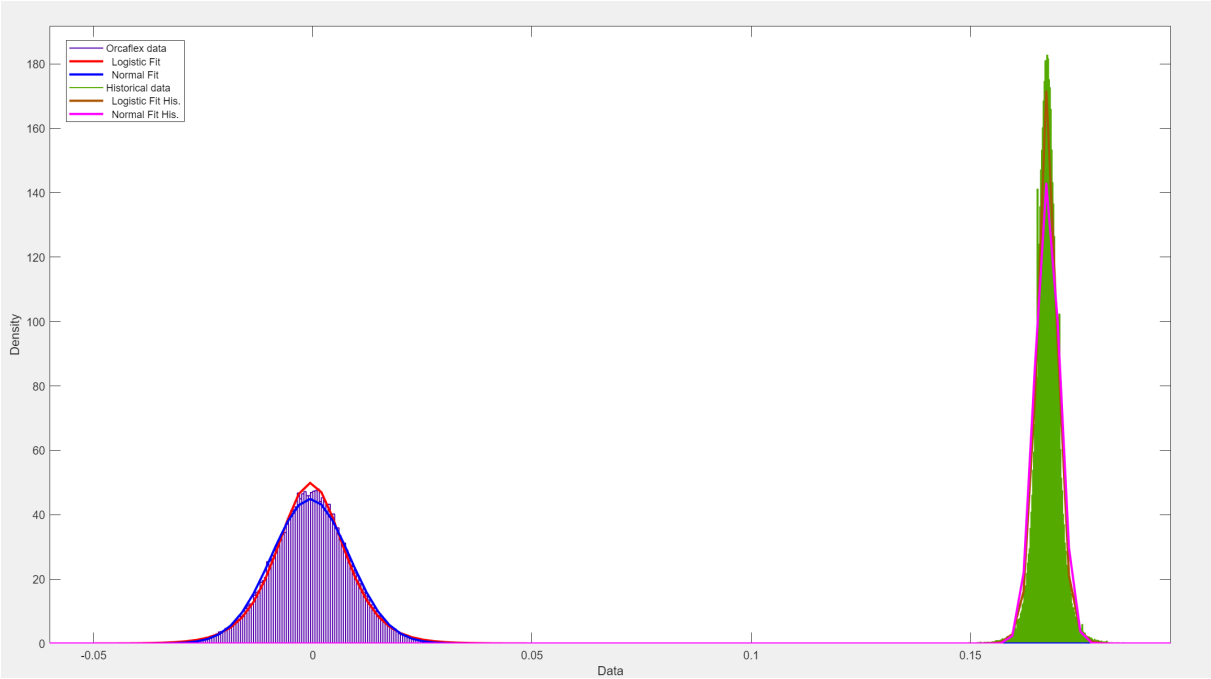


Figure B.14: Normal distribution, OrcaFlex and historical data

B.2 Sensor positions

B.2.1 COB sensors measurement area

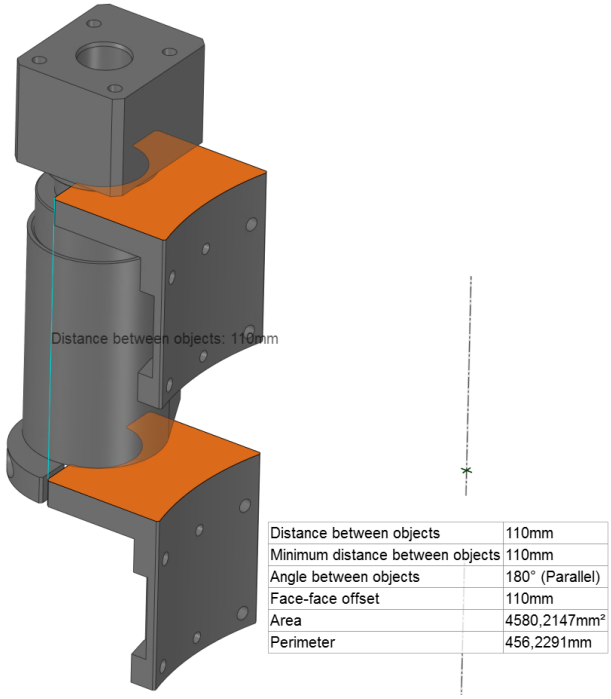
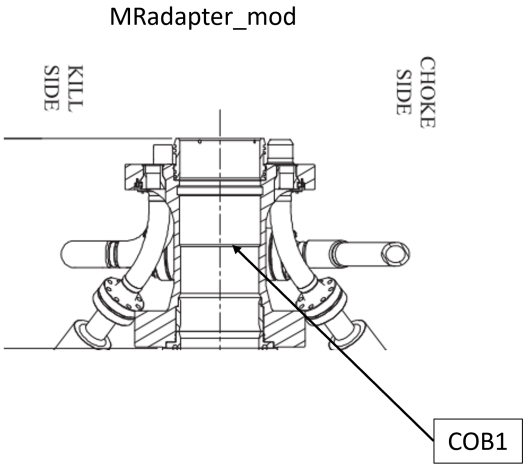


Figure B.15: COB sensors measurement area

B.2.2 COB sensors position (z)



COB1 position (z):

Middle of the MRadapter_mod

OrcaFlex MRadapter_mod length = 1,115 m

1,115 m / 2 = 0,5575 m

Pos. from WH datum:

11.403 m (BOP) + 1,463 m (RFJ) + 0,5575 m

= 13,4235 m

Figure B.16: COB1 position (z)

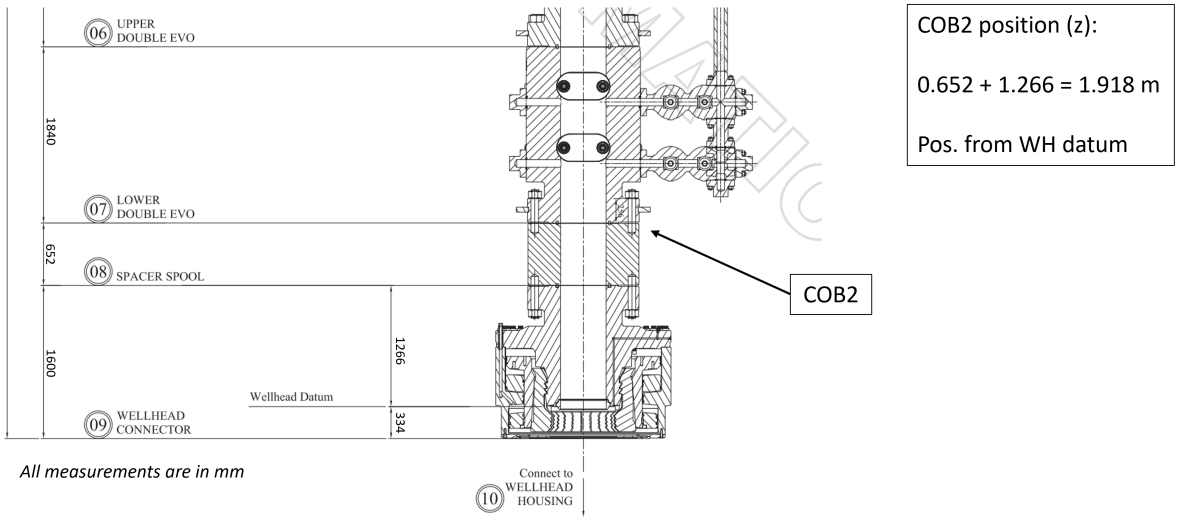


Figure B.17: COB2 position (z)

B.2.3 BOP local axis system, OrcaFlex compared to historical

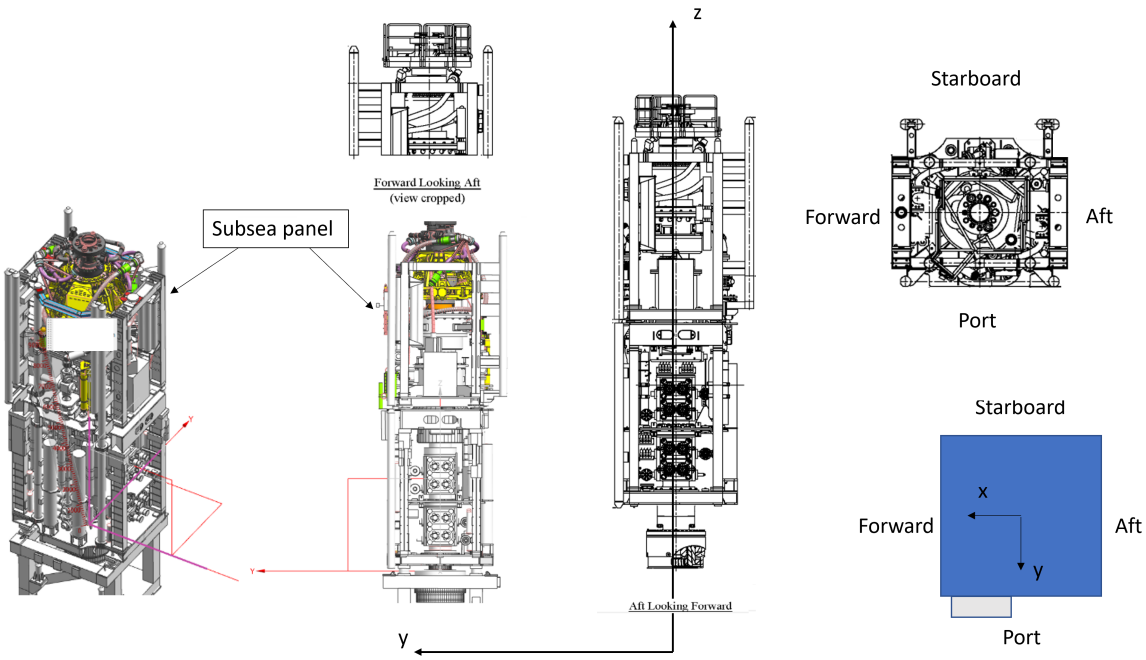


Figure B.18: BOP local axis, based on CAD models and schematics

B.2.4 Cardinal orientation

2.1.1 Location Data

Table 2-1: Template Position Data, Askeladd North

Template Position Data – Askeladd North [1]		
Askeladd North Template J	Heading	60°
	Water depth	270 m
	Easting	479 041 m
	Northing	7 922 339 m

Orientation = 60° + 180° = 240°

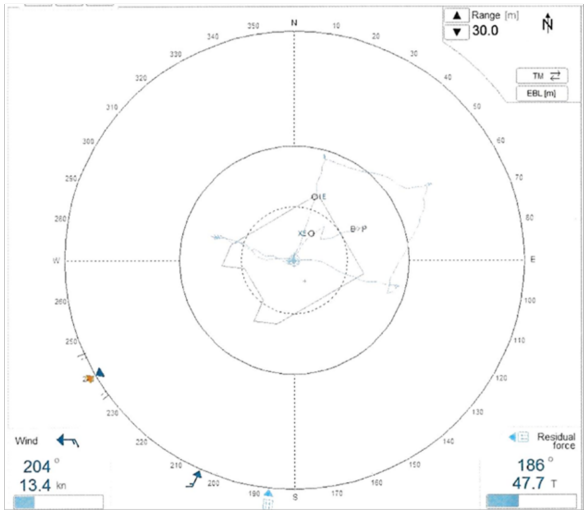


Figure B.19: Historical rig orientation relative to north

B.2.5 SMU historical orientation

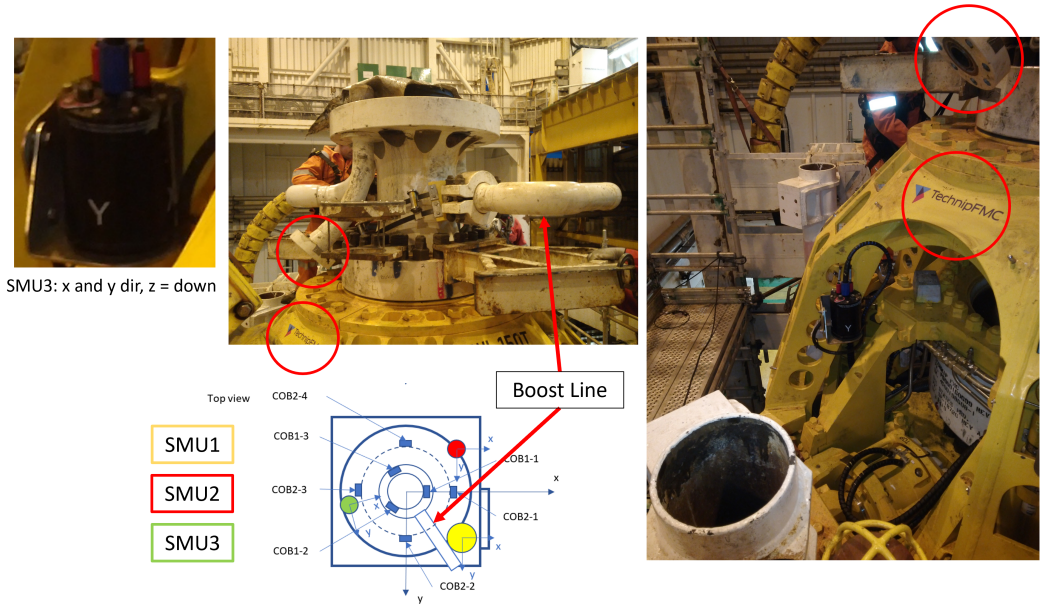


Figure B.20: SMU historical orientation, based on pictures from the actual assembly process.

B.3 OrcaFlex Tests

Extracted and converted data successfully

Table B.13: Askeladd overview

Con	Rev	Description	by
Con001	Rev001	Based on Projects2021\RFJstudy\Con006\Rev001	moldesl
Con002	Rev001	GOV file created from a simulation file (Askeladd J-1), OrcaFlex 11.0f.	msorensen
	Rev002	GOV file created from original GOV file (Askeladd J-1)	sberge
		Changed line sections (LFJline), and stress diameter (OD and ID) LineTypes (MRadapter_mod and BOPlowerstack). Lib/Vessels/DSA/Rev003b	
		Changed line sections (BOP), (RFJ_2kFJ_Spacer_fineMesh.yml), students2022\Askeladd\Input\BOP	
		Changed top end stiffness (Conductor) to []. Input\Wellhead\WHcomposite_WHconPara_IncludedWHincl.yml (11.2 update)	
		Changed top end stiffness (BOP) to []. Input\BOP\RFJ_2kFJ_Spacer_fineMesh.yml (11.2 update)	
		IncludeSeabedFrictionInStatics: Yes/No to StaticsSeabedFrictionPolicy: None. Put # in front of LayAzimuth: and AsLaidTension. Lines: BOP, MR and DrillPipe	
		Removed ConnectionDamping: MR, BOP, Wellhead	
Con002	Rev003	Added "WaveDirectionSpreadingExponent" to Torsethaugen.yml, created variables for wave number og spectral dir and spreadexp.	msorensen
		Renamed "WaveNumberOfComponents" to "WaveNumberOfComponentsPerDirection" in Torsethaugen.yml	
		Removed "WaveSpectrumMaxComponentFrequencyRange" from torsethaugen.yml	
Con003	Rev001	Setup RES file to be able to gather correct sensor data. SMU1, SMU2, SMU3, COB1, COB2. SMU3 needs changes.	sberge
		Changed line sections (BOP). (RFJ_2kFJ_Spacer_fineMesh.yml), students2022\Askeladd\Input\BOP	
		Added a bouy "SMU3" attached to the line "LFJline". Used to gather SMU3 data.	
	Rev002	Added two buoys "SMU2 and SMU1" attached to the line "BOP". Used to gather SMU2 and SMU1 data.	
		Added a new TorsetHaugen file, this gives the option to turn on and off the irregular seastate. @SeaModel: TorsetHaugen = off, TorsetHaugen_IS = on.	

	Rev003	Implemented changes to Sensor locations based on new knowledge regarding the relationship between the historical and OrcaFlex global axis.	
	Rev004	First section (0000 - 1200) of a "24 hours" simulation (Stage1 = 10000s (max. time) = 2h45m, historical = periods of 3h) (30.12.2019, RFJ on)	
	Rev005	Second section (1200 - 0000) of a "24 hours" simulation (Stage1 = 10000s (max. time) = 2h45m, historical = periods of 3h) (30.12.2019, RFJ on)	
	Rev006	"24 hours" simulation, (29.12.2019, RFJ on)	msorensen
	Rev007	Orientated the bouys (SMU1, SMU2 and SMU3) axis to match global axis [0, 0, 0] to [180, 0, 180]. Changed both COB sensors in the RES file, based on new knowledge regarding the historical BOP axis.	sberge
	Rev008	Based on Con003Rev006. 24 hours simulation, (15.01.2020, RFJ off)	msorensen
	Rev009	24 hours simulation, (16.01.2020, RFJ off)	
	Rev010	24 hours simulation, (17.01.2020, RFJ off)	
	Rev011	24 hours simulation, (18.01.2020, RFJ off)	
	Rev012	24 hours simulation, (19.01.2020, RFJ off)	
	Rev013	24 hours simulation, (20.01.2020, RFJ off)	
	Rev014	24 hours simulation, (21.01.2020, RFJ off)	
	Rev015	24 hours simulation, (22.01.2020, RFJ off)	
Con004	Rev001	Based on Con003Rev007. Removed "Env" and moved the WaveDir and CurDir down to the weather section (Use of TOTALDIRM from historical weather data). Simulation "24 hours" 29.12.2019 with irregular seastate (_IS)	sberge
	Rev002	Simulation "24 hours" 29.12.2019, RFJ on, without irregular seastate.	
	Rev003	Same as Rev002, just 400s runtime, 1 sim historical time 0000	
	Rev004	Same as Rev001, just 400s runtime, 1 sim historical time 0000	
	Rev005	24 hours simulation, (15.01.2020, RFJ on)	msorensen
	Rev006	24 hours simulation, (16.01.2020, RFJ on)	
	Rev007	24 hours simulation, (17.01.2020, RFJ on)	
	Rev008	24 hours simulation, (18.01.2020, RFJ on)	
	Rev009	24 hours simulation, (19.01.2020, RFJ on)	
	Rev010	24 hours simulation, (20.01.2020, RFJ on)	
	Rev011	24 hours simulation, (21.01.2020, RFJ on)	
	Rev012	24 hours simulation, (22.01.2020, RFJ on)	
	Rev013	3 hours simulation changed WH stiffness (UB ->LB) (29.12.2019 kl.0000). Added accelz to all SMUs (RES file)	sberge
	Rev014	Based on Con004Rev013, increased WH stiffness on both UB and LB by 30%. (29.12.2019 kl.0000). 400s * 2. (Test, changing default model WH stiffness)	

Askeladd Overview - Continued from last page

Con005	Rev001	Adjusted the COB2 position more accurately to historical setup. 24 hours simulation, (28.06.2020, RFJ on)	
	Rev002	24 hours simulation, (29.06.2020, RFJ on)	
	Rev003	24 hours simulation, (30.06.2020, RFJ on)	
	Rev004	24 hours simulation, (12.06.2020, RFJ on)	
	Rev005	24 hours simulation, (13.06.2020, RFJ on)	
	Rev006	24 hours simulation, (14.06.2020, RFJ on)	
	Rev007	24 hours simulation, (01.06.2020, RFJ on)	
	Rev008	24 hours simulation, (02.06.2020, RFJ on)	
	Rev009	24 hours simulation, (03.06.2020, RFJ on)	
Con006	Rev001	24 hours simulation, (30.06.2020, RFJ on)	msorensen
	Rev002	24 hours simulation, (12.06.2020, RFJ on)	
	Rev003	24 hours simulation, (13.06.2020, RFJ on)	
	Rev004	24 hours simulation, (14.06.2020, RFJ on)	
	Rev005	24 hours simulation, (01.06.2020, RFJ on)	
	Rev006	24 hours simulation, (02.06.2020, RFJ on)	
Con007	Rev001	Fixed SMU positions, and orientation to match historical setup. (Use this for any further work on the model)	sberge
	Rev002	"24 hours" simulation, (29.12.2019, RFJ on) rerun due to discovery of major bug in the mergeFiles script.	

B.3.1 Preservation of original model lengths

LFJ line

Original model:

- Total length, LFJabvFlex2k = 1.463 m
- Total length, LFJ Line = 1.463 m

The original model's LFJ Line setup, can be seen in table B.14

Table B.14: Original LFJ Line setup

LineType	Length (m)	TSL
LFJabvFlex2k	1.463	0.2

We changed the LFJ Line to have a higher amount of line segments, around the z-axis position of SMU3. This resulted in the setup in table B.15

Table B.15: New LFJ line setup

LineType	Length (m)	TSL
LFJabvFlex2k	1.179	0.2
LFJabvFlex2k	0.01	0.01
LFJabvFlex2k	0.274	0.2

New model:

- Total length, LFJabvFlex2k = (1.179 + 0.01 + 0.274)m = 1.464 m
- Total length, LFJ Line = 1.464 m

BOP line

Original model:

- Total length, LFJblwFlex2k = (0.420 + 0.206)m = 0.626 m
- Total length, LMRP = (2.664 + 0.01)m = 2.674 m
- Total length, BOPlowerstack = (8.093 + 0.01)m = 8.103 m
- Total length, BOP Line = (0.626 + 2.674 + 8.103)m = 11.403 m

The original model's BOP line setup, can be seen in table B.16

Table B.16: Original LFJ line setup

LineType	Length (m)	TSL
LFJblwFlex2k	0.420	0.5

LFJblwFlex2k	0.206	0.2
LMRP	2.664	0.5
LMRP	0.01	0.01
BOPlowerstack	0.01	0.01
BOPlowerstack	8.093	2

We changed the BOP line to have a higher amount of line segments, around the z-axis position of SMU1, SMU2 and COB2. This resulted in the setup in table B.17

Table B.17: Original BOP line setup

LineType	Length (m)	TSL
LFJblwFlex2k	0.420	0.5
LFJblwFlex2k	0.206	0.2
LMRP	2.584	0.2
LMRP	0.01	0.01
LMRP	0.08	0.02
BOPlowerstack	5.905	0.2
BOPlowerstack	0.01	0.01
BOPlowerstack	0.165	0.055
BOPlowerstack	0.11	0.1
BOPlowerstack	1.913	0.2

New model:

- Total length, LFJblwFlex2k = (0.420 + 0.206)m = 0.626 m
- Total length, LMRP = (2.584 + 0.01 + 0.08)m = 2.674 m
- Total length, BOPlowerstack = (5.905 + 0.01 + 0.165 + 0.11 + 1.913)m = 8.103 m
- Total length, BOP Line = (0.626 + 2.674 + 8.103)m = 11.403 m

MR line

Original model:

- Total length, MRadapter_mod = (1.015 + 0.1)m = 1.115 m

We only changed the bottom line sections with the Line Type = MRadapter_mod, the rest of the MR remains identical to the original model. These line sections are representing the MRA component.

The original model's MR Line setup, with a Line Type = MRadapter_mod, can be seen in table B.18

Table B.18: Original MR Line setup

LineType	Length (m)	TSL
MRadapter_mod	1.015	0.2
MRadapter_mod	0.1	0.1

We changed the MR line to have a higher amount of line segments, around the z-axis position of COB1. This resulted in the setup in table B.19

Table B.19: New MR Line setup

LineType	Length (m)	TSL
MRadapter_mod	0.5025	0.1
MRadapter_mod	0.11	0.1
MRadapter_mod	0.5025	0.1

New model:

- Total length, MRadapter_mod = (0.5025 + 0.11 + 0.5025)m = 1.115 m

C Software and machinie learning

C.1 Architecture

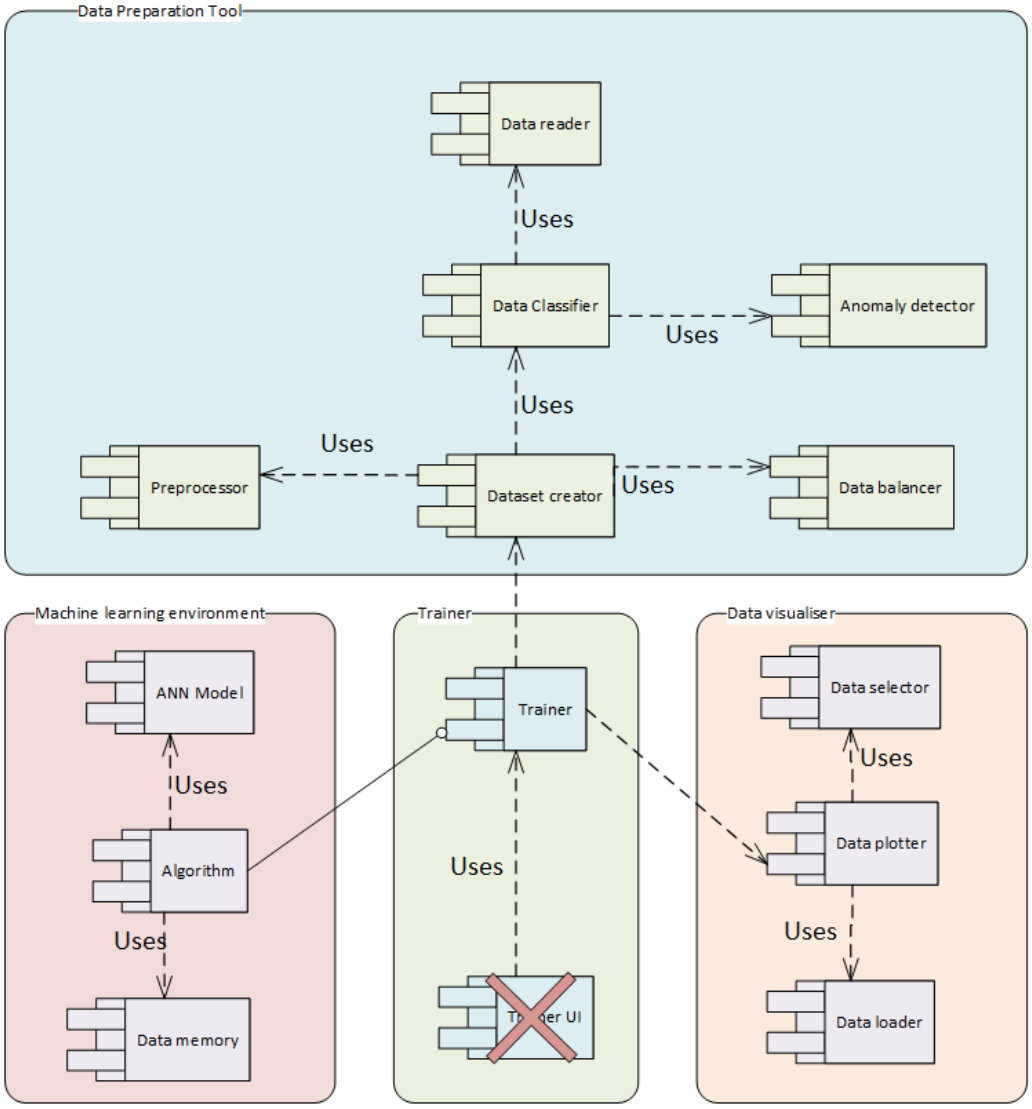


Figure C.1: High level architecture design in phase one

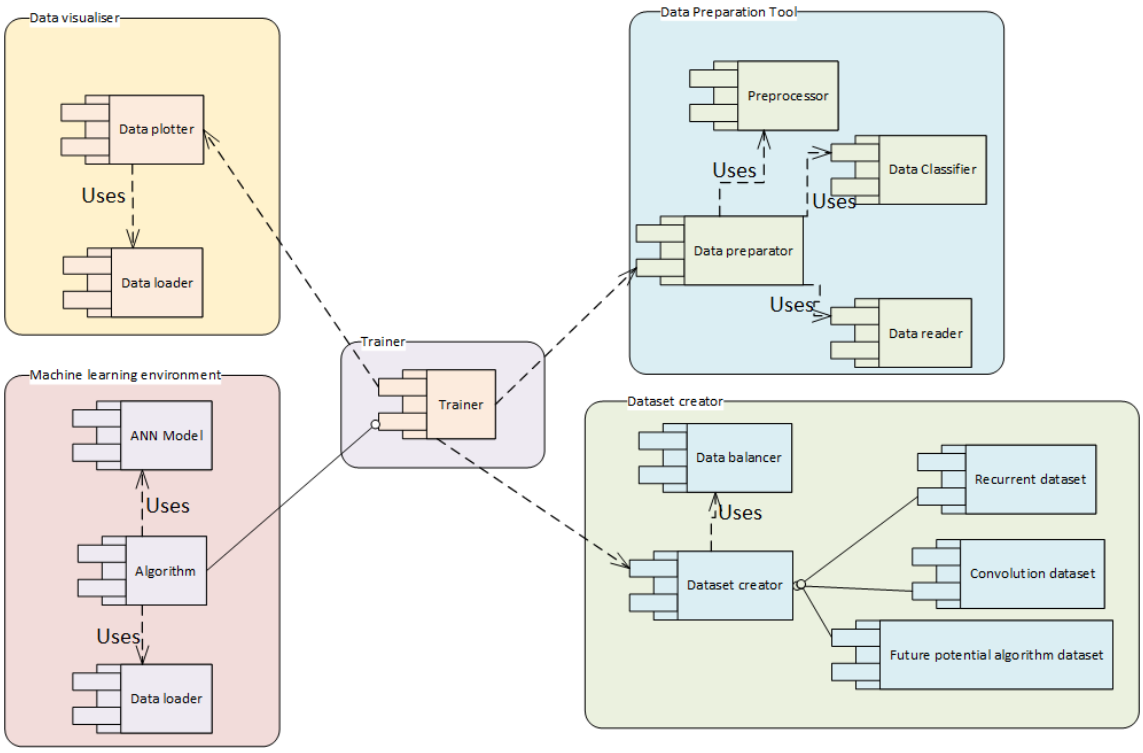


Figure C.2: High level architecture design in phase two

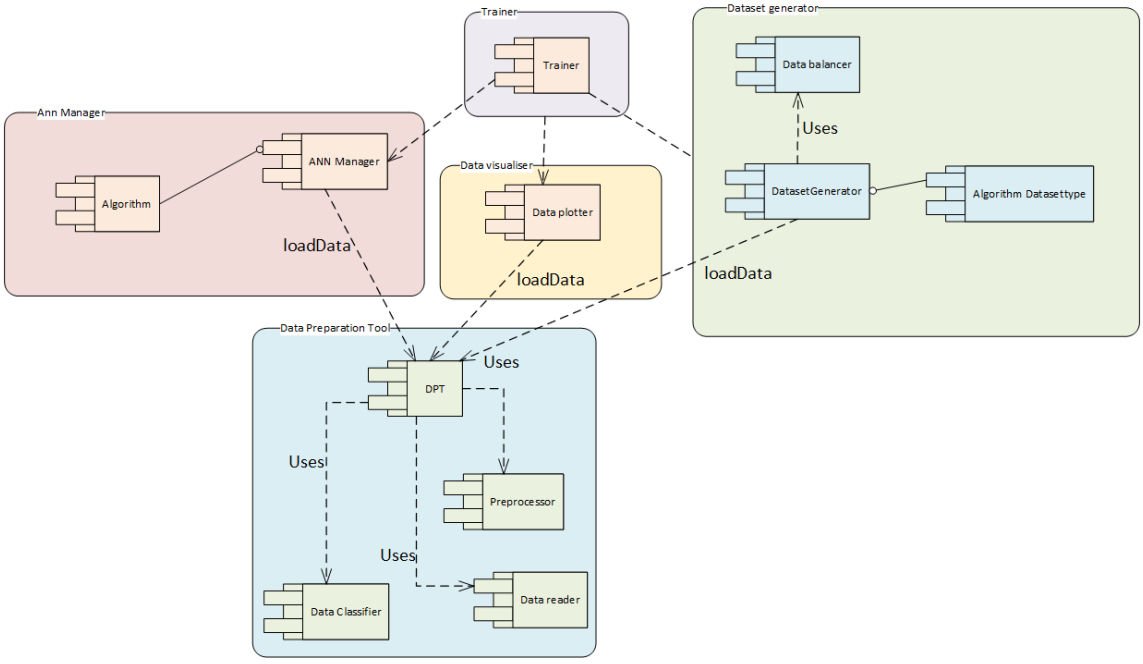


Figure C.3: High level architecture design in phase three

C.2 Machine learning Results

C.2.1 RNN results

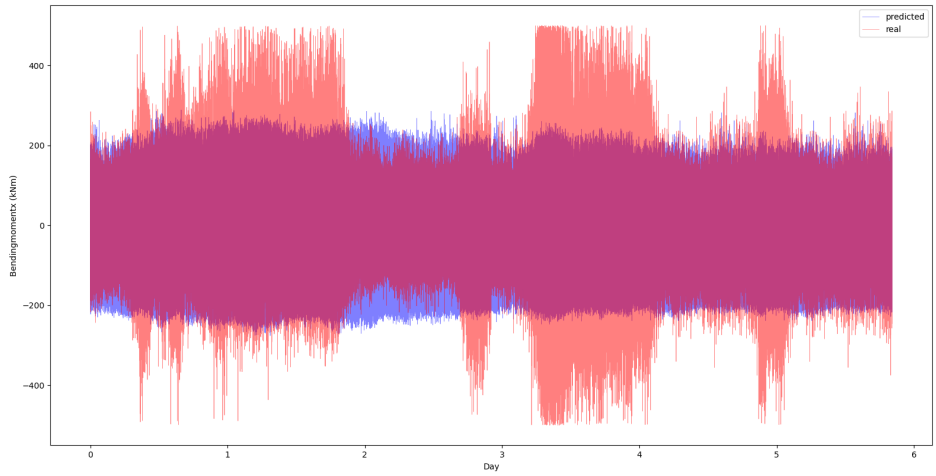


Figure C.4: Deep RNN trained on orcflex data. Valdiated on six days of historical data

D Mathematical Model

D.1 Full derivation of the system

D.1.1 Kinetic energy

The kinetic energy (T) can be expressed as in equation

$$T = \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2 \tag{D.1}$$

where m is the mass and v is the velocity.
We can express the velocity as the time-derivative of the positions

$$v = \dot{x} + \dot{z}$$

allowing us to express equation D.1 in the following manner

$$T = \frac{1}{2}m_1(\dot{x}_1 + \dot{z}_1)^2 + \frac{1}{2}m_2(\dot{x}_2 + \dot{z}_2)^2$$

Using the relationship $(a+b)^2 = a^2 + 2ab + b^2$

$$T = \frac{1}{2}m_1(\dot{x}_1^2 + 2\dot{x}_1\dot{z}_1 + \dot{z}_1^2) + \frac{1}{2}m_2(\dot{x}_2^2 + 2\dot{x}_2\dot{z}_2 + \dot{z}_2^2) \tag{D.2}$$

We can now insert the positional equations (4.92, 4.93, 4.94, 4.95) into the expression for kinetic energy (D.2)

$$T = \left(\frac{1}{2}m_1\left((\ell_1 \sin\dot{\theta}_1)^2 + 2(\ell_1 \sin\dot{\theta}_1 \cdot \ell_1 \cos\dot{\theta}_1) + (\ell_1 \cos\dot{\theta}_1)^2\right)\right) +$$
$$\left(\frac{1}{2}m_2\left((\ell_1 \sin\theta_1 + \ell_2 \sin\theta_2)^2 + 2((\ell_1 \sin\theta_1 + \ell_2 \sin\theta_2) \cdot ((\ell_1 \cos\theta_1 + \ell_2 \cos\theta_2)) + (\ell_1 \cos\theta_1 + \ell_2 \cos\theta_2)^2\right)\right)$$

Using the trigonometric identify $cos^2 + sin^2 = 1$ we get

$$T = \frac{1}{2}m_1\ell_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2\left[\ell_1^2\dot{\theta}_1^2 + \ell_2^2\dot{\theta}_2^2 + 2\ell_1\ell_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2)\right] \tag{D.3}$$

which can be expressed as

$$T = \frac{1}{2}(m_1 + m_2)\ell_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2\ell_2^2\dot{\theta}_2^2 + m_2\ell_1\ell_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2) \tag{D.4}$$

D.1.2 Euler-Lagrange equations

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_1}\right) - \frac{\partial\mathcal{L}}{\partial\theta_1} = 0 \tag{D.5}$$

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}_2}\right) - \frac{\partial\mathcal{L}}{\partial\theta_2} = 0 \tag{D.6}$$

Starting with the first Lagrangian differential equation (D.5), we begin by deriving the Lagrangian (4.101) with respect to θ_1

$$\frac{\partial\mathcal{L}}{\partial\theta_1} = (m_1 + m_2)\ell_1 g\sin\theta_1 - m_2\ell_1\ell_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) \tag{D.7}$$

Next, we derive our Lagrangian (4.101) with respect to $\dot{\theta}_1$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = (m_1 + m_2)\ell_1^2\dot{\theta}_1 + m_2\ell_1\ell_2\dot{\theta}_2\cos(\theta_1 - \theta_2) \tag{D.8}$$

and then we take the time derivative of (D.8)

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}\right) = (m_1 + m_2)\ell_1^2\ddot{\theta}_1 + m_2\ell_1\ell_2\ddot{\theta}_2\cos(\theta_1 - \theta_2) - m_2\ell_1\ell_2\dot{\theta}_2\sin(\theta_1 - \theta_2)(\dot{\theta}_1 - \dot{\theta}_2) \tag{D.9}$$

Substituting (D.7, D.9) into (D.5) yields the first Lagrangian differential equation (D.10)

$$\begin{aligned}(m_1 + m_2)\ell_1^2\ddot{\theta}_1 + m_2\ell_1\ell_2\ddot{\theta}_2\cos(\theta_1 - \theta_2) \\ + m_2\ell_1\ell_2\dot{\theta}_1^2\sin(\theta_1 - \theta_2) + \ell_1(m_1 + m_2)g\sin\theta_1 = 0\end{aligned} \tag{D.10}$$

We repeat the operation by deriving the Lagrangian (4.101) with respect to θ_2

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = -\ell_2 m_2 g\sin\theta_2 + m_2\ell_1\ell_2\dot{\theta}_1\dot{\theta}_2\sin(\theta_1 - \theta_2) \tag{D.11}$$

and then deriving the Lagrangian (4.101) with respect to $\dot{\theta}_2$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = m_2\ell_2^2\dot{\theta}_2 + m_2\ell_1\ell_2\dot{\theta}_1\cos(\theta_1 - \theta_2) \tag{D.12}$$

and then we take the time derivative (D.12)

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}\right) = m_2\ell_2\ddot{\theta}_2 + m_2\ell_1\ell_2\ddot{\theta}_1\cos(\theta_1 - \theta_2) - m_2\ell_1\ell_2\dot{\theta}_1\sin(\theta_1 - \theta_2)(\dot{\theta}_1 - \dot{\theta}_2) \tag{D.13}$$

Substituting (D.11, D.13) into (D.6) yields the second Lagrangian differential equation (D.14)

$$m_2\ell_2^2\ddot{\theta}_2 + m_2\ell_1\ell_2\ddot{\theta}_1\cos(\theta_1 - \theta_2) - m_2\ell_1\ell_2\dot{\theta}_1^2\sin(\theta_1 - \theta_2) + \ell_2 m_2 g\sin\theta_2 = 0 \tag{D.14}$$

D.1.3 Hamiltonian

We begin by using the expressions for generalized momentum (4.106, 4.107), and write them on matrix form (D.15)

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = K\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}, \quad K = \begin{bmatrix} (m_1 + m_2)\ell_1^2 & m_2\ell_1\ell_2\cos(\theta_1 - \theta_2) \\ m_2\ell_1\ell_2\cos(\theta_1 - \theta_2) & m_2\ell_2^2 \end{bmatrix} \tag{D.15}$$

The kinetic energy (4.100) can be expressed as

$$T = \frac{1}{2}\begin{bmatrix}\dot{\theta}_1 & \dot{\theta}_2\end{bmatrix} K \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \frac{1}{2}\begin{bmatrix}\dot{\theta}_1 & \dot{\theta}_2\end{bmatrix}\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \frac{1}{2}(\dot{\theta}_1 p_1 + \dot{\theta}_2 p_2) \tag{D.16}$$

and $K^T = K$, so the inverse transformation becomes

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = K^{-1}\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \tag{D.17}$$

$$K^{-1} = \frac{1}{m_2\ell_1^2\ell_2^2[m_1 + m_2 - m_2\cos^2(\theta_1 - \theta 2)]}$$
$$\begin{bmatrix} m_2\ell_2^2 & -m_2\ell_1\ell_2\cos(\theta_1 - \theta_2) \\ -m_2\ell_1\ell_2\cos(\theta_1 - \theta_2) & (m_1 + m_2)\ell_1^2 \end{bmatrix} \quad \text{(D.18)}$$

$(K^{-1})^T = K^{-1}$, so using the relation

$$\begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 \end{bmatrix} (K^{-1})^T = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} K^{-1} \quad \text{(D.19)}$$

the kinetic energy can be written as

$$T = \frac{1}{2} \begin{bmatrix} p_1 & p_2 \end{bmatrix} K^{-1} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \quad \text{(D.20)}$$

Substituting the kinetic energy (D.20) and the potential energy (4.97) into the Hamiltonian (4.85) can now be expressed as

$$\mathcal{H} = T + V = \frac{1}{2} \begin{bmatrix} p_1 & p_2 \end{bmatrix} K^{-1} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} - (m_1 + m_2)g\ell_1\cos\theta_1 - m_2g\ell_2\cos\theta_2 \quad \text{(D.21)}$$

which on standard format becomes (4.89) [61, p.10]

$$\mathcal{H} = \frac{m_2\ell_2^2p_1^2 + \ell_1^2(m_1 + m_2)p_2^2 - 2m_2\ell_1\ell_2p_1p_2\cos(\theta_1 - \theta_2)}{2\ell_1^2\ell_2^2m_2[(m_1 + m_2)\sin^2(\theta_1 - \theta_2)]} - (m_1 + m_2)g\ell_1\cos\theta_1 - m_2g\ell_2\cos\theta_2$$

D.2 Small angle approximation

D.2.1 Kinetic energy

For the kinetic energy, we use the fact that

$$\cos(\theta_1 - \theta_2) \approx \cos 0 = 1$$

and that

$$\dot{\theta}_1^2\ell_1^2 + 2\dot{\theta}_1\ell_1\dot{\theta}_2\ell_2 + \dot{\theta}_2^2\ell_1^2 = (\dot{\theta}_1\ell_1 + \dot{\theta}_2\ell_2)^2$$

to get the expression for kinetic energy (D.22)

$$T = \frac{1}{2}m_1\ell_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2(\dot{\theta}_1\ell_1 + \dot{\theta}_2\ell_2)^2 \tag{D.22}$$

We then expand (D.22) to get (4.116) repeated below

$$T = \frac{l_1^2 m_1\dot{\theta}_1^2}{2} + \frac{l_1^2 m_2\dot{\theta}_1^2}{2} + l_1 l_2 m_2\dot{\theta}_1\dot{\theta}_2 + \frac{l_2^2 m_2\dot{\theta}_2^2}{2}$$

D.2.2 Potential energy

For the potential energy, we use that

$$\cos\theta_1 \approx \left(1 - \frac{1}{2}\theta_1^2\right)$$

and that

$$\cos\theta_2 \approx \left(1 - \frac{1}{2}\theta_2^2\right)$$

to get the expression for potential energy (D.23)

$$V = \frac{1}{2}m_1 g\ell_1\theta_1^2 + \frac{1}{2}m_2 g(\ell_1\theta_1^2 + \ell_2\theta_2^2) - m_1 g\ell_1 - m_2 g(\ell_1 + \ell_2) \tag{D.23}$$

We then expand (D.23) to get (4.115) repeated below

$$V = \frac{gl_1 m_1\theta_1^2}{2} - gl_1 m_1 + \frac{gl_1 m_2\theta_1^2}{2} - gl_1 m_2 + \frac{gl_2 m_2\theta_2^2}{2} - gl_2 m_2$$

D.2.3 Euler-Lagrange equations

Substituting the energy equations (D.22, D.23) into the expression for the Lagrangian ($\mathcal{L} = T - V$) yields a new Lagrangian expression (D.24)

$$\mathcal{L} = \frac{1}{2}m_1\ell_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2(\dot{\theta}_1\ell_1 + \dot{\theta}_2\ell_2)^2 - \frac{1}{2}m_1 g\ell_1\theta_1^2 - \frac{1}{2}m_2 g(\ell_1\theta_1^2 + \ell_2\theta_2^2) + m_1 g\ell_1 + m_2 g(\ell_1 + \ell_2) \tag{D.24}$$

We expand (D.24) to get (4.117) repeated below

$$\mathcal{L} = -\frac{gl_1 m_1\theta_1^2}{2} + gl_1 m_1 - \frac{gl_1 m_2\theta_1^2}{2} + gl_1 m_2 - \frac{gl_2 m_2\theta_2^2}{2} + gl_2 m_2 + \frac{l_1^2 m_1\dot{\theta}_1^2}{2} + \frac{l_1^2 m_2\dot{\theta}_1^2}{2} + l_1 l_2 m_2\dot{\theta}_1\dot{\theta}_2 + \frac{l_2^2 m_2\dot{\theta}_2^2}{2}$$

Starting with the first Lagrangian differential equation (D.5), we begin by deriving the Lagrangian (4.117) with respect to θ_1

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = -(m_1 + m_2) g \ell_1 \theta_1 \tag{D.25}$$

Next, we derive our Lagrangian (4.117) with respect to $\dot{\theta}_1$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = m_1 \ell_1^2 \dot{\theta}_1 + m_2 \ell_1 (\dot{\theta}_1 \ell_1 + \dot{\theta}_2 \ell_2) \tag{D.26}$$

and then we take the time derivative of (D.26)

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_1}\right) = m_1 \ell_1^2 \ddot{\theta}_1 + m_2 \ell_1 (\ddot{\theta}_1 \ell_1 + \ddot{\theta}_2 \ell_2) \tag{D.27}$$

Substituting (D.25, D.27) into (D.5) yields the first Lagrangian differential equation (D.28)

$$m_1 \ell_1^2 \ddot{\theta}_1 + m_2 \ell_1 (\ddot{\theta}_1 \ell_1 + \ddot{\theta}_2 \ell_2) = -(m_1 + m_2) g \ell_1 \theta_1 \tag{D.28}$$

We expand (D.28) to get (4.118) repeated below

$$l_1 \left(g m_1 \theta_1 + g m_2 \theta_1 + l_1 m_1 \ddot{\theta}_1 + l_1 m_2 \ddot{\theta}_1 + l_2 m_2 \ddot{\theta}_2 \right) = 0$$

Continuing with the second Lagrangian differential equation (D.6), we begin by deriving the Lagrangian (4.117) with respect to θ_2

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = -m_2 g \ell_2 \theta_2 \tag{D.29}$$

Next, we derive our Lagrangian (4.117) with respect to $\dot{\theta}_2$

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = m_2 \ell_2 (\dot{\theta}_1 \ell_1 + \dot{\theta}_2 \ell_2) \tag{D.30}$$

and then we take the time derivative of (D.30)

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_2}\right) = m_2 \ell_2 (\ddot{\theta}_1 \ell_1 + \ddot{\theta}_2 \ell_2) \tag{D.31}$$

Substituting (D.29, D.31) into (D.6) yields the second Lagrangian differential equation (D.32)

$$m_2 \ell_2 (\ddot{\theta}_1 \ell_1 + \ddot{\theta}_2 \ell_2) = -m_2 g \ell_2 \theta_2 \tag{D.32}$$

We expand (D.32) to get (4.119) repeated below

$$l_2 m_2 \left(g \theta_2 + l_1 \ddot{\theta}_1 + l_2 \ddot{\theta}_2 \right) = 0$$

D.2.4 Generalized momentum

We begin by using the expressions for generalized momentum (4.120, 4.121) repeated below

$$p_1 = \frac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = l_1^2 m_1 \dot{\theta}_1 + l_1^2 m_2 \dot{\theta}_1 + l_1 l_2 m_2 \dot{\theta}_2$$

$$p_2 = \frac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = l_1 l_2 m_2 \dot{\theta}_1 + l_2^2 m_2 \dot{\theta}_2$$

We solve (4.120) for $\dot{\theta}_1$

$$\dot{\theta}_1 = \frac{-l_1 l_2 m_2 \dot{\theta}_2 + p_1}{l_1^2 (m_1 + m_2)} \tag{D.33}$$

Solving (4.121) for $\dot{\theta}_2$

$$\dot{\theta}_2 = -\frac{l_1 \dot{\theta}_1}{l_2} + \frac{p_2}{l_2^2 m_2} \tag{D.34}$$

Substituting (D.34) into (D.33) and solving for $\dot{\theta}_1$

$$\dot{\theta}_1 = \frac{-l_1 p_2 + l_2 p_1}{l_1^2 l_2 m_1} \tag{D.35}$$

Substituting (D.35) into (D.34) and solving for $\dot{\theta}_2$

$$\dot{\theta}_2 = \frac{p_2}{l_2^2 m_2} - \frac{-l_1 p_2 + l_2 p_1}{l_1 l_2^2 m_1} \tag{D.36}$$

D.2.5 Hamiltonian

Substituting (D.35) for $\dot{\theta}_1$ and (D.36) for $\dot{\theta}_2$ into our kinetic energy expression (4.116)

$$T = \frac{p_2^2}{2 l_2^2 m_2} + \frac{p_2^2}{2 l_2^2 m_1} - \frac{p_1 p_2}{l_1 l_2 m_1} + \frac{p_1^2}{2 l_1^2 m_1} \tag{D.37}$$

Now, we can express the Hamiltonian ($\mathcal{H} = T + V$) on standard form using the energy equations (D.37, 4.115)

$$\mathcal{H} = \left(\frac{(p_1 \ell_2 - p_2 \ell_1)^2}{2 m_1 \ell_1^2 \ell_2^2} + \frac{1}{2} m_2 \left(\frac{p_2 (p_2 m_1 \ell_1 \ell_2) - m_2 (p_1 \ell_2 - p_2 \ell_1))}{m_1 m_2 \ell_1 \ell_2^2} + \frac{p_1 (p_1 \ell_2 - p_2 \ell_1)}{m_1 \ell_1^2 \ell_2} \right)^2 \right) + \left(\frac{1}{2} m_1 g \ell_1 \theta_1^2 + \frac{1}{2} m_2 g (\ell_1 \theta_1^2 + \ell_2 \theta_2^2) - m_1 g \ell_1 - m_2 g (\ell_1 + \ell_2) \right) \tag{D.38}$$

we then simplify (D.38) to get the Hamiltonian (4.123) repeated below

$$\mathcal{H} = \frac{g l_1 m_1 \theta_1^2}{2} - g l_1 m_1 + \frac{g l_1 m_2 \theta_1^2}{2} - g l_1 m_2 + \frac{g l_2 m_2 \theta_2^2}{2} - g l_2 m_2 + \frac{p_2^2}{2 l_2^2 m_2} + \frac{p_2^2}{2 l_2^2 m_1} - \frac{p_1 p_2}{l_1 l_2 m_1} + \frac{p_1^2}{2 l_1^2 m_1}$$

D.3 Torsional springs

D.3.1 Euler-Lagrange equations

Substituting the potential energy (4.131) and the kinetic energy (4.116) into the Lagrangian equation ($\mathcal{L} = T - V$) yields

$$\mathcal{L} = -\frac{gl_1m_1\theta_1^2}{2} + gl_1m_1 - \frac{gl_1m_2\theta_1^2}{2} + gl_1m_2 - \frac{gl_2m_2\theta_2^2}{2} + gl_2m_2 - \frac{k_1\theta_1^2}{2} - \frac{k_2\theta_1^2}{2} + k_2\theta_1\theta_2 - \frac{k_2\theta_2^2}{2} + \frac{l_1^2m_1\dot{\theta}_1^2}{2} + \frac{l_1^2m_2\dot{\theta}_1^2}{2} + l_1l_2m_2\dot{\theta}_1\dot{\theta}_2 + \frac{l_2^2m_2\dot{\theta}_2^2}{2} \tag{D.39}$$

Starting with the first Lagrangian differential equation (D.5), we begin by deriving the Lagrangian (4.117) with respect to θ_1

$$\frac{\partial\mathcal{L}}{\partial\theta_1} = -gl_1m_1\theta_1 - gl_1m_2\theta_1 - k_1\theta_1 - k_2\theta_1 + k_2\theta_2 \tag{D.40}$$

The kinetic part of the Lagrangian differential equations (D.27) is not affected by the torsional springs, and the first Lagrangian differential equation (D.5) becomes (4.133), repeated below

$$gl_1m_1\theta_1 + gl_1m_2\theta_1 + k_1\theta_1 + k_2\theta_1 - k_2\theta_2 + l_1^2m_1\ddot{\theta}_1 + l_1^2m_2\ddot{\theta}_1 + l_1l_2m_2\ddot{\theta}_2 = 0$$

Continuing with the second Lagrangian differential equation (D.6), we begin by deriving the Lagrangian (4.117) with respect to θ_2

$$\frac{\partial\mathcal{L}}{\partial\theta_2} = -gl_2m_2\theta_2 + k_2\theta_1 - k_2\theta_2 \tag{D.41}$$

The kinetic part of the Lagrangian differential equations (D.31) is not affected by the torsional springs, and the first Lagrangian differential equation (D.6) becomes (4.134), repeated below

$$gl_2m_2\theta_2 - k_2\theta_1 + k_2\theta_2 + l_1l_2m_2\ddot{\theta}_1 + l_2^2m_2\ddot{\theta}_2 = 0$$

D.4 Damping

Using the expanded Euler-Lagrange equations including damper force (8.3, 8.4) repeated below

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta_\alpha}}\right) - \frac{\partial T}{\partial \theta_\alpha} + \frac{\partial R}{\partial \dot{\theta}_\alpha} + \frac{\partial V}{\partial \theta_\alpha} = \sum F_q$$

Starting with the first Euler-Lagrange equation (8.3) using the kinetic energy (4.122), potential energy (4.131) and viscous energy dissipation (8.2)

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta_1}}\right) = l_1^2 m_1 \ddot{\theta}_1 + l_1^2 m_2 \ddot{\theta}_1 + l_1 l_2 m_2 \ddot{\theta}_2 \tag{D.42}$$

$$\frac{\partial T}{\partial \theta_1} + \frac{\partial R}{\partial \dot{\theta}_1} + \frac{\partial V}{\partial \theta_1} = -g l_1 m_1 \theta_1 - g l_1 m_2 \theta_1 - k_1 \theta_1 - k_2 \theta_1 + k_2 \theta_2 + b\dot{\theta}_1 - b\dot{\theta}_2 \tag{D.43}$$

Substituting (D.42, D.43) into (8.3) yields (8.5), repeated below

$$-b\dot{\theta}_1 + b\dot{\theta}_2 + g l_1 m_1 \theta_1 + g l_1 m_2 \theta_1 + k_1 \theta_1 + k_2 \theta_1 - k_2 \theta_2 + l_1^2 m_1 \ddot{\theta}_1 + l_1^2 m_2 \ddot{\theta}_1 + l_1 l_2 m_2 \ddot{\theta}_2 = \sum F_q$$

Repeating the steps with the second Euler-Lagrange equation (8.4)

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\theta_2}}\right) = -l_1 l_2 m_2 \ddot{\theta}_1 + l_2^2 m_2 \ddot{\theta}_2 \tag{D.44}$$

$$\frac{\partial T}{\partial \theta_2} + \frac{\partial R}{\partial \dot{\theta}_2} + \frac{\partial V}{\partial \theta_2} = -g l_1 m_1 \theta_1 - g l_1 m_2 \theta_1 - k_1 \theta_1 - k_2 \theta_1 + k_2 \theta_2 - b\dot{\theta}_1 + b\dot{\theta}_2 \tag{D.45}$$

Substituting (D.44, D.45) into (8.4) yields (8.6), repeated below

$$b\dot{\theta}_1 - b\dot{\theta}_2 + g l_2 m_2 \theta_2 - k_2 \theta_1 + k_2 \theta_2 - l_1 l_2 m_2 \ddot{\theta}_1 + l_2^2 m_2 \ddot{\theta}_2 = \sum F_q$$