Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Bjørn Hartmann

Digital Twin Monitoring of Offshore Knuckle Boom Crane

Master's thesis in Mechanical Engineering
Supervisor: Bjørn Haugen, Terje Rølvåg
December 2019

NTNU
Norwegian University of
Science and Technology

NTNU

Date

Faculty of Engineering
Department of Mechanical and Industrial Engineering

MASTER 2019
FOR
STUD.TECHN. BJØRN HARTMANN

DIGITAL TWIN MONITORING OF OFFSHORE KNUCKLE BOOM CRANE
Digital Twin monitorering av offshore knekk bom kran

Several software companies are developing digital twin solutions for predictive maintenance and monitoring of structural integrity. These are based on very expensive proprietary formats and solutions not applicable to academia and SME companies. NTNU/MTP is therefore developing a cloud based monitoring system (CBMS) for integrity monitoring of physical structures and mechanisms. The CBMS is currently in a prototype phase and we want to benchmark this system on the MTP's knuckle boom crane.

Tasks will include:

1. Configure the CBMS for monitoring of the most critical knuckle boom crane failure modes and identify these(fatigue, yield, buckling, instability etc.).
2. Identify the monitoring and post-processing CBMS requirements for monitoring of critical failure modes. Implement the specifications together with the CBMS project students.
3. Optimize the crane simulation model and prepare sensor, mechanical and hydrodynamic FMUs for the CBMS.
4. Identify and implement an event trigger and crane initialization system. The objective is to avoid redundant simulations and to align the initial position of the physical and virtual crane.
5. Implement the digital twin (DT) solution (connect crane hardware with simulation software and the CBMS)
6. Setup and benchmark the DT solution

If time permits:

7. Write a scientific digital twin paper with the supervisors

<u>Contact</u>:

At the department (supervisor, co-supervisor):	Terje Rølvåg and Bjørn Haugen
From the Marine department:	Eilif Pedersen
From EDR Medeso:	Morten Haugen Blåsternes

NTNU	Hazardous activity identification process	Prepared by	Number	Date
HSE		HSE section	HMSRV2601E	09.01.2013
		Approved by		Replaces
		The Rector		01.12.2006

Unit: *Knuckle boom crane, Tyholt, Trondheim* **Date:** 14/5 -19

Line manager:

Participants in the identification process (including their function)**:**

Short description of the main activity/main process: Project thesis for student Bjørn Hartmann, STRUCTURAL DIGITAL TWIN MONITORING OF OFFSHORE KNUCKLE BOOM CRANE

Is the project work purely theoretical? (YES/NO): NO *Answer "YES" implies that supervisor is assured that no activities requiring risk assessment are involved in the work. If YES, briefly describe the activities below. The risk assessment form need not be filled out.*

Signatures: *Responsible supervisor:* [illegible] *Student:* [illegible]

ID nr.	Activity/process	Responsible person	Existing documentation	Existing safety measures	Laws, regulations etc.	Comment
1	Operation of crane	Student	N/A	N/A	N/A	
2	Instrumentation	Student	N/A	N/A	N/A	

NTNU	Risk assessment	Prepared by	Number	Date
HSE/KS		HSE section	HMSRV2603E	04.02.2011
		Approved by		Replaces
		The Rector		01.12.2006

Unit: *Knuckle boom crane, Tyholt, Trondheim* **Date:**

Line manager:

Participants in the identification process (including their function)**:**

Short description of the main activity/main process: Project thesis for student Bjørn Hartmann, STRUCTURAL DIGITAL TWIN MONITORING OF OFFSHORE KNUCKLE BOOM CRANE

Is the project work purely theoretical? (YES/NO): NO *Answer "YES" implies that supervisor is assured that no activities requiring risk assessment are involved in the work. If YES, briefly describe the activities below. The risk assessment form need not be filled out.*

Signatures: *Responsible supervisor:* *Student:*

Activity from the identification process form	**Potential undesirable incident/strain**	**Likelihood:**	**Consequence:**			**Risk Value (human)**	**Comments/status Suggested measures**
		Likelihood (1-5)	Human (A-E)	Environment (A-E)	Economy/ material (A-E)		
1 Operation of crane	Breakdown of crane	1	A1	A1	A1		
2 Instrumentation	Breakdown of crane under instrumentation	1	C1-D1	A1	A1		Serious if crane breaks down and lands on top

Abstract

Digital twin technology is forecasted to experience substantial growth. Much of the traction originates from the fourth industrial revolution of cyber-physical systems, bridging the gap between the physical and virtual worlds, a bridge that will revolutionize maintenance and product development. The definition of a digital twin is used across various disciplines such as processes, products and services. In structural mechanics, however, the definition is limited to the generation of a virtual replication of a physical structure with identical structural behavior.

This master thesis presents an end to end digital twin solution. Implemented solutions with accompanying methods and theories are described in detail. Instrumentation, inverse method, modeling, subsystem coupling, real-time simulation, cloud-based monitoring system (CBMS) and remaining useful life estimations are considered. The digital twin referred to in this thesis is a system of a barge and a crane. The crane is located at the center of Marine Technology at Tyholt, Trondheim. The physical barge was never realized during the thesis. Focus is given to the applied inverse method and the proposed co-simulation implementation.

An inverse method in structural dynamics is the identification of loads based on measured responses on a physical system. An inverse method based on finite element beam theory was applied. A co-simulation is a transient simulation of coupled subsystems. Independent dynamic simulation subsystems powered by Fedem technology were coupled through the use of the Functional Mock-up Interface (FMI), a tool-independent standard created for cross-platform model exchange. As the physical barge was never realized in the thesis, the concepts and proposed solutions for co-simulation are described, tested and validated in a virtual environment only.

Strain gauges were used to measure the responses used in the inverse method. Results show that the accuracy of the method is highly dependent on the electrical noise in the physical setup. However, for the case of satisfactory noise levels, the inverse method with its quasi-static assumption is assumed to be well formulated for the standalone crane digital twin. The co-simulation tested in the noise-free virtual environment showed promising results. The identified loads, as well as resulting strains, were satisfactory accurate. Further testing in a real-life application will be essential for further validation. Results are however promising, and a strong foundation for further development of a real-time structural digital twin co-simulation has been laid.

Sammendrag

Digital tvilling-teknologi er forventet å oppleve kraftig vekst. Mye av trekkraften springer ut i den fjerde industrielle revolusjonen ved cyber fysiske systemer som slår bro mellom den fysiske og virtuelle verden. En bro som vil revolusjonere vedlikehold og produktutvikling. Definisjonen av en digital tvilling er brukt på tvers av plattformer som prosesser, produkter og tjenester. I strukturell mekanikk er definisjonen begrenset til en virtuell gjengivelse av en fysisk struktur med identisk strukturell adferd.

Denne masteroppgaven presenterer en «ende til ende» digital tvilling-løsning. Implementerte løsninger med tilhørende metode og teori er beskrevet i detalj. Instrumentering, modellering, system-kobling, sanntidssimulering, skybasert overvåkingssystem og restlevetid estimeringer er inkludert. Den digitale tvillingen referert til i denne oppgaven er et system av en kran og en flåte. Kranen befinner seg på det marintekniske senteret ved NTNU, Trondheim. Den fysiske flåten ble aldri realisert i løpet av oppgaven. Fokuset er på den anvendte inversmetoden og den foreslåtte co-simuleringsimplementeringen. En inversmetode i strukturell dynamikk er metoden for å identifisere laster basert på målte responser på et fysisk system. En inversmetode basert på elementmetoden ble anvendt. En co-simulering er en transient simulering av koblete subsystemer. Uavhengige dynamiske simuleringssubsystemer fra Fedem Technology ble sammenkoblet gjennom Functional Mock-up Interface (FMI), en verktøyuavhengig standard for modellutveksling mellom plattformer. Siden den fysiske flåten aldri realiserte seg i løpet av oppgaven er konseptene og løsningene for co-simulering beskrevet testet og validert kun i virtuelle omgivelser.

Strekklapper ble brukt til å måle responser til bruk i inversmetoden. Resultatene viste at nøyaktigheten på metoden var høyst avhengig av mengden elektrisk støy på det fysiske oppsettet. Likevel, dersom støynivåene var lave nok, er inversmetoden med sine quasi-statiske antagelser konkludert som velformulert for tilfellet ved en digital tvilling av en kran alene. Co-simuleringen testet i de støyfrie virtuelle omgivelsene viste lovende resultater. De identifiserte lastene, samt resulterende tøyninger var tilfredsstillende nøyaktige. Videre testing i fysiske omgivelser er avgjørende for videre validering. Resultater er imidlertid lovende, og et sterkt grunnlag for videre utvikling av en sanntids digital tvilling med co-simulering har blitt lagt.

Preface

This Master thesis is a part of the M. Sc. Mechanical engineering degree at the Norwegian University of Science and Technology (NTNU). The work was carried out in the fall of 2019.

The complete creation of a structural digital twin of a knuckle boom crane-barge system is described throughout the thesis. The physical crane is located at the center of Marine Technology in Trondheim, Norway. Theory and application of chosen methods are described in detail, in the hope that it will be of interest and value for further work. All work has been executed by the author of the thesis, Bjørn Hartmann.

December 24th, 2019

[illegible]

Bjørn Hartmann

Acknowledgements

First and foremost, I would like to thank Professors Terje Rølvåg and Bjørn Haugen for continuous guidance throughout the thesis. Both have always taken the time to discuss problems and willingly shared their expert knowledge with me. This has been crucial to the progress of the thesis, and provided important knowledge, inspiration and insight.

I would also like to thank EDR Medeso, and their highly competent employees. I would like to thank Morten Haugen Blåsternes for the opportunity to work with EDR Medeso and sit at their offices at Lysaker for periods of the Master thesis. They have provided guidance and positive and constructive feedback throughout the thesis.

Runar H. Refsnæs has put his work aside and given assistance whenever Fedem-FMU issues have arisen, for that I am thankful. The implementation of the co-simulation described throughout the thesis would not have been possible without the help from Runar.

Bjørn Hartmann
December 24th, 2019
Trondheim, Norway

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction to the thesis

This thesis is part of the M. Sc. Mechanical engineering degree at the Norwegian University of Science and Technology. The main object of the thesis is to present and apply a method for the development of a complete digital twin system simulation using generated Fedem FMU components.

A digital twin is a structurally identical finite element model of a physical structure, capable of mirroring the physical models every movement. This mirroring is accomplished through an inverse method. The inverse method term in structural dynamics is the identification of loads based on measured responses on a physical system. The ability to monitor an asset through its digital twin will completely revolutionize the approach to essential elements of the product life-cycle. Maintenance can be scheduled based on operation rather than conservative worst-case scenarios. Abuse of assets can be reduced to a minimum as alerts can be given whenever an asset is falsely handled or loaded, leading to the elimination of sudden breakdowns and reduction of fatigue wear. The common practice with design life based on conservative safety factors leads to over-engineering, creating a more massive and costly asset than required. Also, a working digital twin solution gives a distinct marked advantage and promotes sales based on the previous arguments.

The above arguments are solid, and one may wonder why there is a single asset out there not monitored through its digital twin. Having a digital twin is, however, dependent on accurate load replication of the inverse method. The solution to the inverse problem is non-trivial, and generic state of the art methods largely remains to be found.

The end to end solution of a digital twin for monitoring a knuckle boom crane mounted to a barge is explained throughout the thesis. This system is intended for offshore windmill maintenance operations. Instrumentation, inverse method, modeling, subsystem coupling, real-time simulation, cloud-based monitoring system (CBMS), and remaining useful life estimations are considered.

The physical barge for the crane-barge system was never realized in the thesis. Therefore, digital twin testing and validation of the system is based on a virtual environment. The single crane subsystem is tested in real-life applications.

1.2 Objectives

The main objectives are listed below.

- Configure the CBMS for monitoring of the most critical knuckle boom crane failure modes and identify these(fatigue, yield, buckling, instability, etc.).
- Identify the monitoring and post-processing CBMS requirements for monitoring of critical failure modes.
- Evaluate and benchmark FMUs for co-simulation
- Identify and implement an event trigger and crane initialization system. The objective is to avoid redundant simulations and to align the initial position of the physical and virtual crane.
- Implement the digital twin (DT) solution (connect crane hardware with simulation software and the CBMS)
- Setup and benchmark the DT solution

1.3 Structure of the Report

To ease navigation, the number of chapters in the report is kept to a minimum. Chapters mainly contain the following.

Chapter 2 gives an introduction to selected previous works as well as to some important concepts and software used for the development of the digital twin.

Chapter 3 discusses and explains applied theory in detail to provide a theoretical background for Chapter 4.

Chapter 4 presents the application of explained theory and further methods applied to create the digital twin.

Chapter 5 illustrates and discusses findings and results from the previous chapters.

Chapter 6 is dedicated to concluding remarks

Chapter 7 suggests further work related to the thesis and its findings.

Appendix presents the essential code used in the thesis. Code examples are given piecewise for easier navigation. The full implementation code to the digital twin can be given upon request.

Chapter 2

Literary survey

The global market for digital twins is expected to multiply in the coming years. Market research has shown that by 2023, the "Digital Twin Market" will reach a value of %16 billion USD, growing at an annual rate of 25% LLC (2017). Definitions of digital twins, however, are found across various platforms as the internet of things develop. The term is used to describe virtual models of processes, products, and services, all with the ability to analyze and monitor a system. The digital twin discussed in this thesis is limited to the monitoring of structural behavior and integrity of a physical model, often referred to as the performance digital twin Siemens (2019).

2.1 Structural digital twins

The creation of a structural twin requires data from the real model to be passed to the simulated twin; data that can represent properties of the physical model such as frequency of motion, local strain, or provided input current. Recorded data is required to be sufficient to reproduce the structural behavior through simulation. To this, there are several approaches.

The master thesis of Christiansen, Erlend (Christiansen (2018)) and the project and master thesis of Moi, Torbjrn (Moi (2018) and Moi (2019)) written in 2018 and 2019 are of high relevance to the work done in this thesis. These both discuss the use of cranes for digital twins, instrumented with strain gauges. The two theses differ in the applied inverse method.

In the case of Christiansen (2018), Christiansen uses recorded bending strain to detect the working payload in the vertical direction. The crane is here simplified to a 2D truss-system, and trigonometric and geometric identities are used to find areas of maximum stress for mounting of the strain gauges. This method does not take into account the possible side-way forces caused by the operating crane, such as a swinging pendulum.

The project thesis of Moi (Moi (2018)) is focused on the development of a general component inverse method based on finite element beam theory, taking into account both the side-way forces and compensating for induced strain from rotation. This method is, however, not flawless, as it is a static approach to a dynamic system, meaning that all nonlinear and dynamic effects are neglected. However, for the case of low-frequency loads and relatively slow body motion, it is considered a good approach. The basic idea behind this inverse method is applied to the digital twin created in this thesis. In Torbjørn's master thesis, several modifications are suggested to increase the robustness of the method. The interpolation method suggested was inspirational for the interpolation described in Section 3.7. The theory of the method is discussed further in Section 3.7.

2.1.1 Project Thesis

The crane in this thesis was subject to the author's project thesis in the spring of 2019. The main outtake was the discovered sensitivity of the inverse method to noise in the recorded strain data. Even under static no-loading conditions, the force estimated on the outer end crane arm by the inverse method could be in the range of hundreds of Newtons as a result of an overly stiff stiffness matrix paired with too large noise. These results were unjustifiable for structurally accurate digital twin monitoring. This was motivation for further development of the method.

2.2 The Crane

The knuckle boom crane referred to as "the crane" throughout this thesis is a scaled-down crane model intended for lab use. It is situated at the center of Marine Technology in Trondheim, Norway. The crane and its design was developed in the Master Thesis by Gyberg, Fredrik, in the fall of 2017 ,Gyberg (2017). Before this thesis, the crane has been subject to one project thesis, Hartmann (2019), and one master thesis, Johansen (2019).

2.3 FMU/FMI

The components of a complete system simulation are often developed using different domain specialized software. The system integrator must be able to cope with simulation and modeling environments from different suppliers. The Functional Mock-up Interface (FMI) poses a solution to the above problem. FMI is an independent tool standard that defines an interface for cross-platform model exchange and co-simulation FMI-standard (2019a). The FMI standard does not specify in what way the FMUs are to be interconnected and co-simulated, meaning that it can, in theory, be implemented to any software. There are currently over a hundred softwares supporting the FMI standard, and the number is rapidly growing, fmi standard.org (2019). Functional Mock-up Unit (FMU) is an executable simulation model that implements the FMI standard through the two flavors Co-simulation (CS) and model exchange (ME). The names of the two do, however, not properly represent their applications. CS and ME can both be used for model exchange and co-simulation of multiple models. The difference lies in the way the FMU models are stepped forward in time. When simulating a CS FMU, the solver of the FMU model is supplied by the exporting tool. In the case of a ME FMU, the importing tool solves the FMU model directly, using numerical equation supplied by the exporting tool FMI-standard (2019b).

Both the interfaces model exchange and co-simulation FMUs share the zipped .fmu extension and that computations are evaluated based on the C programming language. All FMU's have a generated XML file containing the associated variables. Inputs and outputs to an FMU are predefined single dimension variables. Once an FMU is created, the model code is typically stored in a compiled binary format, and thus there is no simple way of making modifications to the internal model. The pros and cons of this can be argued, and it does not favor scalability nor debugging, but this black box implementation is encapsulating sensitive solver and model info to allow the exchange of models and enable cooperation on system models across companies.

FMU-handling

The python based application programming interface(API) FMPy was used to handle FMU operations throughout the thesis. All FMU operations are performed using this API. FMPy is a free, open-source FMU-API with some documentation online FMPy (2019b). It supports both FMI 1.0 and 2.0 as well as co-simulation and model exchange. It is easy to use and can be "git"-installed FMPy (2019). The use of the API is further described in Section 3.7.8.

2.3.1 Co-Simulation

The use of co-simulation enables the interconnection of two FMU instances created in their respective domain-specific tools. Co-simulation is a transient simulation of coupled subsystems, where the different subsystems are computed and stepped forward in time independently from each other. The exchange of data between the coupled model is restricted to discrete communication points; at each time step. Each subsystem is free to use its embedded solver and local time step. The FMI standard for co-simulation is based on the so-called master/slave framework. The subsystems are aligned in time by a co-simulation master. The subsystems are referred to as slaves.

Co-Simulation on Marine Structures

Co-simulation of specific subsystems may cause numerical challenges, in particular, those assumed to be tightly or strongly coupled Thule et al. (2017). The independent subsystem property leads to a trade-off between numerical stability and computational weight, which is discussed further in 3.1. The extensive work of Skjong, Skjong (2017), shines light upon these challenges, with specific attention to the marine system of a crane and a ship hull. The following is extracted directly from co-simulation research on marine systems, and concludes on a somewhat discouraging note;

"Consider, for example, the rigid mechanical connection between a vessel's hull and crane. Ideally, such a connection calls for solving the hull–crane system as one, and the straightforward (explicit) co-simulation of both as separate subsimulators with separate solvers is unfeasible. In other words, it may be best to simply refrain from splitting tightly-coupled systems for co-simulation altogether." Sadjina et al. (2019)

2.4 Software

2.4.1 Fedem

Fedem is short for Finite Element Dynamics in Elastic Mechanisms and is a software for running nonlinear dynamics simulations of mechanical bodies. The FMU used in this thesis is exported from Fedem. Further introduction to applied theory can be found in Section 3.2.

2.4.2 Ansys Twin Builder

Ansys Twin Builder(ATB) is a new state of the art complete system simulation software, released in 2018. It offers a multi-domain workflow specifically designed for the creation of digital twins. ATB is a part of the Ansys Electronics Desktop, and is built on a combination of previous software such as Ansys Simplorer and Ansys SCADE Engineering.com (2019).

Twin Model

The Ansys specific Twin file extension is mentioned as it has very similar capabilities as those of an FMU. While not used in the digital twin application, the generation of a Twin file is included for future use with Fedem FMUs. Ansys provides a specific software development kit (SDK) for handling the ATB-generated .twin files.

Chapter 3

Theory

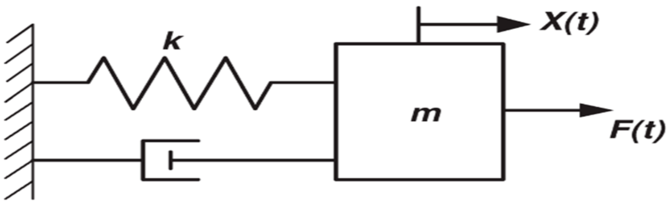
3.1 Co-Simulation

The decoupling of each subsystem between time-steps, as mentioned in 2.3.1, leads to each subsystem being a discrete dynamical system. These sub-simulators of the system, throughout this thesis in the form of FMUs, are validated to reach convergence and be numerically stable if treated as a single system. The co-simulation of the energetically coupled subsystems is, however, only considered to be conditionally stable. The overall stability depends on the global macro-step time. Further, the stability of the system is highly dependent on the decomposition/coupling technique Viel and Minimes (2014). This can be illustrated by interconnecting two simple spring camper systems, see Figure 3.1.

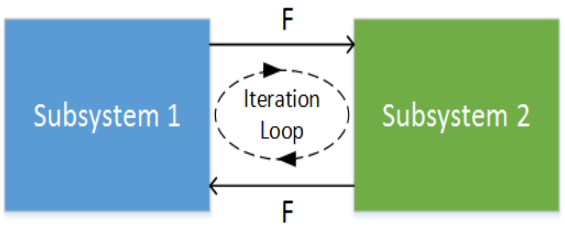
In this force-force connectivity configuration, the two sub-models have forces as both input and output. Further, assume one of the masses is given a forced deflection and is then released. This force-force loop outside the sub-model environments essentially requires iterations between the sub-models to retain stability, as the forces are required to be equal.

By altering the co-simulation decomposition method to the use of a force-displacement or force-velocity coupling, one avoids the constrain of the force-force coupling. The doctoral thesis of Skjong, Skjong (2017), among other work, shows there is still an entire field of numerical relationships to consider before an unconditionally stable solution method can be found. However, for this thesis and the specific case of a crane and barge, the hypothesis has been that a force-displacement pairing would provide satisfying results. The possible excitation of high-frequency waves to the barge may again induce a similar problem as the fore-force connection case, in that the system again becomes what Skjong calls a system coupled through frequencies. The numerics of this is beyond the scope of the thesis.

In this thesis, no global error estimation method has been implemented. In Sadjina et al. (2019), the co-simulation phenomena of incorrectly transferred energy between two cou-



(a) Mass spring damper subsystem



(b) Co-simulation of two mass spring damper systems

Figure 3.1: Illustration of the iterative force-force connectivity of two mass spring damper systems

pled simulators is discussed. Energy is either created or destroyed during each global time-step. As a result, the total dynamic energy in the system is changed, which can result in varying degree of simulation accuracy. Implementing an error estimation method remains as further work. Figure is taken from Sadjina et al. (2019) and illustrates the energy distorted or emerged during co-simulaton. The numerics of this has not been studied further.

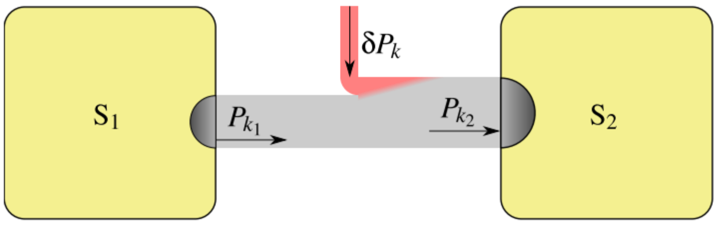


Figure 3.2: Residual power δP_k when simulation subsystems S_1 and S_2, figure from Sadjina et al. (2019)

3.2 FEDEM (FMU) Dynamic Solver

The structural monitoring of a physical asset in real-time requires real-time simulation as the asset operates. One of the main challenges with this is computational speed, due to computational complexity in solving nonlinear dynamic systems. A popular solution to this issue is the use of reduction methods. These are methods resulting in models that are computationally lightweight while still preserving the physical behavior of the original model.

In Fedem, the reduction of the model is accomplished by a combination of static and fixed interface normal modes (Section 3.2.2) followed by solving the equations of motion(Section 3.2.1).

3.2.1 Equations of Motion

The dynamic equation of motion at time k seen in Equation (3.1) is solved on incremental form. $\mathbf{F}^I$ represents inertia forces, $\mathbf{F}^D$ damping forces, $\mathbf{F}^S$ internal elastic or spring forces and $\mathbf{Q}$ the input forces such as gravitation and applied loads, all functions of time, position, velocity and acceleration.

$$\mathbf{F}_k{}^I + \mathbf{F}_k{}^D + \mathbf{F}_k{}^S = \mathbf{Q}_k \tag{3.1}$$

The subscript k denotes the equation at time k. For time k+1 the above equation can be written as:

$$\Delta\mathbf{F}_k{}^I + \Delta\mathbf{F}_k{}^D + \Delta\mathbf{F}_k{}^S = \Delta\mathbf{Q}_k \tag{3.2}$$

Rewriting each element of Equation (3.2) to Equation (3.3) gives the linear dynamic equation of motion for the system, where $\mathbf{M}_{IK}$, $\mathbf{C}_{IK}$ and $\mathbf{K}_{IK}$ are the mass, damping and stiffness matrices of the system respectively. $\Delta\mathbf{r}$ is the the change in position during the time increment.

$$\mathbf{M}_{IK}\Delta\ddot{\boldsymbol{r}} + \mathbf{C}_{IK}\Delta\dot{\boldsymbol{r}} + \mathbf{K}_{IK}\Delta\boldsymbol{r} = \Delta\mathbf{Q}_k \tag{3.3}$$

The above equation is used for each increment by a predictor step followed by a corrector step using newton-raphson iterations. All theory is extracted from the FEDEM theory user guide Technology (2018).

3.2.2 Component Mode Synthesis

All simulations carried out by the use of Fedem applies a model order reduction method known as Component Mode Synthesis (CMS). CMS is a dynamic condensation method combining static (Guyan reduction) and fixed-interface modes, resulting in a significantly reduced number of degrees of freedom (DOFs) for the system.

Dependent on the mesh and complexity of a model to be simulated, the model can consist of a large number of nodes and hence a large number of DOFs. CMS divides these nodes into internal and external nodes for every individual component or part in the system. The external nodes are referred to as master-nodes and the internal as slave-nodes. In Fedem, the master nodes are called "triads" and are defined by selection in Fedem.

For the static case, neglecting dynamic mass and damping contributions, and if all applied forces are acting on the external nodes, the system of the substructure can be written as

$$\begin{bmatrix} \mathbf{K}_{ii} & \mathbf{K}_{ie} \\ \mathbf{K}_{ei} & \mathbf{K}_{ee} \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \mathbf{v}_e \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_i \\ \mathbf{Q}_e \end{bmatrix} \tag{3.4}$$

Where the $\mathbf{v}_i$ and $\mathbf{v}_e$ are the internal and external displacement vectors respectively, and **K** and **Q** are the stiffness and load matrix and vector for both internal ($_i$) and external ($_e$) degrees of freedom. From the first equation in the system of equations in Equation 3.4, we can express the internal displacement in each component as a function of its external DOFs. Since no external forces are allowed to act on the internal DOFs, we have

$$\mathbf{v}_i = -\mathbf{K}_{ii}^{-1}\mathbf{K}_{ie}\mathbf{v}_e = \mathbf{B}\mathbf{v}_e \tag{3.5}$$

where **B** often is referred to as the influence matrix, and represent the static modes of the component. The elements of the influence matrix can be found by applying unit loads, in turn, to all the external DOFs of the component, while setting all other external DOFs to zero. These static modes of the components are exact, in the sense that apart from round off errors, solving the complete system of parts with boundary conditions would produce the same result.

These static modes do, however, not consider any dynamic effects, and may give inaccurate results for dynamic simulations where inertia and damping effects are prominent. This is accounted for by the fixed-interface modes. These fixed-interface modes are often referred to as "Craig-Bampton modes" and are found by fixing the external DOFs of the substructure and look at the eigenmodes for this constrained configuration. The free vibration of this undamped system can be expressed as Equation (3.6), which for a harmonic motion is resulting in the eigenvalue problem in Equation (3.7) where ϕ is an eigenvector of the system.

$$\mathbf{M}_{ii}\ddot{\mathbf{v}}_i^{\,i} + \mathbf{K}_{ii}\dot{\mathbf{v}}_i^{i} = \mathbf{0} \tag{3.6}$$

$$(\mathbf{K}_{ii} - \omega^2\mathbf{M}_{ii})\phi = \mathbf{0} \tag{3.7}$$

Solving the system eigenvalue problem for a selected number of eigenvalues and vectors gives us the expression for the displacements of the internal degrees of freedom as a combination of eigenvector amplitudes **y** as

$$\mathbf{v}_i = \sum_{n=1}^{k} \phi_j y_j = \Phi\mathbf{y} \tag{3.8}$$

for the selected k eigenvectors. $\mathbf{\Phi}$ is the eigenvector matrix.

The full dynamic system in Equation (3.9) can now be reduced to Equation (3.10) for the super-element displacements by expressing the system by the external degrees of freedom $\mathbf{v}_e$ and the mode shape amplitudes $\mathbf{y}$.

$$\mathbf{M}\ddot{\mathbf{v}} + \mathbf{C}\dot{v} + \mathbf{K}\mathbf{v} = \mathbf{Q} \tag{3.9}$$

$$\mathbf{v} = \begin{bmatrix} \mathbf{v}_e \\ \mathbf{v}_i \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{B} & \mathbf{\Phi} \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \mathbf{y} \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{v}_e \\ \mathbf{y} \end{bmatrix} \tag{3.10}$$

Fedem simulation can be performed in real-time, recovering strains and stresses at selected nodes in the model. This is rather impressive given the complexity of the model. All nodes are, however, not solved at all times, as the simulation would not be able to keep up with real-time. The reduction method, however, expresses all internal degrees of freedom at the nodes as a function of the external degrees of freedom. Selected nodes can therefore be extracted during simulation with little computational weight.

3.2.3 Viscous Rayleigh damping

All structures experience loss in energy during motion. Damping is the absorption of mechanical energy from the system in motion, mostly due to the conversion of potential energy to heat.

The damping matrix **C** in Equation (3.3), is not computed in its full in Fedem. The assumption that there is a linear relationship between damping, mass and stiffness of a system expressed as in Equation (3.13) is instead used, this form of damping is often referred to as proportional or Rayleigh damping, which is a type of viscous structural damping for dynamic systems.

$$\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K} \tag{3.11}$$

α and β are scalars with units 1/s and s, respectively. The damping ratio for a given natural frequency ω_i can be shown to be given by

$$\zeta_i = \frac{1}{2}(\frac{\alpha}{\omega_i} + \beta\omega_i) \tag{3.12}$$

Where ζ_i is the damping ratio for the selected frequency. By choosing two frequencies, rearrange and substitute, one can find the relation

$$\begin{aligned} \alpha &= \frac{2\omega_1\omega_2}{\omega_2^2 - \omega_1^2}(\zeta_1\omega_2\zeta_2\omega_1) \\ \beta &= \frac{2}{\omega_2^2 - \omega_1^2}(\omega_2\zeta_2 - \omega_1\zeta_1) \end{aligned} \tag{3.13}$$

From Equation 3.13, one can see that the Rayleigh damping coefficients are dependent on two natural frequencies and the damping ratio for that frequency. The two damping ratios can be found experimentally employing a "hammer test" explained further in 3.3.

The rayleigh damping coefficients give correct damping for the two measured frequencies. It does not give any assurance correct damping ratios for other frequencies. Still, as the lowest natural frequencies are most likely to oscillate under low-frequency loads, the viscous damping applied is believed to be sufficient. The mass proportional damping term in rayleigh damping is due to drag, and it must be applied with caution, as it may often provide unrealistically large drag effects on low frequencies. α is therefore set to zero.

3.2.4 Limitations

It should be mentioned that due to the reduction technique used, based on eigenmodes and static modes, Fedem cannot include nonlinear material or contact effects. The nonlinear geometric effects from large rotations and displacements are, however, included through what is known as a co-rotation formulation. This formulation essentially subtracts the rigid body motions from the displacements to obtain the local deformations Haugen (2017).

3.3 "Hammer test"

As mentioned in the previous section, the damping ratios leading to the coefficients α and β can be determined by an experimental "hammer test".

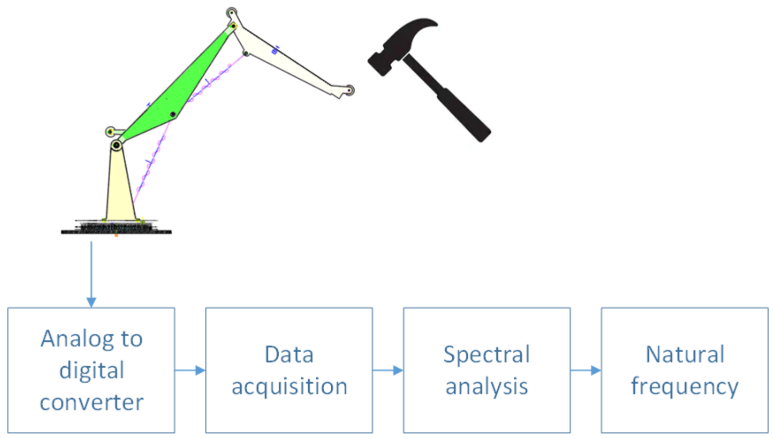


Figure 3.3: Workflow hammer-test

Subjecting a structure to a pulse load by the hit of a hammer causes the structure to oscillate. The most prominent oscillation frequencies after impact are at the resonance or natural frequencies of the structure. An accelerometer or gyroscope can measure these oscillations.

The natural frequencies of oscillations can be found by performing a fast Fourier transform (FFT) on the recorded data, converting the data from the time domain to frequency components. By performing a band-pass filter operation to the recorded accelerometer data, one can inspect the decreasing rate of the amplitude of the free damped vibration at the resonance frequency. This decreasing rate is known as the logarithmic decrement and can be expressed by

$$\delta = \frac{1}{n} ln(\frac{y(t)}{y(t+nT)}) \tag{3.14}$$

where y(t) is the amplitude at time t and y(t+nT) is the decreased amplitude n periods later, visualized in Figure 3.4. Fortunately, there is a relation between the damping ratio ζ and the logarithmic decrement δ given as in Equation (3.15).

$$\zeta = \frac{1}{\sqrt{1+(\frac{2\pi}{\delta})^2}} \tag{3.15}$$

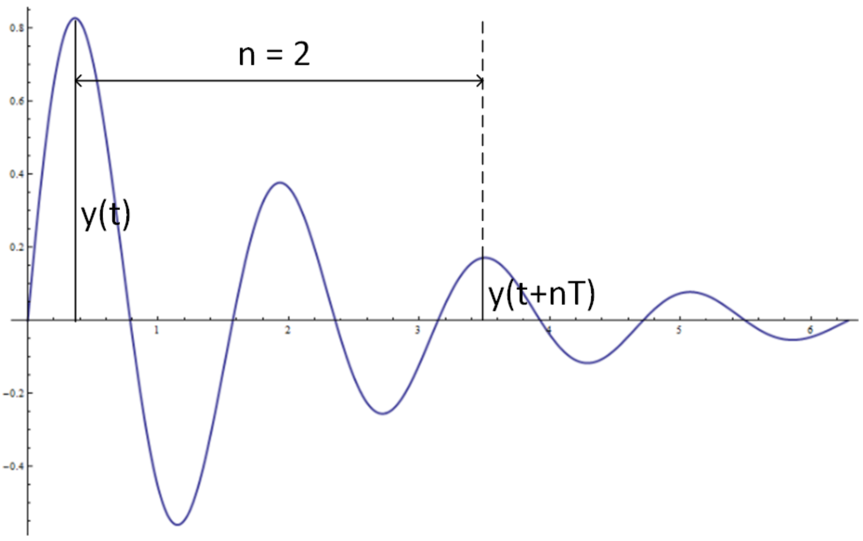


Figure 3.4: Illustration of the parameters of Equation (3.14)

Knowing both natural frequencies ω_i and ζ_i, the rayleigh damping coefficients can be found. The results from the hammer-test performed in this thesis can be seen in Section 5.1.

3.4 Hydrodynamics

While one can simulate both crane and barge as one model inside Fedem, a considerable amount of time has been dedicated to the modeling of a hydrodynamic component in another software than Fedem. This was attempted to show the core benefit of using the FMI standard, namely cross-platform co-simulation.

A model was made and simulated in Ansys Aqwa. Still, no means of exporting the model to an FMU with the ability to change parameters during the simulation was found, despite direct ongoing interaction with employees at the Ansys development team in Belgium.

The use of the Matlab/Simulink based Marine System Simulator (MSS) toolbox Perez et al. (2006), was also inspected. MSS is an open-source library for solving hydrodynamic equations of motion for marine system models. The models are generated in the hydrodynamic software Wamit or Ship X. While using MSS is a possible solution, the process of modeling, exporting, and successfully implementing an FMU barge component proved to be rather tricky. The procedure wass eventually deemed too time-consuming. The MSS toolbox has some existing examples of larger ship models. However, there would not have been a way to validate the results of simulating a Fedem crane FMU with an MSS toolbox ship example model.

The only hydrodynamic component used for co-simulation was therefore created in Fedem and exported as an FMU for co-simulation.

3.4.1 Wave Modelling

During the thesis, the system model was exposed to both regular and irregular waves. In an irregular sea state, waves are modeled through super-positioning several regular waves, as shown in Figure 3.5. This creates a more realistic realization of the offshore ocean surface, as opposed to a regular wave. The validation of the inverse method by the use of Fedem fabricated data was challenged by using irregular waves in the form of a JONSWAP spectrum, defined by a significant wave height H_s and a spectral peak period T_p. Such a spectrum is of relevance as a likely future application of the digital twin is in the north sea.

When regular waves are used to model an offshore sea state, a given H_{max} is used in place of H_s, which is a commonly applied method to find results of interest. For the case where an ideal regular wave is to be applied in one of the Tyholt towing tanks, H_{max} is chosen as the actual wave amplitude. The alignment of a physical and virtual twin for a regular wave based on the phase, frequency and amplitude has been implemented and is described in Section 4.8.

Although no method of aligning the digital and physical models in irregular waves has been implemented, showcasing the fact that the inverse method seems to be working well also for these waves may be of later interest for the development of irregular wave alignment methods. This is discussed further in 5.

3.4.2 Fedem Hydrodynamics

The additional loads on a Fedem model from the inclusion of an ocean environment are added to the simulation as external forces. Buoyancy, drag and added mass forces are

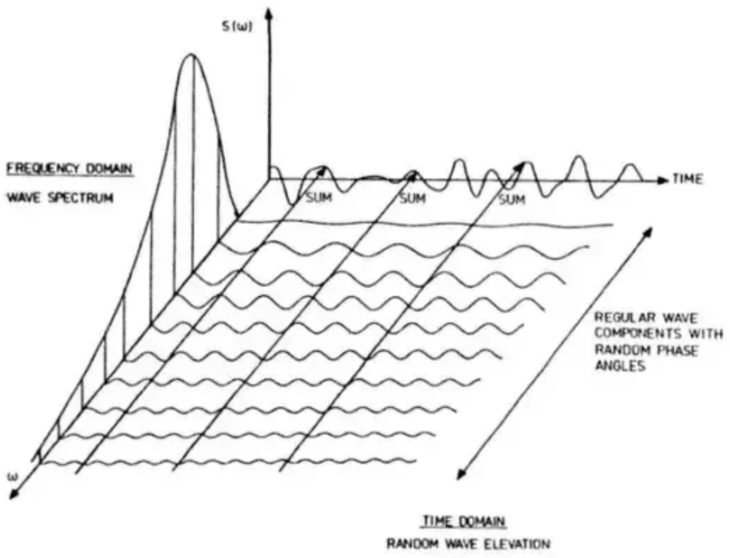


Figure 3.5: Figure illustrating the connection between a frequency domain and a time domain representation of waves in a long-crested short term sea state, figure from Faltinsen (1993)

computed for all elements included in the hydrodynamic simulation. The hydrodynamic calculations in Fedem are limited to Morison-elements. The potential flow based Morison equation is used to calculate the wave force per unit length on a semi-submerged or submerged cylinder, limiting Fedem hydrodynamics to beam elements. The integration over the total cylinder yields a total wave force. The mass and drag components in the Morison equation are dependent on mass and drag coefficients, respectively. These coefficients are empirically determined and dependent on several parameters, such as the Reynolds number, Keulegan-Carpenter number and surface roughness ratio Faltinsen (1993).

The primary advantage of potential flow methods over the computational fluid dynamic (CFD) methods based on the Navier-Stokes equations in dynamic simulation, is the computational numerical weight. Potential flow methods have unknowns on the surface of the immersed body only and do not require a 3D mesh throughout the region of flow, as CFD methods do. The Morison equation giving a force dF per unit length on a strip of the cylinder can be seen in Equation (3.16). Here, u and a_1 are horizontal undisturbed fluid velocity and acceleration on the midpoint of the strip, ρ is the water density, D cylinder diameter, and $C_M$and C_D the experimentally determined mass and drag coefficients. Further explanation of applied solver theory on hydrodynamics is deemed to be beyond the scope of this thesis.

$$dF = \rho\pi\frac{D^2}{4}C_M a_1 + \frac{\rho}{2}C_D D|u|u \tag{3.16}$$

When numerically analyzing offshore structures, it is essential to classify the structure in terms of what hydrodynamic forces are dominating as different solver theories have different regions of validity. Figure 3.6 shows the relationship between wavelength, wave height and structure parameters resulting in corresponding dominating hydrodynamic forces and hence the relative importance of mass, viscous drag and diffraction forces on the structure in question. When the structure is large compared to the wavelength, Morison's theory is no longer valid Faltinsen (1993). As seen in Figure 3.6, when the ratio of wavelength λ to cylinder diameter D falls below 5, the diffraction forces which are part of the hydrodynamic excitation forces, become too large to be neglected. Morison's equation does not consider diffraction forces and is not considered valid in this region.

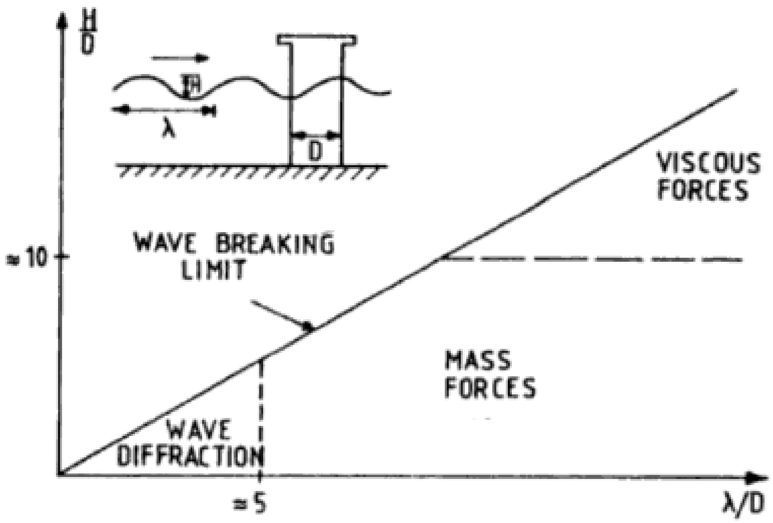


Figure 3.6: Relative importance of mass, viscous drag and diffraction forces on marine structures, figure from Faltinsen (1993)

The barge used in this thesis is to be deployed to one of the towing tanks at the Marine Center, Tyholt, for testing. In Gyberg (2017), where the physical crane was designed and developed, the design sea conditions were specified as seen in Table 3.1.

Sea conditions	
H_s	0.2 m
H_{max}	0.4 m
T_p	2.8 s

Table 3.1: Design sea state parameters, values from Gyberg (2017)

As the barge has not yet been built, its geometry is assumed to be as modeled. To classify the model, the operating wavelength needs to be determined. For an assumed infinite water depth, the wavelength λ can be expressed as

$$\omega^2 = g\frac{2\pi}{\lambda} \tag{3.17}$$

where ω is the wave frequency, g gravity, and λ the wavelength. Solving for λ under the design conditions, we get a wavelength λ above 10m, which divided by the cross-sectional diameter of the barge floating elements of 1.5m is well above the threshold of 5, and the Morison equation remains valid. It can also be noted from Figure 3.6 that under these configurations, the mass forces are dominating. The periods of the towing tanks at Tyholt can be adjusted to render the Morison equation invalid, so care must be taken when doing physical tests.

3.5 FEDEM FMU Generation

As described in Section 2.3, the use of an FMU enables the cross-software application of the crane model. In FEDEM, the generation of an FMU is at the time of this thesis not embedded in the graphical user interface. The construction of a FEDEM co-simulation FMU is done in a few steps:

Step 1: All inputs to the FMU have to be defined as "external function"-functions, and all outputs have to be defined as "1:1"-functions inside FEDEM. The "stop-time" in the dynamic solver settings needs to be disabled, and the model must be solved for "prepare for batch execution".

Step 2: FEDEM provides a folder with the files needed for creating an FMU from a prepared FEDEM model. Firstly, a .yml file is generated, specifying the name of the FEDEM model. FEDEM gives the template of the file.

Step 3: A python "build-file" is then run in the command line by the following command: python buildFMU.py -c "PATH-TO-YML-FILE" -o "PATH-TO-SAVE-FMU-TO". The command must be run from the FMUGenerator folder within the provided R7_3 folder, as buildFMU.py looks for .dll files in the R7_3 folder, one folder up from the "build-file".

Step 4: The .fmu component is stored in the specified folder ready for use after a completed run of the "build-file". This is a co-simulation type FMU.

Special case with external wave function

The external wave function discussed in Section 4.8 adds an additional step to the FMU generation process. The external wave functionality requires an external dynamic linked library(.dll) provided by Fedem, to be used inside Fedem. As per the buildFMU.py file used in the thesis, this is not accounted for. Hence, the .dll file is not included as a dependency. After FMU-generation, the zipped FMU must, therefore, be extracted, and the .dll file must be put in a specific "plugins" folder before re-zipping the folder to an FMU.

The exported Fedem FMU is a co-simulation FMU and the complete solver is included in the FMU. To run a Fedem FMU, the user must have a valid Fedem license.

3.6 Twin Export

Ansys provides the possibility to export a system as a Twin Model. This file format has the extension .twin and is today very similar to an FMU. Ansys aims to implement further functions to this extension and, with time, have a licensed file-extension, as opposed to the open-source FMU.

The generation of a Twin Model can be done from the Ansys Twin Builder Interface. This step by step procedure is included in the thesis due to its nontrivial nature and the hope of future compatibility with a complete system simulation containing one or several Fedem FMUs.

Step 1: Import the FMUs, ROMs, Modelica blocks, or other components you want to simulate and interconnect them. The input/output pins not connected should be the ones you still want available after compiling the system simulation as a .twin file.

Step 2: For all pins not connected, draw "dummy-lines" from the empty pins to well outside of the visualization of the system simulation.

Step 3: Go to Add design and choose to add a Subcircuit from a selection. Select the whole system you have interconnected, but make sure that all "dummy-lines" are extended so that the manual selection will "cut" the lines, leaving a part of the line outside of the selection.

Step 4: A new Subcircuit is now created. Embedded in the subsection is your system model, and the "dummy-lines" are now the new inputs and outputs to the new Subcircuit. The input and outputs of the new Subcircuit are given standard names and should be renamed for later convenience.

Step 5: Right-click the new Subcircuit and choose "compile as twin model".

Step 6: The new Subcircuit will appear as a model in the project manager and can be exported as a .twin model or an FMU.

3.7 Inverse method

Load identification based on measured data is referred to as an inverse problem. The approach to identifying these loads is known as an inverse method. The essence of the theoretical basis of the inverse method described in detail in Moi (2018) is shortly presented in this section, followed by the modifications made to increase the accuracy of predicted forces and the overall robustness of the method. The final equation for estimating the applied loads can be seen in Equation (3.28).

3.7.1 Theoretic Basis

The inverse method's general idea is to reduce a structure to a cantilever beam. In the author's project thesis, only the upper arm of the crane was treated as a cantilever beam. For reasons explained in the next subsection, a multi-component inverse method has been implemented.

Cantilever Beam Stiffness Matrix

The familiar relationship between stiffness, force and displacement for a cantilever beam in the linear region can be expressed as equation (3.18), where **K** is the stiffness matrix, **u** displacements and **F** the force vector, all in matrix/vector form.

$$\mathbf{K} \times \mathbf{u} = \mathbf{F} \tag{3.18}$$

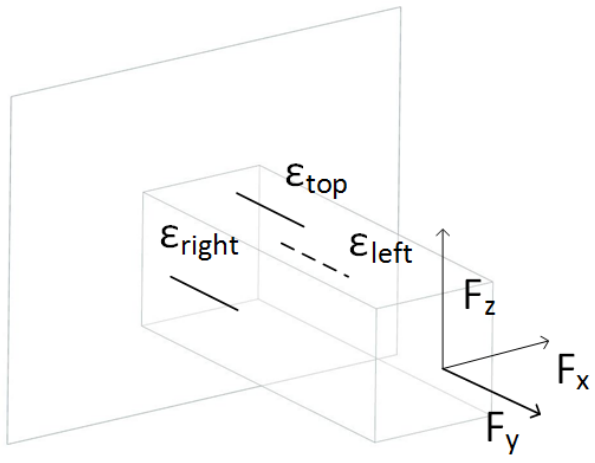


Figure 3.7: An ideal cantilever beam instrumented with strain gauges

If a cantilever beam is instrumented with three strain gauges as in Figure 3.7, Equation (3.18) can be expressed through the strain gauge values as in Equation (3.20). Where ε is the vector in equation (3.19) and **S** is a 3×3 strain-stiffness matrix giving the relationship between strains and forces.

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{top} \\ \varepsilon_{left} \\ \varepsilon_{right} \end{bmatrix} \tag{3.19}$$

$$\mathbf{S} \times \boldsymbol{\varepsilon} = \mathbf{F} \tag{3.20}$$

$$\boldsymbol{\varepsilon} = \mathbf{S}^{-1} \times \mathbf{F} \tag{3.21}$$

Applying unit loads in each axis direction in turn to Equation (3.21) results in the inverted strain-stiffness matrix $\mathbf{S}^{-1}$. Assuming this matrix is invertible, inverting gives the strain-stiffness compliance matrix used in the inverse method for calculating applied forces from strains, see Equation (3.20). The process is applicable to any equal number of strains and forces, given a system of linearly independent equations. Below is an example of a unit load applied in the X-direction for a system with three forces and three strain gauges (Equation (3.22)).

$$\begin{bmatrix} \varepsilon_{top} \\ \varepsilon_{left} \\ \varepsilon_{right} \end{bmatrix} = \mathbf{S}^{-1} \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{3.22}$$

3.7.2 Stiffness Matrix Conditioning Inspection

A general inverse method based on beam theory was described in the previous subsection. While good in theory, in the author's project thesis, this method proved to be rather sensitive to noise in recorded sensor data, with high "phantom-loads" as a result. These "phantom-loads" are estimated outer end crane loads that are not present on the physical structure. In the project thesis, Hartmann (2019), it was discovered that altering the placement of the inverse method strain gauges seemed to lower the methods sensitivity to noise.

A measure of how sensitive a matrix is to perturbation in the input data and roundoff errors during the solution process is the Matrix Condition Number. For the linear system of equations in Equation (3.21), the condition number of $\mathbf{S}^{-1}$, cond($\mathbf{S}^{-1}$), measures the sensitivity of the solution **F** to changes in the in the input data $\boldsymbol{\varepsilon}$, Nicholas (2016). To illustrate this, assume that, for simplification, $\mathbf{S}^{-1}$ is the 2x2 matrix in Equation (3.23), and $\boldsymbol{\varepsilon}$ and **F** are vectors of length 2.

$$\mathbf{S}^{-1} = \begin{bmatrix} 11.07 & 7.56 \\ 26.19 & 17.82 \end{bmatrix} \tag{3.23}$$

$$\boldsymbol{\varepsilon_1} = \begin{bmatrix} 11.07 \\ 26.19 \end{bmatrix}, \qquad \boldsymbol{\varepsilon_2} = \begin{bmatrix} 11.071 \\ 261.19 \end{bmatrix} \tag{3.24}$$

$$F_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad F_2 = \begin{bmatrix} 0.7 \\ 0.23 \end{bmatrix} \tag{3.25}$$

Solving Equation (3.21) for **F** for the two ε in Equation (3.24), where the vectors differs only in the magnitude of noise, one finds very different solutions of **F**, as seen in Equation (3.25).

The chosen $\mathbf{S}^{-1}$ for the previous example is close to being singular. The sensitivity to noise and the resulting condition number is related to the linearity of the rows in the matrix; in other words, how close the matrix is to being singular. A singular matrix has an infinite condition number, while the further the matrix is from being singular, the closer to 1 the condition number is. The matrix condition number of a matrix **A** can be found by

$$cond_p(\mathbf{A}) = ||\mathbf{A}||_p||\mathbf{A}^{-1}||_p \tag{3.26}$$

In this thesis, the matrix condition number has been used in an iterative procedure, mounting virtual strain gauges on different places on the crane, before computing the stiffness matrix and conditioning number for the different configuration of these gauges in the inverse method. The aim was to minimize the condition number of the stiffness matrix and hence minimize the estimated phantom loads. The conditioning numbers were only calculated for the initial position of the crane. The results of the iteration can be seen in Section 4.1.3.

3.7.3 Interpolation Between Stiffness Matrices

The iterative procedure described in the previous subsection may lead to the inverse strain gauges being placed on different parts of the crane for the least sensitivity to noise. A multi-component inverse method, however, in the case of a crane with actuators, introduces a stiffness-dependency on actuator lengths as the geometry of the crane changes as the actuators move. Treating the crane as a single cantilever beam across this spectrum of positions could give inaccurate stiffness-approximations. A solution to this is to compute a stiffness matrix for all possible actuator-combinations. This, however, would be a very costly computational process, as the actuator combinations are numerous. Instead, stiffness-matrices are calculated for chosen configurations of each of the actuators, referred to as interpolation-points. The stiffness is assumed to change linearly between these interpolation-points, allowing interpolation between them. The procedure is as follows:

- For each combination of the n interpolation points for the k number of actuators, an inverse method stiffness matrix is computed and stored. Upon completion, a stiffness value for all interpolation points is available for each of the entries to the stiffness matrix.

- The stiffness values are arranged so that for each entry in **S**, there is a matrix **s** with the stiffness at each interpolation point, illustrated in Equation (3.27) for a 3x3 stiffness matrix.
- Apply an interpolation method to allow interpolation between the chosen interpolation points and the actual actuator configuration. For each actuator considered, a dimension is added in the interpolation. The crane in question has two actuators influencing stiffness; therefore, the interpolation is 2-dimensional. The implementation can be seen in Appendix A.6.

This results in a stiffness matrix **S** dependent on the lower and upper actuator, expressed as **S**(actL, actU).

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_{11} & \mathbf{s}_{12} & \mathbf{s}_{13} \\ \mathbf{s}_{21} & \mathbf{s}_{22} & \mathbf{s}_{23} \\ \mathbf{s}_{31} & \mathbf{s}_{32} & \mathbf{s}_{33} \end{bmatrix}, \qquad \mathbf{s}_{11}(actL, actU) = \begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{n1} & \cdots & s_{nn} \end{bmatrix} \tag{3.27}$$

3.7.4 Interpolation Between Rotation compensation vectors

The estimated force from the inverse method is solely based on strain gauge measurements. For accurate results, there is a need to compensate for strains induced by gravitational forces as the crane components change orientation. The procedure explained in the previous subsection was applied once more. Strain gauge values at each interpolation point combination were stored, before rearranging them to a vector of matrices for interpolation. In terms of rotation compensation, four "actuators" have to be considered for the crane; the lower and upper actuators, as well as the Crane base node rotations about the X and Y axis due to movements induced by sea-conditions. This results in a rotation compensation vector of length 3, where each entry is a 4-dimensional interpolation matrix. The rotation compensation vector can be expressed as **R**(actL, actU, rotX, rotY).

The rotation compensation vector was computed assuming quasi-static conditions. For the crane, this is justified by the assumed low frequencies of both pendulum and actuator movements. The force-frequency from irregular wave sea conditions has been inspected, and for the highest frequency design waves, the frequency of the force acting on the crane is about a third of the crane's lowest natural frequency. This has led to the assumption that the main contributions of the sea-loads are static. See figure 3.8 for visualization of the applied load frequency on the base node from the acting sea conditions.

3.7.5 Drifting compensation

The strain gauge readings may be seen to drift with time under static loading conditions. This can be caused by temperature and humidity fluctuations due to the ventilation system at the lab. To compensate for this, the amount of drifting ε_d on a neutral temperature compensation part with a strain gauge can be subtracted from the recorded strains, leaving strains caused only by external loads, gravity and dynamic effects. For a temperature

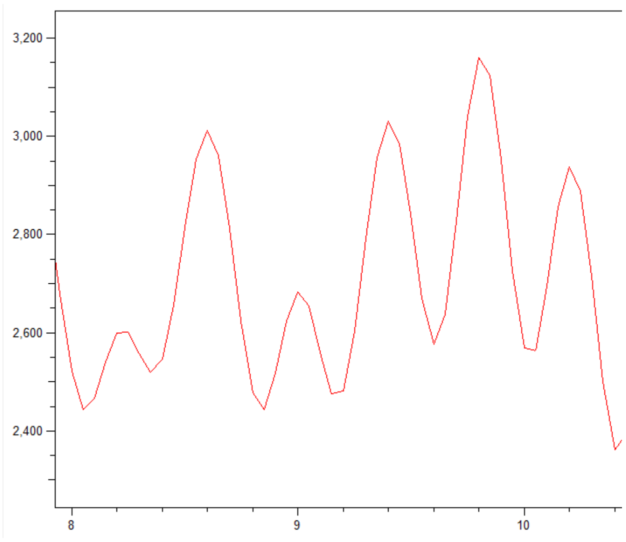


Figure 3.8: Frequency of the applied force to the Crane base node under the most severe design sea-conditions, ca. 3Hz

compensation gauge on a moving barge, the strains from the rotation of the base must be compensated for. This has not been implemented in the thesis as the barge was never realized.

3.7.6 Final Force Estimation Equation

The total equation for estimating the force vector **F** in terms of measured strains can be seen below, where ε_{tot} is the recorded strain.

$$\mathbf{F} = \mathbf{S}(actL, actU) * (\varepsilon_{tot} - \mathbf{R}(actU, actL, rotX, rotY) - \varepsilon_d) \tag{3.28}$$

3.7.7 Low pass filter

In the final equation for force estimation, the high-frequency dynamic effects from oscillations or noise have not been considered, and all calculations are performed, treating the system as quasi-static. Quasi-static assumptions are made as effects of the low-frequency waves and swinging pendulum are governed by static effects. In the project thesis Hartmann (2019), the sensor noise was successfully reduced to a magnitude of about 0.2 microstrains on average. A considerable amount of time was spent to get to this level, and reducing this noise further was not attempted as it was believed to be hard to accomplish. The measures taken to account for remaining noise and dynamic strain effects was the continued application of a Butterworth low-pass filter.

The Butterworth low pass filter is among the simplest of the low pass filters, with a ripple-free maximally flat approximation in the passband area. The order of the filter determines the rate of the roll-off rate of the filter. Higher-order filters have a faster roll-off, as seen in Figure 3.10. A Butterworth filter of order 6 with a cutoff frequency of 5 Hz was seen to filter the noise well. This was used as a sample by sample filter on recorded strain gauge data while running the Digital Twin.

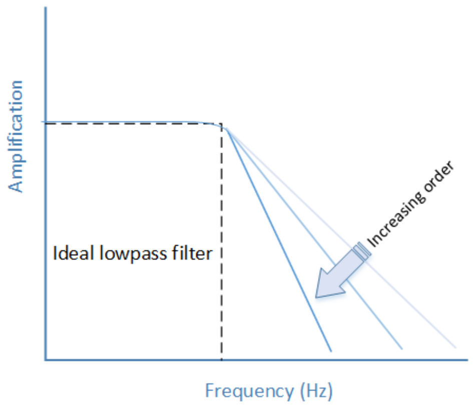


Figure 3.9: Butterworth low-pass filter behaviour

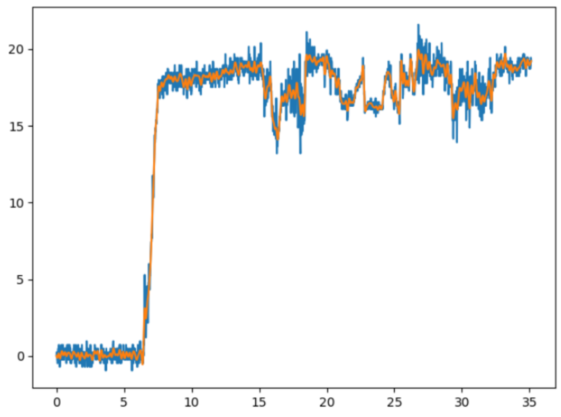


Figure 3.10: Filtered vs unfiltered strain gauge data using a low-pass filter of order 6 and cut-off frequency of 5Hz

3.7.8 FMPy

Constructing the matrices described in Section 3.7, simulating the crane FMU and co-simulation of a system of FMUs is all done by the help of FMPy.

The online FMPY documentation is sparse, hence a short introduction to some essential functions for FMU-handling with FMPy is included. Example code for creating the stiffness and rotation matrices and vectors as well as a simple co-simulation of a barge-crane system is supplied in the Appendix.

read_model_description("fmu_filename") Takes a .fmu-filename and returns an object with the identities of the FMU stored in member functions, such as .modelVariables and .valueReferences. The object is also able to set properties of the FMU such as simulation type and type definitions.

fmu_info("fmu_filename") Takes a .fmu-filename and dumps essential FMU-info to the terminal. Many of the stored properties in the read_model_description object will be printed.

extract("fmu_filename") Takes a .fmu-filename and unzippes the FMU to a temporary folder.

FMU2Slave(guid=, unzipdirectiory=, modelidentifier=, instancename=) Takes four user-defined arguments, some which are stored in the object returned by the read_model_description, and returns a FMU 2.0 slave for use in simulation.

The following functions are members of the slave FMU object returned by FMU2Slave, here called fmu.

fmu.instantiate() These four functions prepare and initialize the FMU-experiment.
fmu.setupExperiment(startTime=)
fmu.enterInitializationMode()
fmu.exitInitializationMode()

fmu.setReal("value_Reference_Number", "load_magnitude") Sets values to specified input channel of the FMU.

fmu.getReal("value_Reference_Number") Gets values from specified output channels of the FMU.

3.8 Data Handling

3.8.1 Wheatstone bridge

The electrical resistance of a strain-gauge wire can be expressed as Equation (3.29). Where R is the resistance, ρ the resistivity of the material, A the cross sectional area and L the length.

$$R = \frac{\rho \times L}{A} \tag{3.29}$$

Length change of the wire causes a change in resistance. This resistance change can be quantified by the help of a "Wheatstone bridge", an electrical circuit balancing two legs of

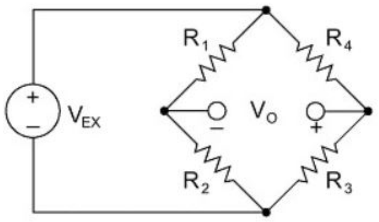


Figure 3.11: A full Wheatstone bridge

a four-element "bridge circuit". The unknown resistance is one of the two legs. See Figure 3.11 for visualization of a full bridge.

The total resistance over the bridge from a supplied voltage of V$_{EX}$ can be expressed as Equation (3.30), where R$_i$ is the resistance in each element, and V$_0$ is the output of the Wheatstone bridge. When V$_0$ is zero, the bridge is said to be balanced. Any unbalanced change in the resistors creates a nonzero output voltage. As all but the strain gauge resistance is known, the strain can be derived from Equation (3.30).

$$\frac{V_0}{V_{EX}} = \frac{R_3}{R_3 + R_4} - \frac{R_2}{R_1 + R_2} \tag{3.30}$$

In this thesis, half-bridge strain gauges with 120Ω half-bridge completion resistors were used. These properties need to be passed to the DAQ Catman software for correct strain measure.

3.8.2 TCP client server

Communication had to be established between the data source and the virtual twin. One way of establishing contact is through the use of socket programming, a way to connect two entities on a network.

In socket programming, there is one receiving end (server) and one sending end (client). There are two main ways of connecting sockets, Transmission Control Protocol (TCP), and User Datagram Protocol (UDP). UDP is a "connectionless" protocol, meaning that there is no error-checking whether the sent package was received vpnMentor (2019). This makes for the fastest of the two protocols. Data not successfully received will, however, be lost for the receiving end. The use of TCP was chosen for transmitting data from the physical model to the FMU, as confirmation of sent packages and the ability to cross-check values sent for both client and server was deemed favorable.

3.8.3 MQTT

While TCP socket programming is a reliable data transmission protocol, it is not ideal for all IoT applications. As an alternative to the traditional client-server message architecture, the Message Queuing Telemetry Transport (MQTT) protocol is using a publish/-subscribe(pub/sub) pattern. In a client-server model, the client and the server have direct communication. This is never the case for a pub/sub-model. Instead, all messages, send or received, are handled by a third component. This "message handler" is called a broker. The broker's job is to couple the message sent by a publishing client to a subscribing client, wanting to receive the message. One broker can have many clients, both publishing and receiving messages at a given time. For the broker to couple the right clients, the use of topics is introduced. A topic is essentially a UTF-8 string accompanying a pub/sub call, HIVEMQ (2019). If the topic of a client publishing a message matches that of a receiving subscribing client, the subscribing client will receive the message. The broker can run on any location. The easy access to live data from several clients at once is ideal in a digital twin where the live data may be used for several applications simultaneously. The protocol is illustrated in Figure 3.12.

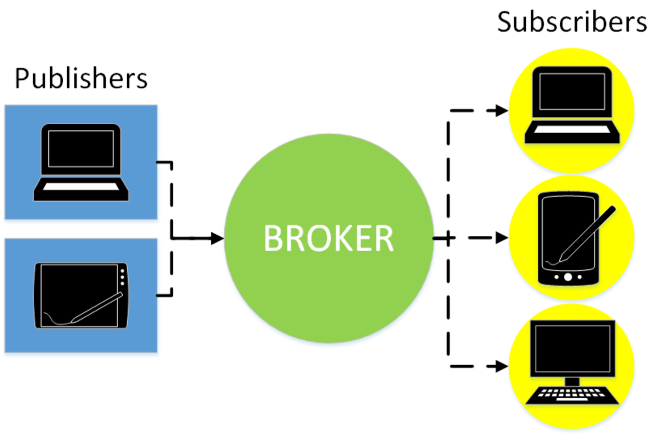


Figure 3.12: The many to one pub/sub hierarchy

The reliability of data sent over MQTT can conveniently be specified when subscribing or publishing to a topic by what is called a Quality of Service (QoS) agreement. This is a scale ranging from 0 to 2, where a message received "at most once"=0, "at least once"=1 and "Exactly once"=2. A QoS of 2 would, therefore, be as reliable as a regular TCP connection, and one could argue to remove TCP altogether. To remove the need of a locally running broker, possibly complicating the future deployment of the twin, the broker used for testing was a free online broker. The QoS of 2 on this external broker showed to transmit data marginally to slow for the digital twin in this thesis. The MQTT pythoin library paho-mqtt was used for for MQTT client-functionality, paho (2019).

3.9 Fatigue Damage

3.9.1 Implemented theory

Cyclically loading a material time and time again eventually leads to material failure, either by sudden failure or the initialization of a micro-crack that will grow as the cyclic loading continues. Failure by cyclic loading is in material science referred to as fatigue. The formation of these cracks come from complex internal material effects and are beyond the scope of this thesis. There are, to this date, no generic mechanistic way of modeling fatigue. Thus common engineering practice is to apply empiric models from decades of experience to predict fatigue behavior. Following is an introduction to the implemented fatigue damage estimation for the real-time digital twin solution.

From cyclically loading test specimens at different stress amplitudes until failure, one has obtained what are called S-N curves, displaying the relationship between stress amplitude S_a and number of cycles until failure N_i. For high cycle fatigue, with cycles to failure typically above 10^4 cycles, this relationship has proven to be well estimated by a linear semi-log curve.

Under the assumption that the fatigue behavior of the digital twin is governed by high cycle fatigue, we assume there is no plastic strain in the model. This allows the application of the Miner's linear damage accumulation rule, one of the simplest yet most used damage accumulation models. Miner's rule is an empiric accumulation model proven over time by experimental results. As shown in Figure 3.15, there is a finite number of cycles until failure N_i, at each of k different stress amplitudes for a specimen. Further, if the consumed fraction of cycles n_i until failure at all stress levels accumulate to a final damage fraction **C**, we have the Miner's rule, expressed as

$$\sum_{n=1}^{k} \frac{n_i}{N_i} = \mathbf{C} \tag{3.31}$$

Miner's rule does not consider any load sequence effects. Load sequence effects have not been considered in this thesis, but should be considered at a later stage as the loading sequence may have a significant effect on fatigue life. Usually, the Miners rule does not account for stresses below the fatigue limit, the stress level below which the material is considered to have an infinite number of cycles to failure. The fact that there is no distinct fatigue limit for Aluminium and to produce a conservative estimate of consumed life, the modified Miner's rule was used in this thesis, essentially only differing from the Miner's rule in that it includes all loading cycles.

Miner's rule assumes the number of cycles and mean-stress amplitudes to be known. This can be obtained from recorded data by applying the workflow seen in Figure 3.13.

Firstly, a peak valley extraction should be performed, to reduce the data points, and capture the stress cycles above a threshold value. Then, a popular cycle counting method called "rain-flow counting", a peak-valley based counting algorithm, is applied. The cor-

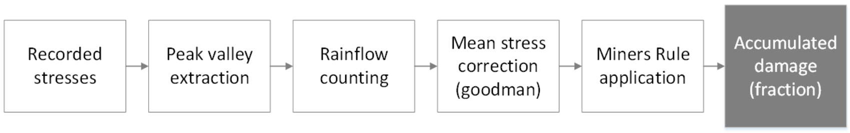


Figure 3.13: Fatigue estimation workflow

rect application of this method will return minimum and maximum stresses and the number of cycles for each unique cycle in a data-set. These stress cycles are, however, not necessarily mean-stress cycles, and to account for this, a mean stress correction called Goodman-correction, seen in Equation (3.32), was applied. This produces a new mean stress equivalent amplitude for each cycle amplitude recorded. These new stress cycles are applicable in the Miners Rule of damage accumulation. Mean stress corrections are not recommended for welded joints. Welded joints usually have internal stresses from solidification, which are hard to quantify and correct for, Niemi et al. (2018). Therefore, no Goodman correction is applied to these joints.

$$\sigma_{eff} = \sigma_a[\frac{\sigma_u}{\sigma_u - \sigma_m}] \tag{3.32}$$

3.9.2 Hot-Spot Method

For a welded structure, structural hot-spots are often found at the welds, and the weld toe in particular Livieri and Lazzarin (2005). The estimation of hot-spot stresses by the use of finite element alone is a challenging task. Welding is a mostly manual process, subject to a range of errors. On top of this, the chemical and thermal interaction between the filler material and parent material in the heat-affected zone, inducing internal stresses and micro-cracks upon solidification, is as good as impossible to accurately predict with available FEM software. Another aspect of estimating weld toe stress is the arising numerical singularities when representing the weld geometry in finite element analysis tools, as most welds include sharp edges prone to numerical divergence for nearby nodes.

An approach for estimating the stress concentration at the weld toe is the so-called hot-spot stress method. This is an extrapolation method that exploits the stress results adjacent to the weld toe to estimate the stress concentration at the hot-spot. The method effectively includes the stress concentration factor at the weld but does not consider the local notch effects caused by the welding process. When used in conjunction with the proper S-N curve, however, these effects are included in the S-N curve.

The extrapolation method used for the hot-spot stress estimation is dependent on the local finite element mesh adjacent to the weld. For the crane, the mesh is uniform and rather coarse, see Figure 3.14. In such cases, the extrapolation is typically done according to Equation 3.33, where σ_{hs} is the hot-spot stress.

$$\sigma_{hs} = 1.5\sigma_{0.5t} - 0.5\sigma_{1.5t} \tag{3.33}$$

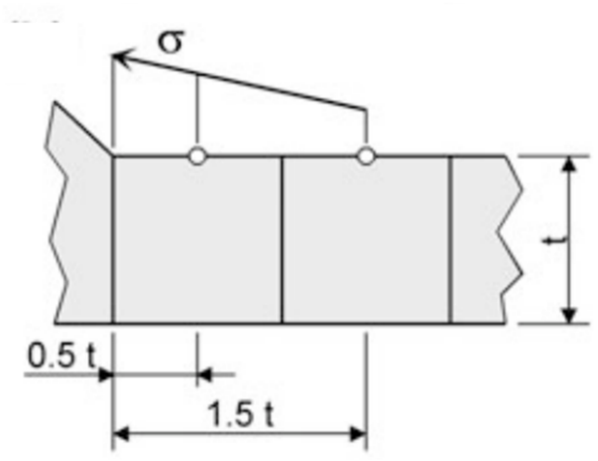


Figure 3.14: Coarse mesh extrapolation method, Figure from Niemi et al. (2018)

3.9.3 S-N curves

Choosing a valid S-N curve is normally dependent on both material, geometry and loading condition. The fatigue resistance of welded aluminum structures, however, has been seen to be independent on the parent material and is primarily governed by the weld Livieri and Lazzarin (2005). In Macdonald and Haagensen (2009), extensive experimental tests of rectangular hollow sections (RHS) for welded aluminum have been done, both for four-point bending, axial loading, and in-plane bending. These tests are of high relevance to the welded RHS profiles of the crane in question. The main outtake from the experiments is that although not entirely coinciding on a single S-N curve, and a marginal degree of scattering persists, the fatigue behavior of the RHS weld can be simplified to be loading condition independent, see Figure 3.15. The research further supports the FAT40 classification as a design S-N curve for this type of aluminum weld. The classification is defined as the fatigue strength at 10^6 cycles. The FAT40 S-N curve, 3.15, is therefore used for the fatigue life estimation of welded aluminum joints in this thesis. A standard tension S-N curve for the aluminum alloy 5754 has been used for estimating fatigue life on the hot spots not adjacent to a weld.

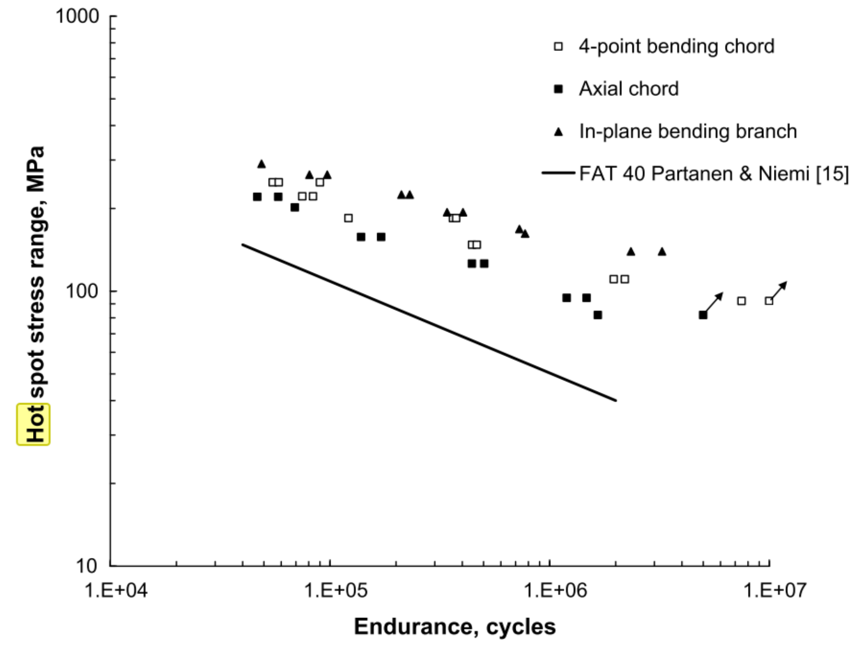


Figure 3.15: Fatigue test results for RHS aluminium welds, figure from Macdonald and Haagensen (2009)

Chapter 4

Method

4.1 Fedem crane model

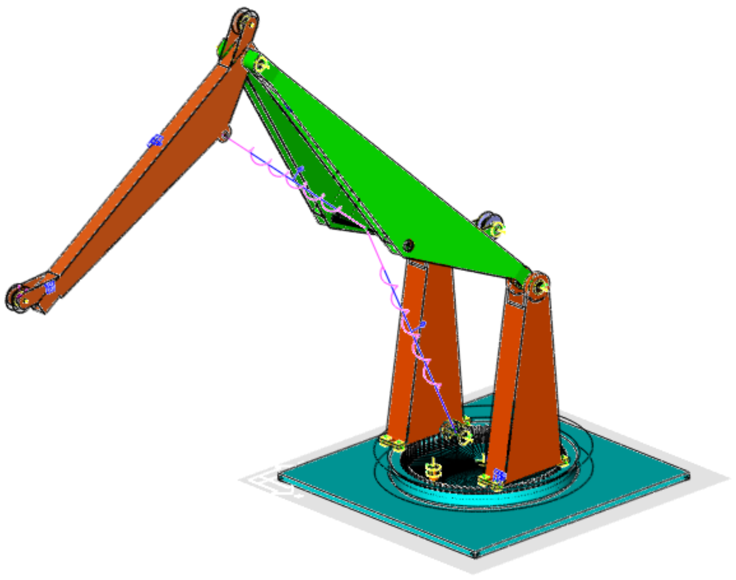
4.1.1 FE model

The unmodified finite element (FE) model of the crane was provided by supervisor Terje Rølvåg as a FEM assembly in Fedem. The computer-aided design (CAD) model was drawn and meshed in Siemens NX prior to exportation to Fedem.

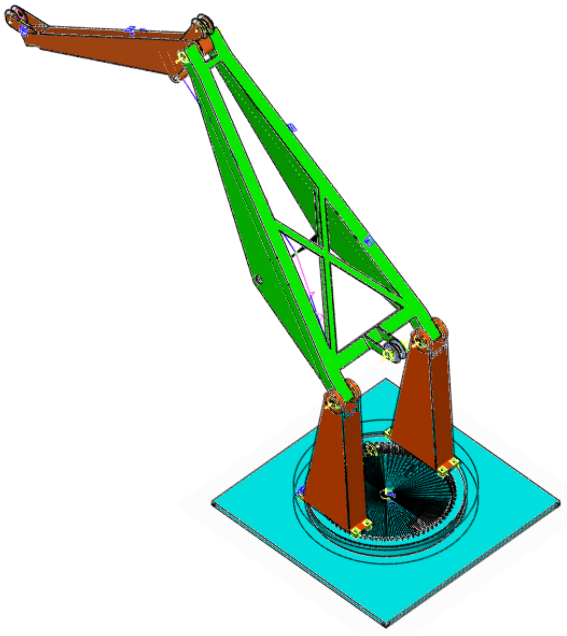
Between the CAD model was made, and the start of this thesis, several changes have been made to the physical crane geometry. The parts that were no longer identical to those on the physical model were therefore changed, re-meshed, and imported to the Fedem-model as .nas files. All crane arm parts were assigned the material used in the physical crane seen in Table 4.4. The remaining parts were assigned default steel properties. For simplification, the housing for the motor was removed. The actuator geometry was deleted and replaced by stiff axial springs and representative masses. No physical tests of actuator stiffness have been conducted. The stiffness of the springs were therefore set to represent a constant 10 cm solid aluminum cylinder with the length of the 71 cm. No spring-damper characteristics were defined for the joints in the model, although possible in Fedem. This should be included in further work as some of the joints on the physical crane have been seen to have significant play.

The mesh size and NX-type mesh used for the three major structural important parts is shown in Table 4.1. No mesh convergence test was conducted. The mesh size is, however, rather small, and a mesh convergence test might show that a larger mesh size would be satisfactory, at least for certain areas far from welds, corners and joints.

The total Fedem FEM model can be seen in Figure 4.1.



(a) Crane, view from the front



(b) Crane, view from behind

Figure 4.1: Complete Fedem crane model

Bottom node

A bottom node was attached to the crane base corners with the help of a rigid spider element. All interaction with the barge model is through this bottom node. Having 6 degrees of freedom, it can transfer both forces and moments from crane operations. This configuration of the spider element will induce an inaccurate stress distribution at the base, but this was deemed of little importance as the crane arms are of interest. See Figure 4.2 for visualization of the base node.

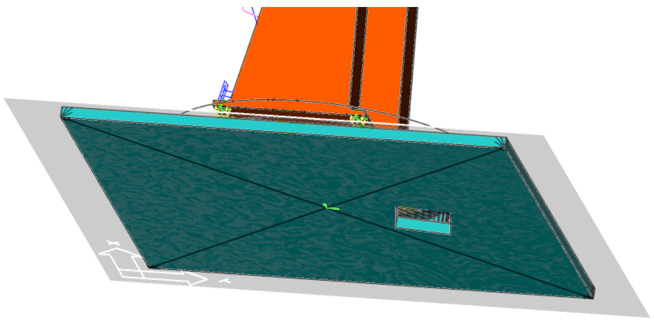


Figure 4.2: Base node for interaction with hydrodynamic barge

Part name	**Mesh type**	**Mesh size [mm]**
Lower arms	CTETRA10	10
Middle arm	CTETRA10	10
Upper arm	CTETRA10	6

Table 4.1: Mesh type and size for the structurally important parts

4.1.2 Fatigue Hotspot Identification

In Fedem, one can do a fatigue life estimation based on a single simulation. The model is wrapped in a "strain-coat", essentially placing strain gauges at all elements on the model surface. The resulting stresses at these gauges are used to give a fatigue life estimation for the simulated case and find fatigue hot spots. The S-N curves used to predict the fatigue life embedded in Fedem are all based on DNV-GL recommended practices (RP) for offshore steel structures. The crane in question is mostly made of aluminum, making the number of cycles until failure inaccurate. The outtake from the simulation, however, is the location of these hot spots for later fatigue applications based on correct fatigue properties. Figure 4.3 shows the total crane with stress concentration hot spots, Figure 4.4 shows these hot spots close up.

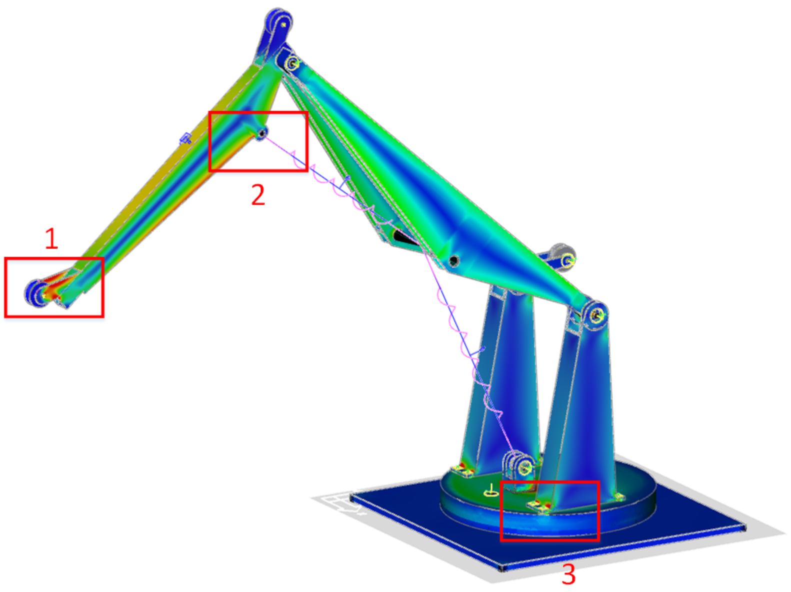


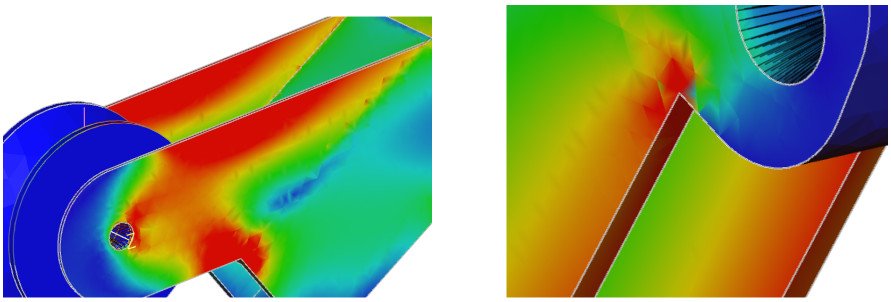
Figure 4.3: Complete crane with hot-spots shown in red rectangles, see Figure 4.4 for closeup of hot-spots

4.1.3 Instrumentation

Strain gauges for inverse method

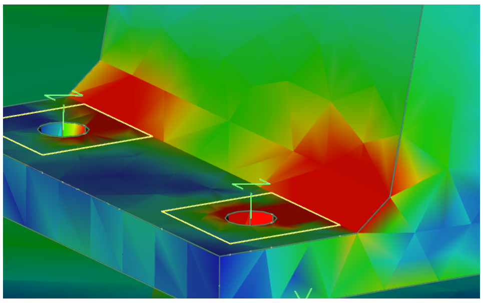
Virtual strain rosettes are a great feature in Fedem in regards to digital twins. The virtual rosettes are attached at specified nodes with specified direction, and are visualized by a blue "sensor-tag". The FE-model was instrumented with three single gauge strain rosettes for use in the inverse method. The iterative conditioning inspection method explained in Section 3.7.2 showed that the inverse strain gauge placement seen in Figure 4.5 gave the matrix the least sensitive to noise in recorded data, in other words, the matrix with the lowest conditioning number. These were named UT - upper top, MS - middle side, and MT - middle top. Figure 4.6 shows all the gauges used in the iteration procedure marked with a red dot.

More than 80 iterations were carried out by a python script computing the stiffness matrix for all the possible inverse strain gauge configurations. One could argue that instrumenting both of the middle arm "arms" would be a more general method, as the two-arm configuration may introduce bending and rotation effects hard to detect by the inverse method if only instrumenting one side. In theory, with an ideal model, this should not be an issue. It may, however, prove to be an issue under real-life operation. Based on the conditioning inspection result as well as the easier instrumentation of the left side of the crane seen from the front, it was concluded to keep these inverse sensor placements. Future instru-



(a) Hot Spot 1: Stress concentration at the outer end of the upper arm

(b) Hot Spot 2: Stress concentration hot spot on the underside of the upper arm.



(c) Hot Spot 3: Stress concentration at the lower arm close to the base

Figure 4.4: Close up of hot spots from fatigue calculation in Fedem

mentation may include an independent strain gauge on the second arm of the middle arm to increase the liability of the method.

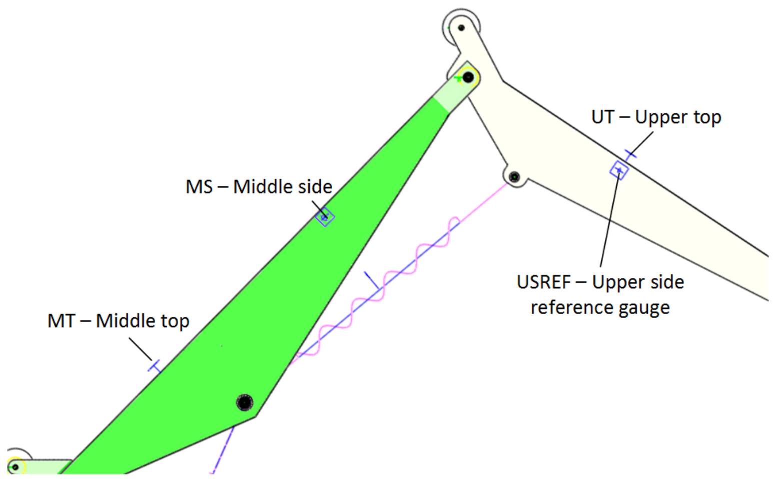


Figure 4.5: Virtual instrumentation of strain gauges used in the inverse method

Reference strain gauge

A reference strain gauge was mounted to the crane at Tyholt, to validate the inverse method by comparing the measured physical strains to the simulated strains on the digital twin. This reference strain gauge was attached on the side of the outer arm at the location seen in Figure 4.5. This was an arbitrarily chosen location.

Hot spot strain gauges

Virtual strain gauges have been placed on Hot Spot 1 and 3, as these were seen to be the ones with the lowest estimated cycles to failure. The extrapolation hot-spot method described in Section 3.9.2 of estimating stress at a weld is only applied to hot spot 3. Stresses are monitored extracting stress values from single-gauge virtual strain gauges. The placement of virtual gauges monitoring these two hot spots can be seen in Figure 4.7. No stress concentration factor was applied, as all welds were modeled as embedded in the parts. Ideally, as Fedem does not provide the possibility to extract principal stresses from a strain gauge, this should have been calculated from a virtual multi-gauge strain rosette. The simple stress measure used as of this thesis only accounts for the cyclic loading in the direction of the strain gauge. This may not be the max principal stress of the element and could overestimate fatigue life.

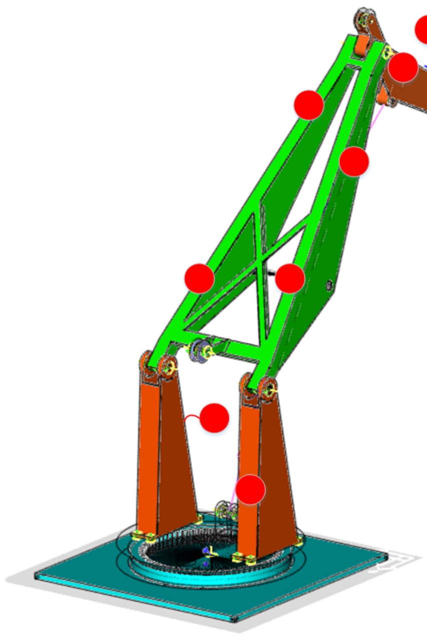


Figure 4.6: All the strain gauges available to the iteration script marked in red

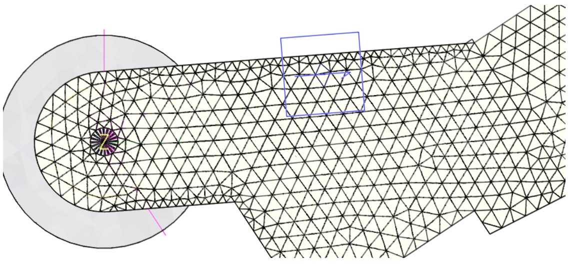
Load attack locations

The attack point for the inverse loads in the three local axis directions were configured as seen in Figure 4.8.

4.1.4 Hammer-Test and Rayleigh Damping Coefficients

The procedure described in Section 3.3 and 3.2.3 was performed experimentally at the crane lab at Tyholt. An accelerometer was rigidly mounted to the crane by the help of a steel C-clamp, as shown in Figure 4.9. It was assumed that the added weight and stiffness due to the clamped accelerometer could be neglected. The crane was subjected to an attempted pulse load by the subtle hit of a hammer. Both the point of the C-clamp attachment and the hammer attack-point was changed to cross-validate the obtained free oscillations.

The recorded oscillations from the hit of a hammer were transformed by performing an FFT on the recorded data. The FFT result in the frequency region of interest can be seen in Figure 4.10. A band-pass operation at 25 and 50Hz was performed on the recorded data, resulting in the two damped oscillations in Figure 4.11. These curves give two damping ratios that are used to estimate the Rayleigh damping coefficients.


(a) Strain gauge for hot spot monitoring Hot Spot 1

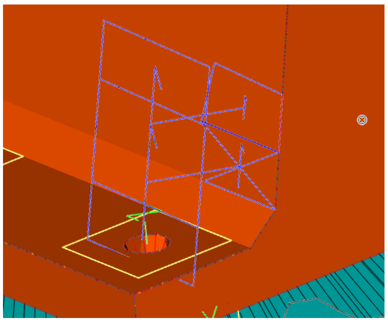

(b) Strain gauges for hot spot monitoring Hot Spot 3 by the "hot-spot" extrapolation method

Figure 4.7: Hot-spot strain gauge instrumentation

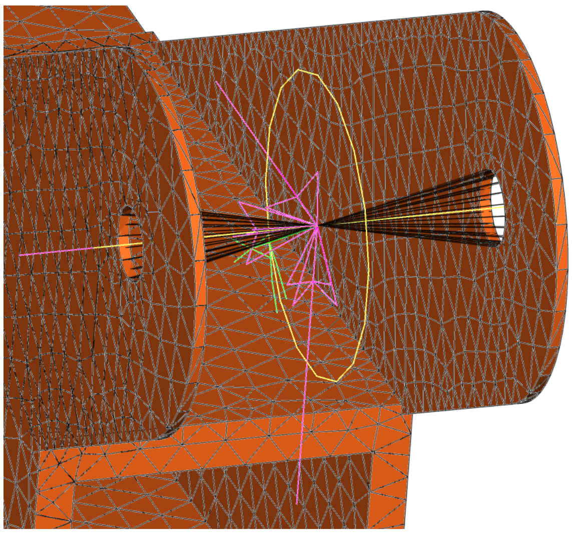


Figure 4.8: Attack point for the three loads computed by the inverse method

The hammer-tests were performed to oscillate the lowest frequencies of the system, aiming to damp these. The mode shapes of the lowest frequencies were recovered from the modal analysis in Fedem, shown in Figure 4.12. Inspecting the shapes can be convenient in determining how to apply the impulse load to ensure the initiation of specific natural frequencies.

The natural frequencies of the modal analysis compared to those measured can be seen in Table 5.2. This is discussed further in the next chapter. From the damped oscillations, the stiffness proportional damping coefficient β was estimated to be 0.0004s. The damping coefficients are used to apply damping to the material. A stiffness-proportional damping

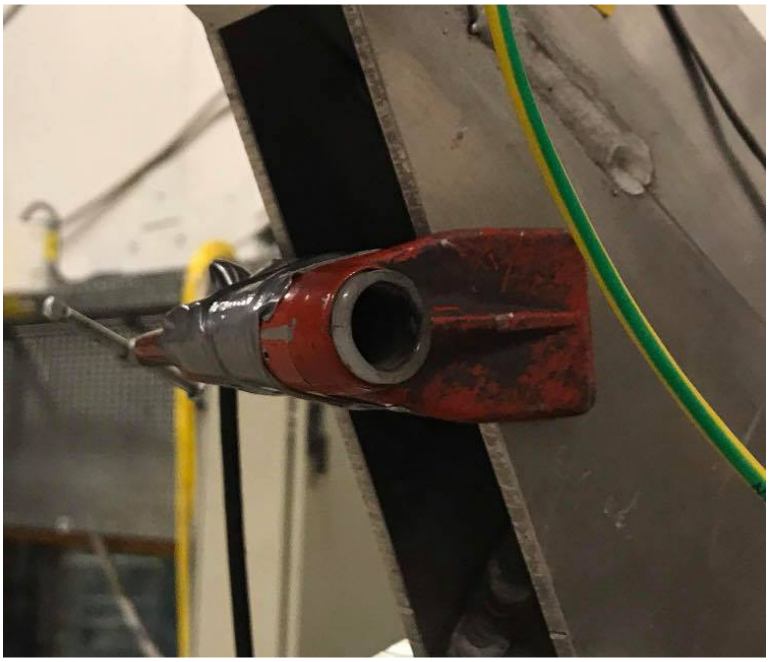


Figure 4.9: C-clamp attached to the outer crane arm

coefficient this low is, however, suspected to be lower than that of the material. Damping effects in the joints are likely effecting the logarithmic decrement of the measured natural frequencies, and applying such a low damping coefficient would not damp the Fedem model with ideally modeled joints correctly. The value of 0.0035 from the author's project was therefore kept for all the crane arms. This is a typical value for aluminum beam structures Himanshu Mevadaa (2015).

4.1.5 Buckling Consideration

Although Fedem does not have a buckling solution option in the GUI, the model reduction explained in Section 3.2.2 requires that the eigenvalue problem of Equation (3.7) is solved for the fixed interfaces. Linear eigenvalue buckling is based upon the theory already embedded in the solver. Therefore, it is possible to obtain the buckling-load of a model by applying an increasing load and continuously inspecting the model's eigenfrequencies. In theory, the buckling load is the magnitude of the increasing load as the system's lowest natural frequency approaches zero.

This "quick and dirty" buckling inspection was performed on the crane model for the case of a negative Z-axis (vertical) load. This load-case was chosen assuming critical loads under operation in the vertical Z-axis direction. Results showed that the higher the applied load in negative direction, the higher the lowest eigenvalue of the system. If applying a load in the positive Z-direction, the eigenvalue was decreasing as the load was increasing.

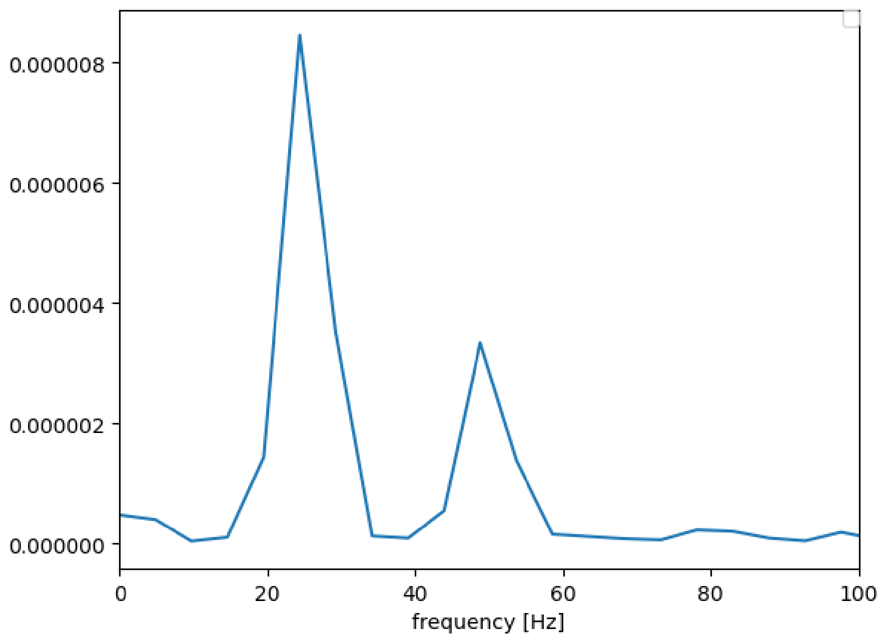


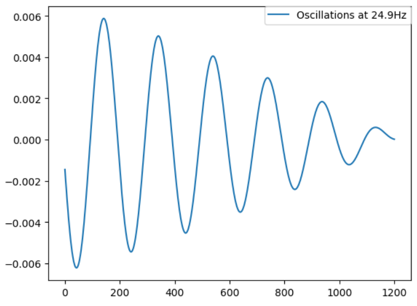
Figure 4.10: FFT on recorded vibrations after hammer impact, natural frequencies of vibration at 49.2Hz and 24.9Hz

While not a complete buckling inspection, it is natural to believe that the crane arms are in tension under lifting operations leading to compression only on the two actuators. These can be inspected as standalone entities in a future, more careful buckling consideration.

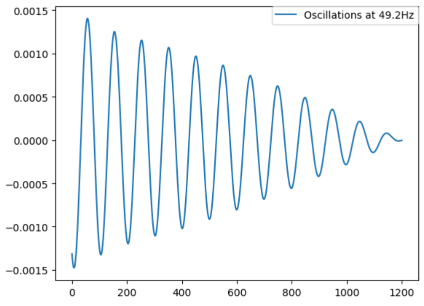
4.2 Fedem barge model

As Fedem is limited to solving the hydrodynamic loads for beam elements, the barge seen in Figure 4.13 was modeled. As opposed to the complex crane geometry modeled in NX, these beams can be modeled in Fedem directly by the use of node pairs. A node pair can be selected to have a beam cross-section between them, compatible with the Morison equation's limitations. A node was attached to all four floating elements on the top of the barge. This node was used for all interaction with the crane model.

As the final geometry of the barge to be deployed in the towing tank at Tyholt remains unknown to the author of the thesis, this design was presumed to be sufficient for conceptual testing purposes. The hydrodynamic properties of the drag diameter D_b and buoyancy diameter D_d were set to the actual diameter of each floating element. The beam section properties can be seen in Table 4.2. Hydrodynamic properties were disabled for the con-

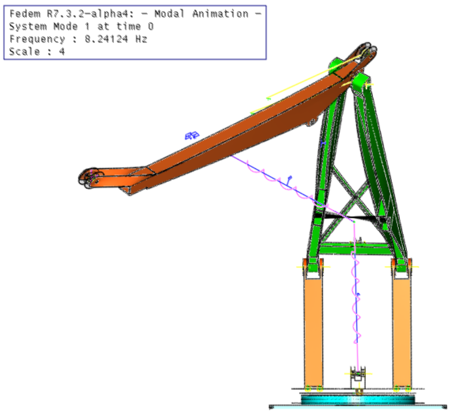


(a) Oscillations at 24.9Hz

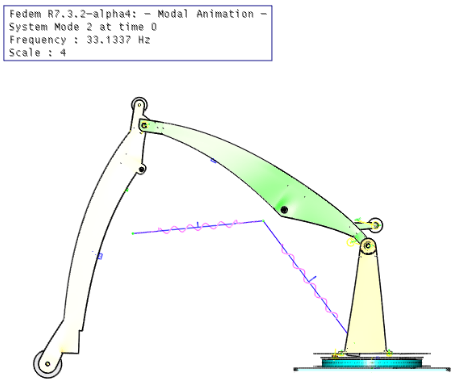


(b) Oscillations at 49.2Hz

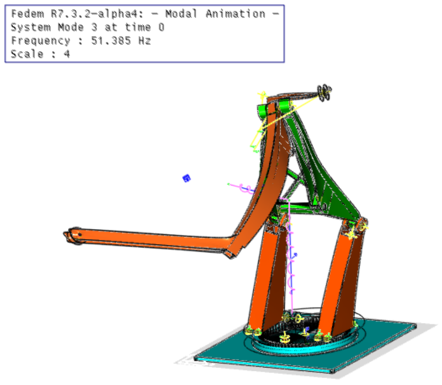
Figure 4.11: Oscillations at natural frequencies of physical crane



(a) Mode 1, 8.24Hz



(b) Mode 2, 33.13Hz



(c) Mode 3, 51.38Hz

Figure 4.12: The three lowest frequency mode shapes

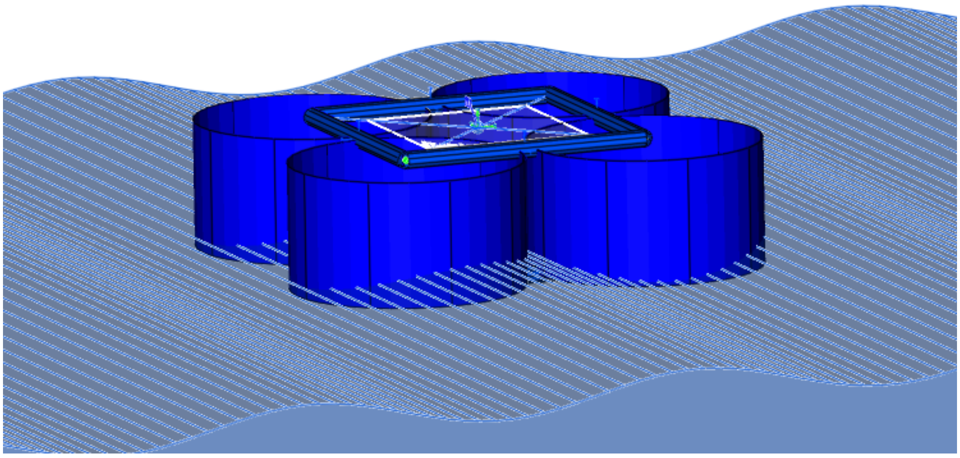


Figure 4.13: Barge model for FMU export

nection elements connecting the floating elements.

Beam Cross Section	D_{outer}	D_{inner}	D_b	D_d
floating elements	1.4	1.39	1.4	1.4
connector elements	0.1	0.09	N/A	N/A

Table 4.2: Beam cross section properties

4.3 Assembly Model for Validation

As Fedem cannot import one model into another for simple assembly, the barge was modeled inside an instance of the Fedem crane model. The complete assembly can be seen in Figure 4.14. The properties of the assembly are like those for the crane and barge.

4.4 Setup of physical system

4.4.1 The crane

The physical crane is located at the center of Marine Technology in Trondheim, Norway. The development of the crane geometry is described in the master thesis of Gyberg (2017), and is not given in detail in this report. A figure of the full crane is seen in Figure 4.15.

A short overview of the crucial parts of the physical crane is seen in 4.3 below.

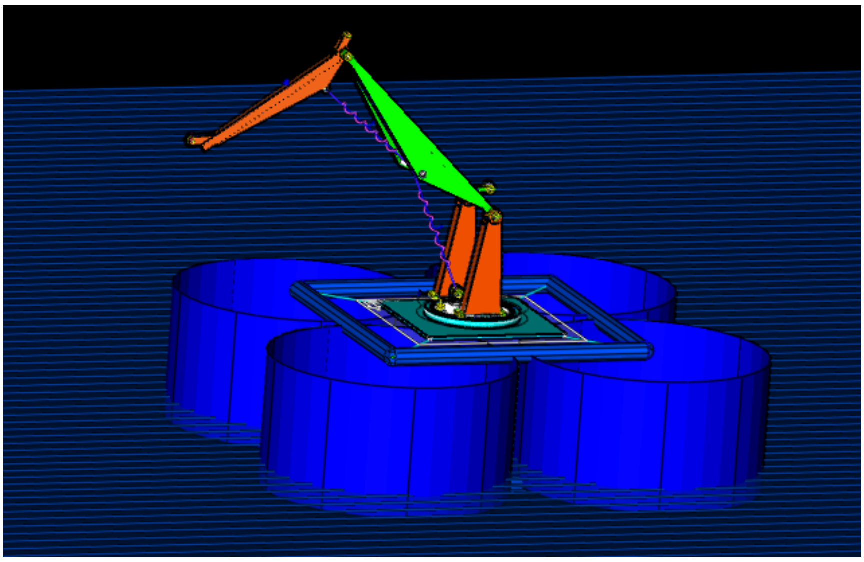


Figure 4.14: The crane and barge assembly inside Fedem

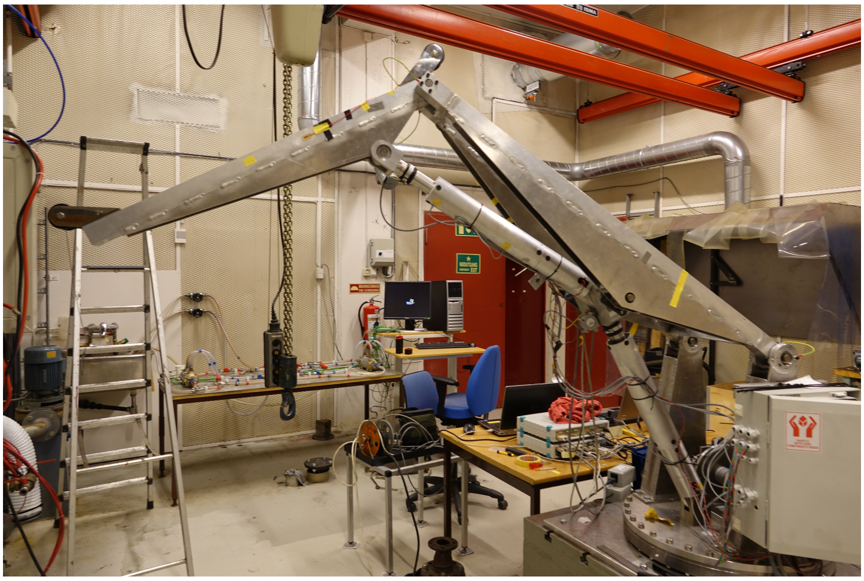


Figure 4.15: The whole physical crane

All four arms of the structure are of aluminum with the properties seen in Table 4.4. The electrical system setup is not discussed in this thesis.

Parts
Lower arms
Middle arm
Upper arm
Slewing plate
Screw-type upper actuator
Screw-type lower actuator
Base rotation actuator
Various axles
Pulleys

Table 4.3: Significant crane parts

Parameter	Value	Unit
Material Specification	EN AW 5754-H22	
Density	2.66	g/cm^3
Modulus of elasticity	68	GPa
Yield Strength	130	MPa
Tensile Strength	130	MPa

Table 4.4: Aluminium properties collected from Gyberg (2017)

4.4.2 Instrumentation

The physical crane was, like the Fedem-model, instrumented with three single gage strain gauges for use in the inverse method. These were mounted at the upper top, middle side, and middle top. A reference strain-gauge was attached on the side of the upper arm, as shown in Figure 4.5. While not used as discussed in the next chapter, an additional temperature compensation strain gauge was attached to a non-rotating unloaded part. The gauges used in the inverse method were the waterproof strain gauges WFLA-6-11-1L from the TML company, further specifications can be seen in Lab (2019). The middle top (MT) inverse method strain gauge can be seen in Figure 4.16.

The encoder for the base engine was used to keep track of the base rotation. The number of output impulses per degree of rotation was found, and the relationship seen in Equation (4.1) was established. Rotation is given in radians in the Fedem FMU and needs to be converted accordingly.

$$\phi = OutputImp \times \frac{10^o}{16500Imp}[deg] \tag{4.1}$$

The physical crane actuators do have encoders that can be used to measure actuator-movement, these are, however, hard to access when the crane is assembled, and time did not allow access to these. Instead compact string potentiometers were used. These were

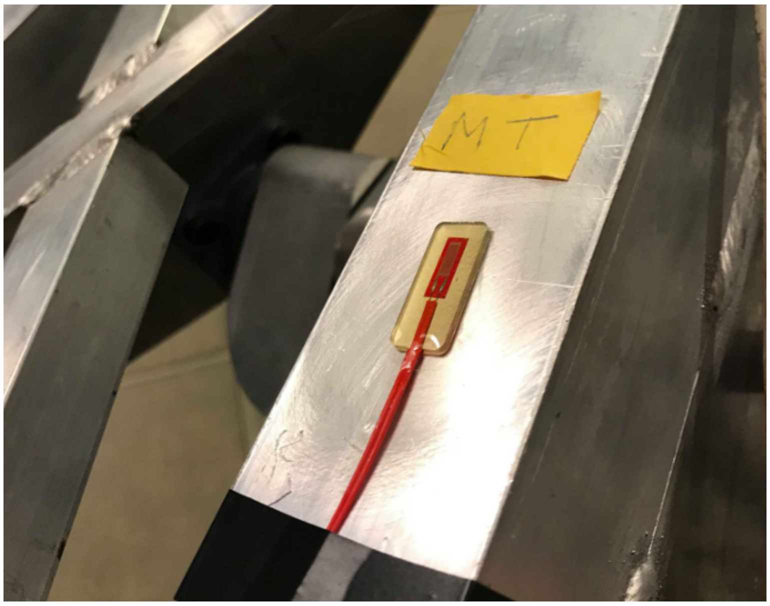


Figure 4.16: Middle top strain gauge

mounted to measure the actuator elongation, as seen in Figure 4.17, and have a 0.25% offset specification Specialties (2019). The four strain gauges, two potentiometers, and one encoder reading result in a total of seven inputs from the crane to the Spider.

4.4.3 Data Acquisition Software

The analog data from the crane is converted to a digital signal with the help of a Spider 8 DAQ module. The data was processed and stored using Catman v4.2.1 software on a computer directly connected to the Spider. It was attempted to stream data directly from Catman over TCP protocol, but the version of the software did not allow wireless real-time transmission. A second computer was running LabVIEW 2016, a software primarily responsible for enabling actuator movement. No modifications were made to this second computer during this thesis.

The setup in Catman with all inputs used is shown in Figure 4.18.

4.4.4 The barge

The physical barge at the Marine Centre at Tyholt was unfortunately not finished in time for the physical testing to take place during this thesis.

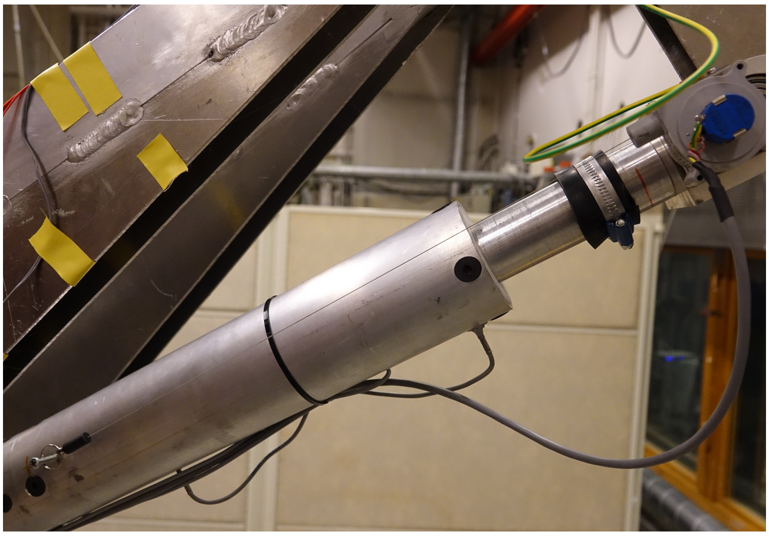
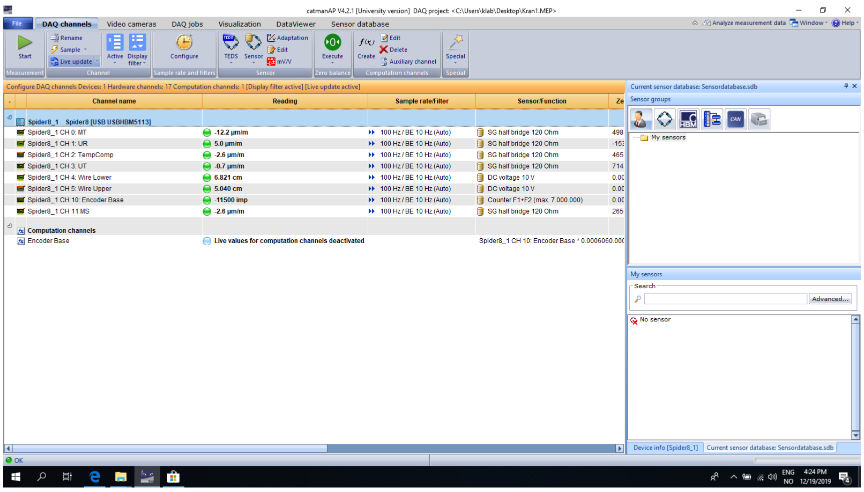


Figure 4.17: Potentiometer; wire going from screw (lower left) to potentiometer (upper right)



Configure DAQ channels Devices: 1 Hardware channels: 17 Computation channels: 1 [Display filter active] [Live update active]

Channel name	Reading	Sample rate/Filter	Sensor/Function	Ze
Spider8_1 Spider8 [USB USBHBM5113]				
Spider8_1 CH 0: MT	**-12.2 µm/m**	100 Hz / BE 10 Hz (Auto)	SG half bridge 120 Ohm	498
Spider8_1 CH 1: UR	**5.0 µm/m**	100 Hz / BE 10 Hz (Auto)	SG half bridge 120 Ohm	-153
Spider8_1 CH 2: TempComp	**-2.6 µm/m**	100 Hz / BE 10 Hz (Auto)	SG half bridge 120 Ohm	465
Spider8_1 CH 3: UT	**-0.7 µm/m**	100 Hz / BE 10 Hz (Auto)	SG half bridge 120 Ohm	714
Spider8_1 CH 4: Wire Lower	**6.821 cm**	100 Hz / BE 10 Hz (Auto)	DC voltage 10 V	0.00
Spider8_1 CH 5: Wire Upper	**5.040 cm**	100 Hz / BE 10 Hz (Auto)	DC voltage 10 V	0.00
Spider8_1 CH 10: Encoder Base	**-11500 imp**	100 Hz / BE 10 Hz (Auto)	Counter F1+F2 (max. 7.000.000)	0.00
Spider8_1 CH 11 MS	**-2.6 µm/m**	100 Hz / BE 10 Hz (Auto)	SG half bridge 120 Ohm	265
Computation channels				
Encoder Base	**Live values for computation channels deactivated**		Spider8_1 CH 10: Encoder Base * 0.0006060.000	

Current sensor database: Sensordatabase.sdb

Sensor groups

My sensors

Search

Advanced...

No sensor

Device info [Spider8_1] Current sensor database: Sensordatabase.sdb

OK

Figure 4.18: Catman configuration

4.5 Co-Simulation

The interconnection between the crane and barge FMU was at a single node in each of the subsystems. This was identical for all connectivity configurations. The connection-node on the barge is at the top center, and on the crane on the underside of the base.

4.5.1 Functional Mock-Up Units (FMUs)

The FMUs were both exported from Fedem. The internal time-step of the embedded solver of each FMU was set to 0.05s. Smaller time-steps proved too slow for real-time simulation and deemed unnecessary as convergence was quickly reached fast with the larger time-step.

The Fedem Crane FMU

The crane FMU has several inputs and outputs. The controllable elements of the crane are the upper, lower and base rotation actuators. These are inputs to the FMU. The applied lift forces attacking at the outer end of the upper arm and the positions at the connection-node between the crane and the barge yields another nine inputs, making a total of 12 inputs. The strains measured at all virtual strain gauges will be outputs from the FMU. These were used both in the inverse method and for structural monitoring and fatigue calculations. In addition, the six reaction forces at the connection node between the crane and barge were outputs from the FMU. Crucial, when specifying the force outputs of the FMU, is using a linear slope of -1 rather than using a 1:1 function inside Fedem. The argument used for the crane output forces are the extracted reaction forces at the bottom node. To apply these as "crane-loads" to the barge FMU, the sign of these reaction forces must be changed. Table 4.5 shows names and reference numbers of the input and outputs to the Fedem crane FMU.

Input Name	FMU ref.	Output name	FMU ref.
Actuator_Translation_Lower	0	UT_strain_inv	12
Actuator_Translation_Upper	1	UR_strain	13
X_Force	2	MT_strain_inv	14
Y_Force	3	outFx	15
Z_Force	4	outFy	16
inPosX	5	outFz	17
inPosY	5	outFRx	18
inPosZ	5	outFRy	19
inRotX	5	outFRz	20
inRotY	5	MS_strain_inv	21
inRotZ	5	outerEndStrainGauge	22
		upperHotspotGauge	23
		lowerHotspotGauge	24

Table 4.5: List of input and output names of the crane-FMU

The Fedem Hydrodynamic FMU

The input and outputs of the hydrodynamic barge FMU are reversed of those used for force-position exchange in the crane for them to be coupled. This leads to the barge having forces as input and positions as output at the connection-node. The total number of pins is listed in Table 4.6.

Input Name	**FMU ref.**	**Output name**	**FMU ref.**
inFx	0	outPosX	9
inFy	1	outPosY	10
inFz	2	outPosX	11
inFRx	3	outRotX	12
inFRy	4	outRotY	13
inFRz	5	outRotZ	14
inAmplitude	6	amplitude	15
inTimeShift	7	timeShift	16
inFrequency	8	frequency	17

Table 4.6: List of input and output names of the barge-FMU

4.5.2 Ansys Twin Builder

The work of the thesis contributed to the development of Fedem software. By failing to co-simulate and compile Fedem-FMUs in Ansys software, several changes were made by Runar H. Heggelien to enable these features. The final compilation of a working co-simulation to a single FMU or Twin was, however, never accomplished due to compilation bugs on the Twin Builder software. The co-simulation was, however, running inside the graphical user interface (GUI), and the simulation time was about a third faster than the co-simulation using FMPy. This is worth noticing and may be of motivation for future efforts to solve these compilation issues. Figure 4.19 shows the working co-simulation inside Ansys Twin Builder.

4.5.3 FMPy Implementation

As the export of a Twin Model for the complete simulation was not accomplished. The system was set up in in the FMU-API FMPy. The co-simulation framework of the two coupled subsystems is rather intuitive, connecting each of the input and output pins of the two FMUs by the use of embedded get and set functions of FMPy between each time-step. The remaining pins of the crane were used in both the inverse method and stress calculations. See Figure 4.20 for interconnection illustration for the force-displacement configuration.

4.5.4 Tested Configurations

The numerical challenges outlined in Section 3.1, led to the different connectivity configurations seen in Table 4.7.

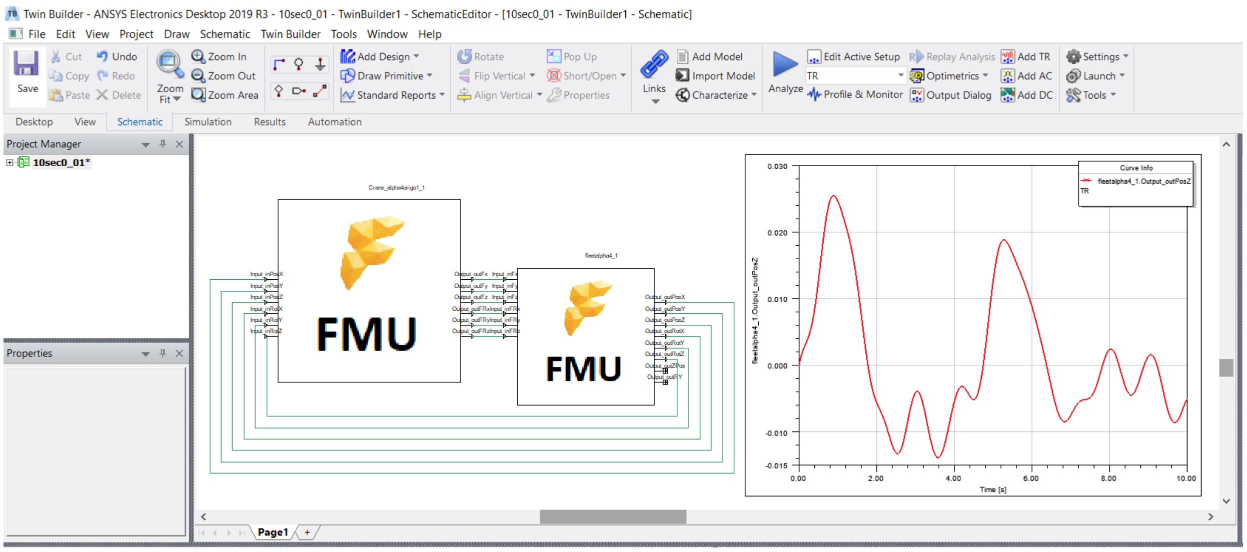


Figure 4.19: Working co-simulation inside Ansys Twin Builder

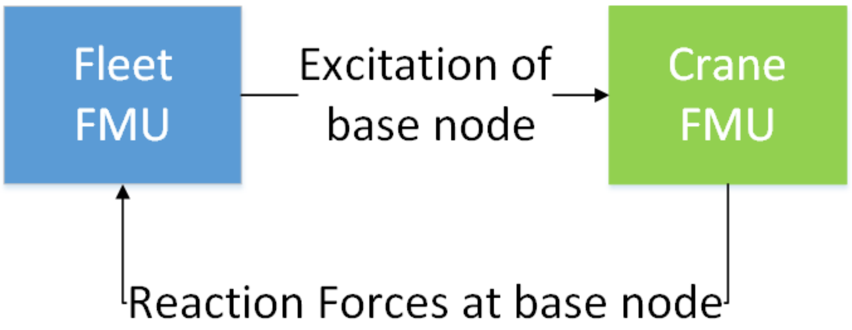


Figure 4.20: FMU interconnection

Interconnection type	Crane input	Barge input
Force-force	Forces	Forces
Force-displacement	Displacements	Forces
Force-velocity	Velocity	Forces

Table 4.7: FMU coupling configurations

4.6 Rotation Compensation and Stiffness Matrices

4.6.1 stiffness matrices

Appendix A.6 shows the python script used to construct the actuator dependent stiffness and rotation compensation matrices and vectors for the crane. The python script implementing the FMPy-API was used as an "FMU-master" in constructing the matrices.

The stiffness matrices were constructed by applying virtual unit loads to predefined load-inputs in the Fedem FMU, as shown in Figure 4.8. The procedure was repeated for every interpolation point chosen for the two actuators. The loads were applied in each of the local axis directions in turn, and residual strains for each load were recorded. The resulting stiffness matrices were stored in a .npy file.

The condition-number of the different stiffness matrices at different actuator configurations were seen to vary from 3-21, which is still very satisfying compared to the matrices applied in the project thesis, Hartmann (2019).

4.6.2 Rotation Compensation

The rotation compensation vectors were constructed identical to the stiffness matrix, but instead of unit loads, the resulting static strains at each configuration were stored. These vectors were also saved to .npy files.

4.7 Cloud Based Monitoring System

Initially, there was planned implementation of the digital twin solution of this thesis to the cloud-based monitoring system (CBMS) of previous information and communication technology (ICT) project and Master thesis students at the Department of Mechanical and Industrial Engineering at NTNU. This was made impossible due to the currently ongoing bottom-up reconstruction of the CBMS. As an alternative, a monitoring system was developed for this thesis. While not run on a remote server, it will still be denoted CBMS as all components can easily be run remote.

Top Down Framework

For the successful implementation of the twin to an IoT-solution, an overall framework needs to be established; a plan for how the elements of the twin are going to interact. The rough framework of the twin was drafted and redrafted as the software capabilities, and ease of implementation became clearer. The final framework used for the digital twin solution is shown in Figure 4.21.

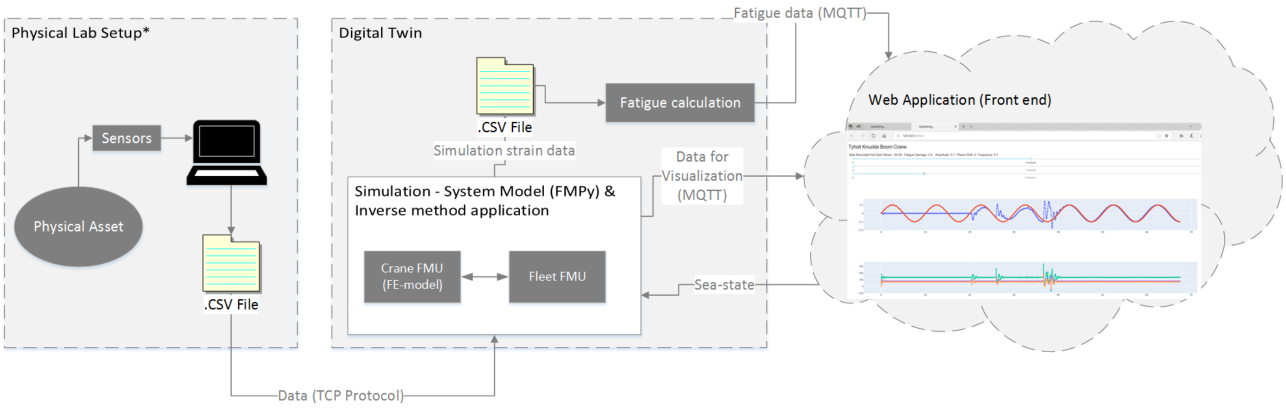


Figure 4.21: Framework for the digital twin

Simulation

The recorded strain data is sent from a CSV file appended by the Catman software, to a listening python client. This data is sent in the frequency of the simulation time-step. In the python file containing the client is also the FMPy co-simulation master. Upon receiving a data-string, the data is filtered before the simulation steps forward in time. In this way, the virtual and physical models are aligned in time. For each time-step, strain data is stored in a local CSV file for damage calculation. The strain data, as well as some data describing the position of the system, is sent over an MQTT protocol to the web application at the "Strains" and "Positions" topics, respectively.

Accumulated damage calculation

The accumulated damage calculation is running alongside the system simulation in a separate process. The calculation is performed every five seconds, publishing the accumulated damage result to the "Damage"-topic. The process of estimated fatigue life explained in Section 3.9 fatigue is implemented, resulting in a single floating value at each point of interest representing the fraction of accumulated damage. These fractions are published to the "Damage"-topic along with a total current run-time.

Data Transmission

A python client was configured to read sensor data generated by the Catman DAQ software and pass them to the listening FMPy-based twin. As the physical crane-Bbarge system was never realized during the thesis, the data transmission from Catman is done for the digital twin case with the crane only. A similar procedure is applicable to the co-simulation of crane and barge. The running digital twin co-simulation is based upon data fabricated in Fedem.

The MQTT protocol is used by the web application, the simulation and the damage accumulation calculation. The simulation and damage accumulation scripts publish data to their respective MQTT topics for visualization in the web application. The simulation script also subscribes to the "SeaState"-topic to specify the sine wave parameters during simulation. The web application subscribes to the published data from the Simulation and damage accumulation scripts for cloud visualization. It also published the slider changes to the "SeaState"-topic.

Web Application

The web application is based on the python library Dash. Dash is an open-source user interface library for creating analytical web applications. Live plots have been implemented for comparing the recorded and simulated strains. One can keep track of entities such as accumulated data, maximum recorded stress, and total run-time through a simple panel. The application also has a plot showing the Z-position movement of the connection node in global coordinates. These movements are included, as described in Section 4.8, to validate and ensure alignment of the physical and virtual hydrodynamic simulation. Lastly, the application has three interactive sliders. The sliders are used for aligning the physical and virtual sea states, through updating the co-simulation by MQTT publish-calls. This alignment is subject of the next subsection.

4.8 Physical and virtual alignment

Crane

Attempts have been made to align the physical crane and virtual crane FMU prior to the FMU initiation, but they have not proven successful. This leads to the continued use of the aligning method by moving the virtual actuators until they are a threshold apart from the physical crane before the virtual model follows the physical crane. Loads are applied only after alignment. It is assumed that the physical crane strain gauges have been zeroed while waiting for the simulation to align, and thus there are induced strains in the simulation once it reaches alignment. This can be accounted for by subtracting the strains induced by alignment from the simulated strains and subtract the rotation compensation strains computed at alignment from all future strain compensations.

Co-Simulation

The towing tank at Tyholt is assumed to produce ideal simple regular waves. Thus the sea state of the simulated barge as a sine wave is sufficient for conceptual testing. For a real-time digital twin solution, however, the replication of the wave alone is insufficient, as the system's dynamic behavior during lifting operations also depends on *where* on the wave the system is located.

The thesis contributed to the implementation of a new wave definition in the Fedem GUI, an "External Wave function". To use this with Fedem version R7.3.2-alpha4, an additional "ExternalWaveGenerator.dll" must be included in the Fedem plugins folder. This wave definition is simply a sine-wave that can be parameterized through a wave amplitude, frequency, and phase shift. The input to these parameters can be defined as an external function inside Fedem, hence creating an input channel to the FMU. This is a powerful tool for the digital and physical alignment, as it allows wave control after the FMU has been generated, as opposed to the previously constant sea-state. The use of this feature in the cloud-based monitoring system (CBMS) is visualized further in 5.6.

Challenges

Aligning a virtual and physical hydrodynamic model is possible if the current sea state is simple, such as a regular ideal sinus wave. In the real world, an offshore sea-state is a complex mix of many super-positioned sine-waves, as explained in Section 3.4.1. Aligning a hydrodynamics simulation to a physical asset and having it follow the exact motion of the asset in the given sea-state is therefore rather unrealistic for an irregular wave. However, the use of a simple regular wave that can be replicated and simulated in real time is an excellent indication of the model accuracy. It is proof of concept to determine if more advanced wave matching methods should be studied. More advanced methods could include some kind of free surface mapping to move a virtual free surface in pair with the physical free surface.

Chapter 5

Results and Discussion

In this chapter, the results of interest are presented and discussed. Results are obtained both from simulation with data from the physical crane at Tyholt and with "Fedem fabricated data". By running a Fedem model with actuator movements and applied loads inside Fedem and exporting results for use in the inverse method, one eliminates sources of error related to noise and model inaccuracy. As these are numerous and may be hard to quantify, such an ideal run is of great value for validation purposes. Units of the presented results are as seen in Table 5.1.

Property	**Unit**
Forces	[N]
Strains	[μm/m]
Positions	[m]
Rotations	[rad]

Table 5.1: Units used in the presented results

Fedem fabricated data was used to validate both the crane alone and the co-simulation. The two following subsections present and discuss the accuracy of the inverse method for the cases of a crane alone and the co-simulation of a barge-crane system. The most attention is given to the latter.

5.1 Hammer-Test

Finding a natural frequency of a physical structure compared to that of a finite element model is a good indication of the accuracy of the model. The natural frequencies from modal analysis compared with those found by the hammer-test is seen in Table 5.2. The lowest frequency from the modal analysis is seen to be 8.24Hz. This mode is oscillating out of the lifting plane of the crane. The crane at Tyholt, unfortunately, has significant play under loading in this plane. The Fedem model, however, does not have this modeled, although possible. This play made it difficult to obtain consistent natural oscillation frequencies out of the lifting plane, and all tests performed were seen to give different results. The unpredictable oscillations are believed to be caused by induced damping by play in the joints. Modeling the play in the joints or eliminating it in the physical structure is necessary to obtain accurate results for this first mode. Elimination of play by altering the physical configuration of the crane is advised. The significant play in the physical structure is assumed to be hard to model in a reliable fashion.

Mode	**Hammer-test [Hz]**	**modal analysis [Hz]**
Mode 1	None	8.24
Mode 2	24.9	33.13
Mode 3	49.2	51.38

Table 5.2: Natural frequencies from hammer-test and modal analysis compared.

In the lifting plane, there were two prominent frequencies of oscillations for all tests performed, seen in table 5.2. These are believed to represent the second and third lowest frequencies found in the modal analysis. These modes are both excited in the lifting-plane of the crane. Additional damping due to play in the joints is relevant for these as well, but far less than for the first mode as it was possible to apply the pulse load without causing significant joint movement. The modal analysis is seen to over-predict the natural frequencies of the physical system. This is expected, as the simplified Fedem model essentially makes a stiffer structure than in reality. The joint connections are unrealistically stiff in the model. Additionally, the hollow cross-sections of the middle and upper arm are spot welded to the longitudinal stiffeners in the real-life crane, as seen in Figure 5.1. In the Fedem model, these are simplified to be part of the hollow cross-section, hence creating contact along the whole length of the beam, increasing the stiffness as a whole. It can be concluded that there are deviations in the physical and virtual model that may affect the inverse method accuracy.

5.2 Sensor Drifting and Noise

From Figure 5.2, one can see that the noise in the temperature compensation strain gauge is exceeding any trend in the recorded data. It was therefore concluded that to remove the temperature compensation in the simulation would be safer than to introduce another source of error by including it. The potentiometers also produced noise; this was mostly filtered and assumed to affect the simulation much.

The noise seen in Figure 5.2 was recorded with the electrical system of the actuators turned

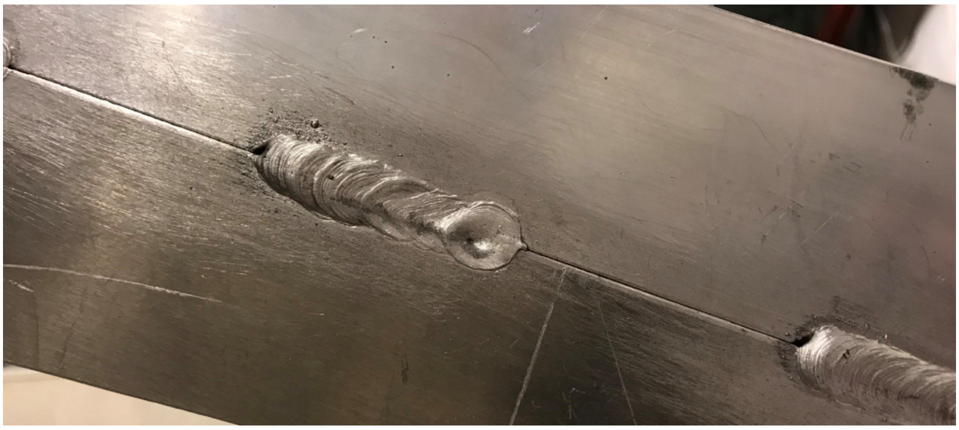


Figure 5.1: Spot welds on the Tyholt crane

off. With the actuators enabled, the recorded noise increased significantly, even without any actuator movement, as seen in Figure 5.3. This strong strain gauge dependence on the electrical system of the actuators is likely caused by the proximity of the two systems causing electrical interference. Attempts were made to move and shield the systems, but with minor effects. Unfortunately, the increased noise from the electrical system's interference was not constant throughout a generic run, but somewhat dependent on actuator speed and position. When both actuators were run, one strain gauge could drift orders of magnitude away from the initial zero value, even when returned to the initial position with no load applied, while another strain gauge was less affected. The drift was hard to predict and seemingly impossible to remove with the current lab setup. While the drift was still in the order of micro-strains, this unreliability made the resulting estimated loads and strains accuracy highly fluctuating.

The interference effect was not present when the actuators were disabled; hence far better force and independent strain gauge estimations were for the virtual digital twin for this case. This is discussed further in Section 5.5.2. These effects are of even more significant concern in the real-life application of a digital twin, where environmental effects are much larger than in a lab with controllable surroundings. Therefore the drifting may increase and come from additional sources. Taking drastic action to decrease disturbance to the strain gauge readings is recommended, such as reinstalling the electrical systems and study possible isolation methods to protect the strain gauges from environmental effects.

Decent strain gauge readings were however recorded, the digital twin results can be seen in Section 5.5.

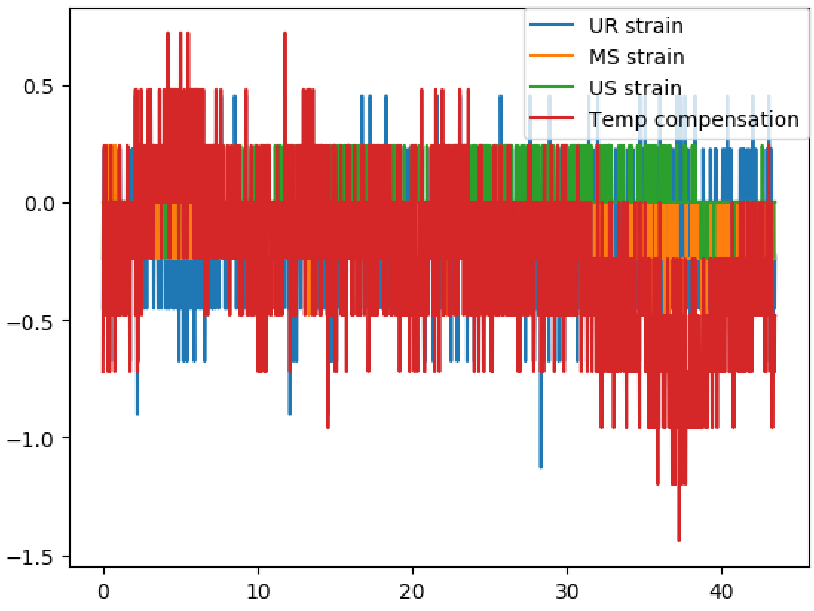


Figure 5.2: Strains from static testing, temperature compensation strain gauge in red

5.3 Crane Simulation - Fabricated Data

Inside Fedem, both the crane actuators were moved with sudden stops. 500N was gradually applied to the outer end of the upper arm on the crane over 10 seconds before it was kept constant. While this load case is rather simple, there are challenges in sudden stops of the actuators. These were applied on purpose to introduce dynamic effects.

The high peaks in the estimated force in Figure 5.6 is due to the sudden start and stops of the actuators during the simulation. These abrupt dynamic changes result in strains caused by inertia forces in the structure. The inverse method being based on quasi-static assumptions is not capable of accounting for this. It can be seen that for such abrupt movements, the phantom loads are rather large. Under real-life applications, however, the crane lifting operations are more low frequent. Apart from these abrupt change regions, the inverse method replicates the strains and applied loads very well, as can be seen in Figures 5.4 and 5.5.

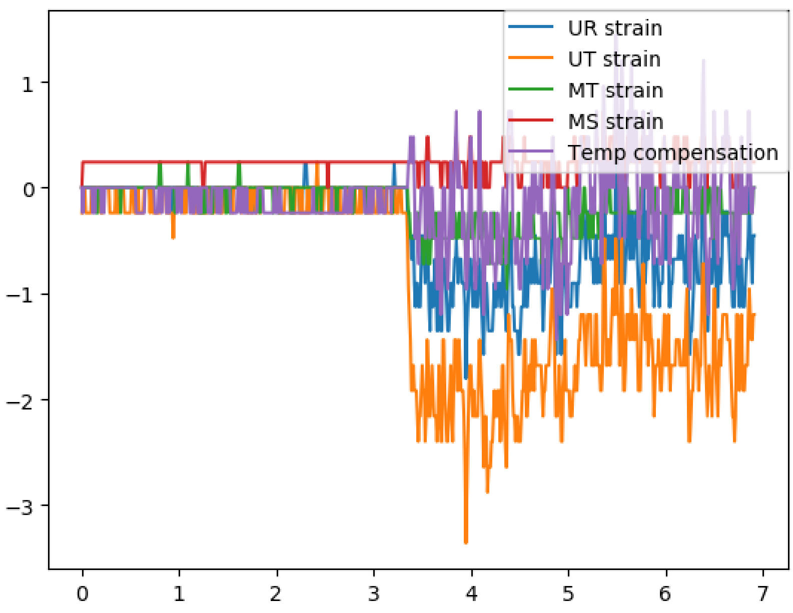


Figure 5.3: Strains from enabling the electrical system of the actuators, no load applied

5.4 Co-Simulation

The barge and crane were assembled inside Fedem to create a system of able to fabricate data for use in the inverse method. Co-simulation was attempted for all configurations of the power relationships shown in Table 4.7, namely Force-Force, Force-Velocity, Force-Position. The full barge-crane assembly was also exported as an FMU to validate the inverse method. For all validation scenarios using the JONSWAP wave spectrum, the same spectrum was specified to the hydrodynamic barge FMU. The wave was coming from the same direction for all configuration, namely from the left looking at the crane from the front.

5.4.1 Force-Force

The Z position for the FMU co-simulation and Fedem system simulation for the Force-Force coupling is compared in Figure 5.7. The sea-state was a JONSWAP wave spectrum with H_s of 0.2 and T_p of 2.8. This co-simulation was, as expected, not providing good correlations with the Fedem fabricated data for this interconnection configuration from the frequency coupling. Supporting theory can be found in Section 3.1. In the event of a co-simulation able to do internal iterations to each of the FMU components, decent simu-

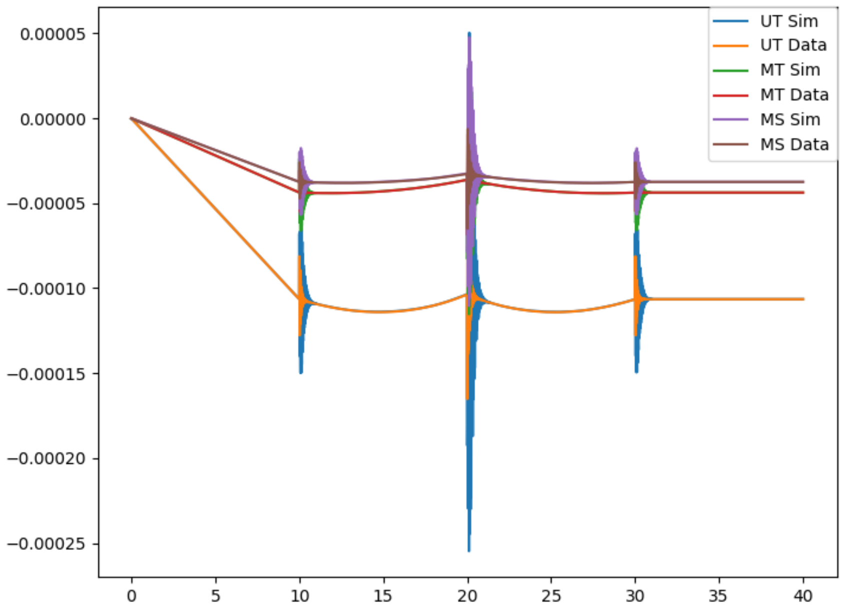


Figure 5.4: Crane, strains simulated vs strains fabricated in Fedem

lations may be possible. Internal iterations are, however, not possible for a co-simulation FMU exported from Fedem. This coupling is not discussed further due to its obvious flaws.

5.4.2 Force-Position

Due to the identical results of the force-velocity and force-position configurations seen in Figure 5.8, only the force-position configuration was inspected in detail. Figure 5.9 compares results from co-simulation and Fedem fabricated data, both in the same sea-state. No loads are applied in the simulation as showcasing undisturbed co-simulation accuracy was intended. If looking at the rotations about the X-axis in the figure, it appears to be additional damping in the simulation of the two decoupled models. A close-up of the damping effect can be seen in Figure 5.10a. The numerics causing this is beyond the scope of the thesis, but there will in many cases be loss of energy in co-simulation, as mentioned in Section 3.1. Results from applying the inverse method in co-simulation scenarios are discussed and illustrated in the remainder of this sub-section.

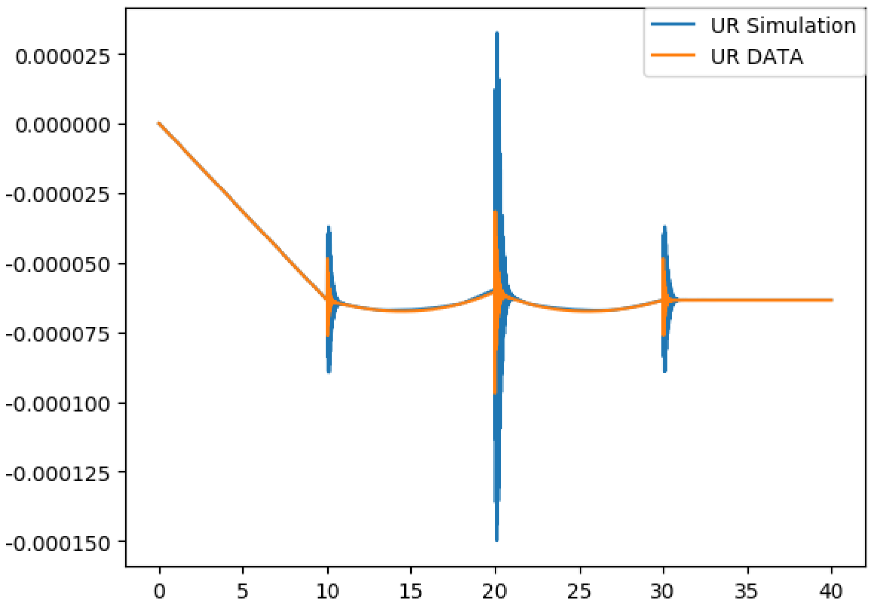


Figure 5.5: Crane, Independent strain gauge comparison

JONSWAP Wave Spectrum

The JONSWAP wave spectrum used for inverse method validation had a H_s of 0.2 and T_p of 4. This irregular wave sea-state was applied to challenge the inverse method and co-simulation with more sudden changes and load frequencies than in a regular sine wave. For this validation process, the actuators were moved simultaneously, and a 100N vertical force was added gradually to the outer end of the crane after 10 seconds.

The position of the co-simulation assembly is seen to have a satisfactory correlation with the simulated Fedem model, see Figure 5.11. There are, however, deviations, especially the rotation about the X-axis is seen to be shifted. Even when no loads are applied to the Fedem system model, there are phantom loads estimated by the inverse method from dynamic effects as the crane-barge system is floating in waves, as well as from energy loss during co-simulation, see Section 3.1. These loads are estimated in an oscillating manner, mostly in the negative Z and positive Y directions, giving moments about the X-axis, as seen in Figure 5.12. As a result, the co-simulated crane assembly is tilting slightly back and forth in the same frequency as the estimated phantom load. The misalignment is surprisingly significant considering how massive the crane is and the size of the estimated force. It should be noted that the application of a force reduces the oscillations in the estimated forces, leading to a more accurate load estimation.

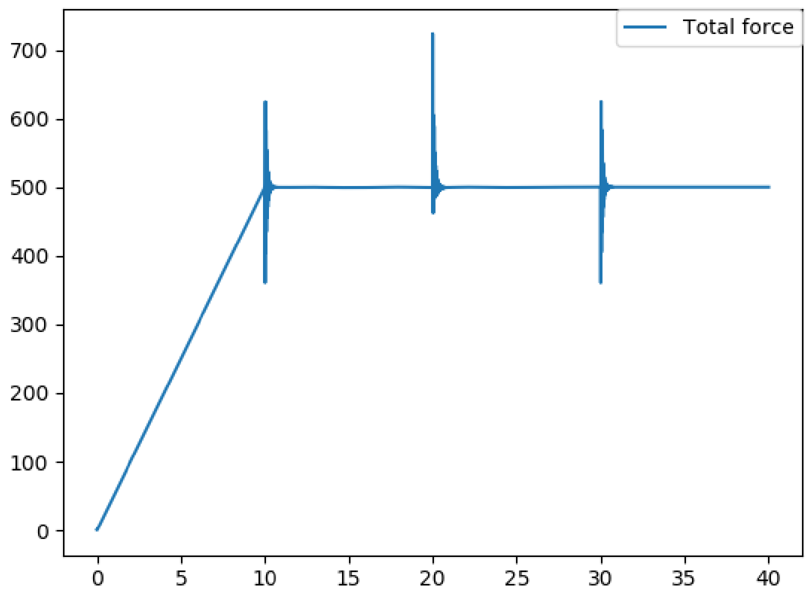


Figure 5.6: Crane, total estimated force in inverse method

The Fedem fabricated data and co-simulation results for the strain gauges used in the inverse method is seen to be fitting well. This is expected, as the inverse method applies force to the outer end of the upper crane arm to replicate these three strains. Like for the case where the crane alone is validated, there are large phantom loads present at the abrupt changes to the actuator movements. See Figure 5.13 for visualization.

The results for the independent strain gauges in Figure 5.14 are seen to have a surprisingly high correlation. The reduction in phantom-load after application of an actual force does surprisingly not seem to affect the accuracy of the independent strain gauges correlation. The co-simulation is generally seen to be over-estimating the fabricated strain. While one wants to replicate the strains as accurate as possible, an overestimation is preferred to an underestimation. An overestimation gives a slightly conservative estimation of the fatigue life of the asset. The relative error between the two can be inspected further in later simulations. If consistent, this overestimation may even be possible to compensate for.

Regular Sine Wave

The regular wave used for validation had a frequency of 0.1Hz and an amplitude of 0.1m. In this load-case, only a load is applied and no actuator movement.

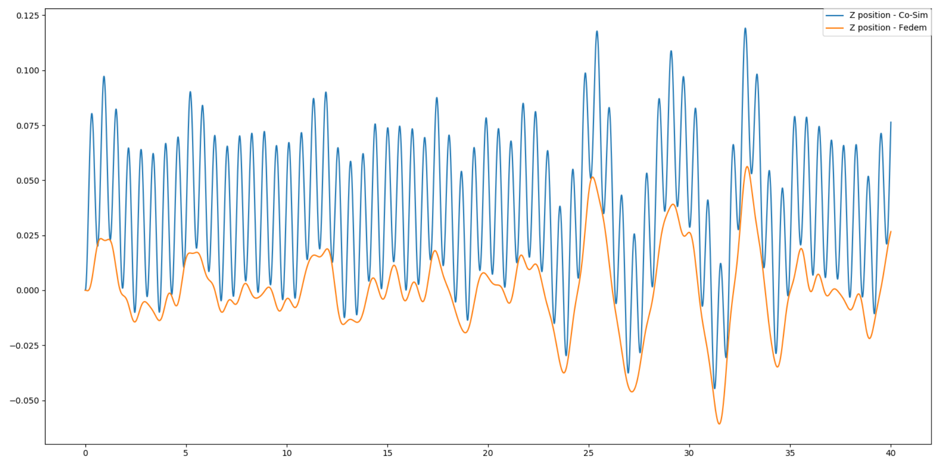


Figure 5.7: Z Position comparison for force-force coupled FMU co-simulation and Fedem assembly simulation

For the sea-state being a regular wave, the estimated phantom-loads were seen oscillate, but at much lower frequencies than for the irregular wave case, see Figure 5.18. The reduction is believed to stem from the lowered frequency of wave oscillations, reducing the dynamic strains for the crane. The phantom loads are, however, acting, and once again, the co-simulation assembly is seen to be tilting back and forth (Figure 5.15). When comparing the applied and estimated loads when a load is acting, the inverse method replicates the load very well; it can therefore be argued that the inverse method is replicating the loads significant to the structure.

The sine wave is seen to be too low frequent to cause significant oscillating strain gauge measurements after the initial damped strain oscillations for the independent strain gauges. The inverse method replicates the strains even better for this case than for the irregular sea-state, as no significant dynamic contribution to the strain gauge measurements are present. The strain gauges used in the inverse method and those used for independent comparison are shown in Figures 5.16 and 5.17.

5.4.3 Assembly FMU

The full crane barge system seen in Figure 4.14 was exported as an FMU for validating the inverse method. The hypothesis was that this would reproduce the most accurate forces and strains. Surprisingly this was not the case. The exact same simulation as for challenging the co-simulation with an irregular wave, was used as input to the inverse method for the

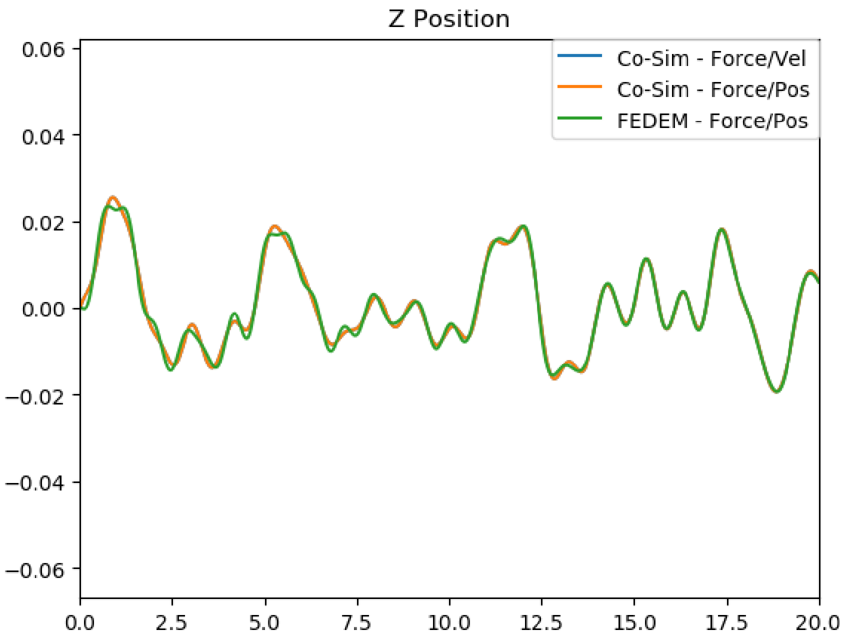


Figure 5.8: Z position comparison of the force-velocity and force-position coupling to the system simulation in Fedem

complete system assembly FMU. As can be seen in Figure 5.19, the independent strain gauges were less accurate for the full assembly as one inside an FMU. Time did not allow the deep-dive into the numerics causing this, as a working co-simulation was the main objective of the thesis.

5.5 Results From Real Data

The previous Section discusses and validates the inverse method for both the run of a single crane and a co-simulation barge-crane system with noise and error-free data. The final test is to validate the inverse method with real-life data. In this subsection, the results from physically testing the digital twin of the crane are presented and discussed.

5.5.1 "Phantom-Loads"

Even under static conditions with no applied load to the outer crane arm, the noise in the measured strain data results in estimated forces on the outer crane arm, referred to as "phantom loads". While not scientifically proven to be a direct effect of the low condition-

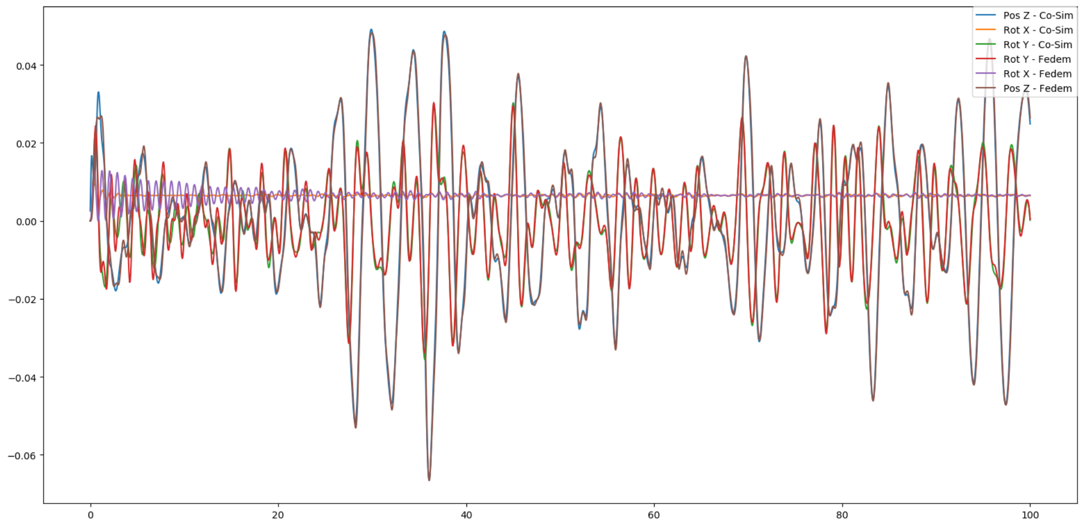


Figure 5.9: Force-position co-simulation over time, good correlation with Fedem fabricated data

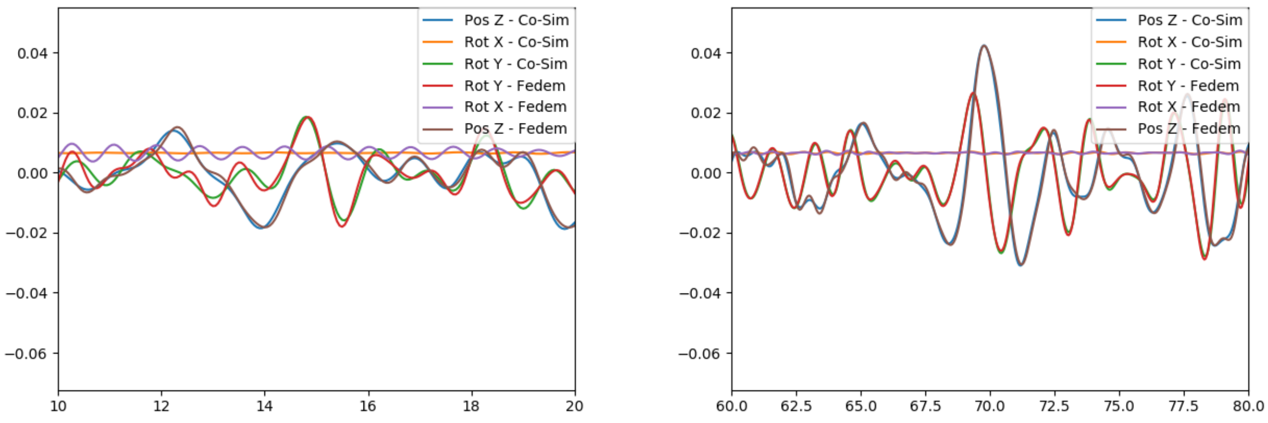
ing numbers of the stiffness matrix, the resulting phantom loads for the new configuration of the inverse method strain gauges were seen to be drastically reduced. The maximum phantom-load under static loading was only about 6N compared to the >20N in the author's project thesis. The static phantom-loads can be seen in Figure 5.20.

5.5.2 Digital Twin Simulation With Physical Crane

The physical and virtual crane were not running alongside each other as the Catman software used to record strain gauge and actuator values could not transmit live data. Instead, data sent from one .csv file to the digital twin over the TCP protocol, showcasing how the continuously run real-time digital twin could be configured. There is no delay in the simulation other than the time to transfer data and step the simulation forward in time as filtering and simulation are done sample by sample.

Static loading

The result from hanging a mass at the upper arm load attack point can be used to validate the Fedem model by comparing the magnitude of the estimated load to the applied load. The applied load of 8kg was seen to estimate a total load of about 88N seen in Figure 5.21. This the length of the complete estimated force vector in all three axis directions, also including the phantom loads in the X-direction where no load is applied. The estimated



(a) 10-20s closeup

(b) 60-80s closeup

Figure 5.10: Co-simulation compared to Fedem fabricated data, closeup

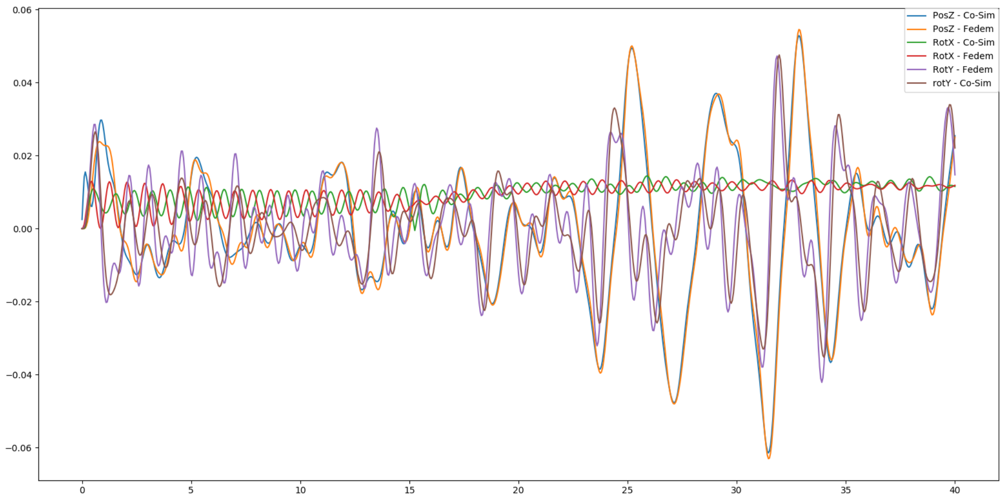
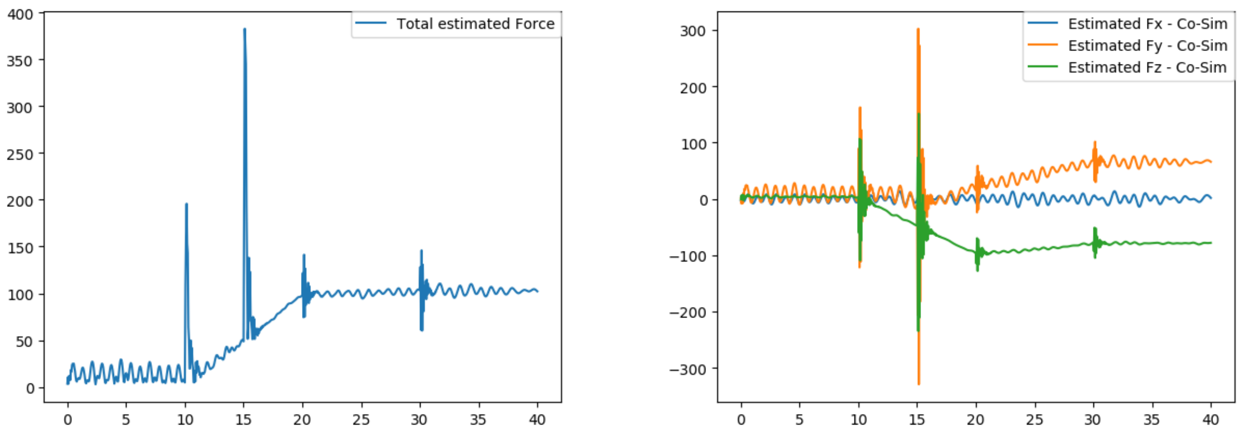


Figure 5.11: Force-position comparison with Fedem assembly simulation, JONSWAP wave spectrum with inverse method applied

load is seen to be bigger than that of the applied load by a couple of Newtons. This is considered to be rather satisfactory results.



(a) Total force

(b) Decomposed

Figure 5.12: Forces estimated from co-simulation in JONSWAP wave spectrum

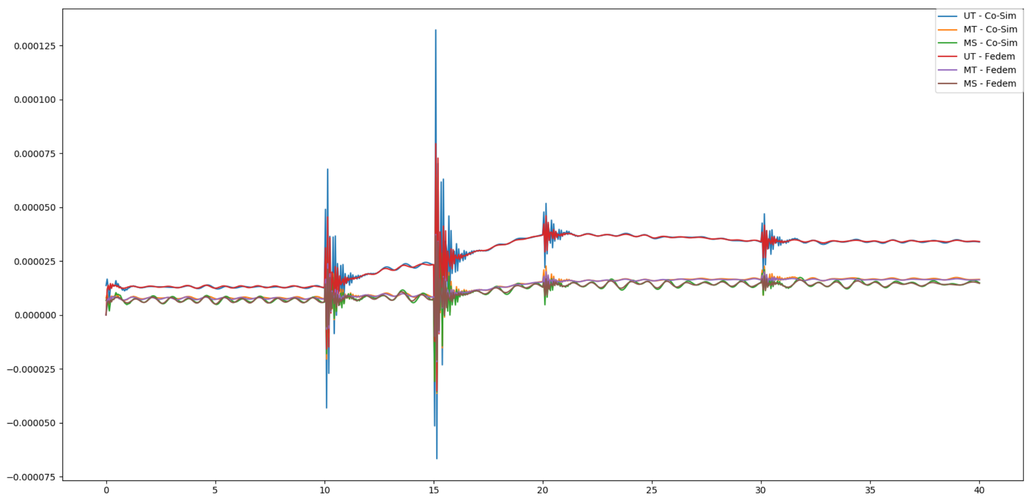


Figure 5.13: Force-position coupling, Comparison of strain gauges used in inverse method

With Load and All Actuators Run

As mentioned in Section 5.2, the disturbance effects are seen to vary significantly. For some runs, the inverse method strain gauges are seen to drift throughout the run, while in

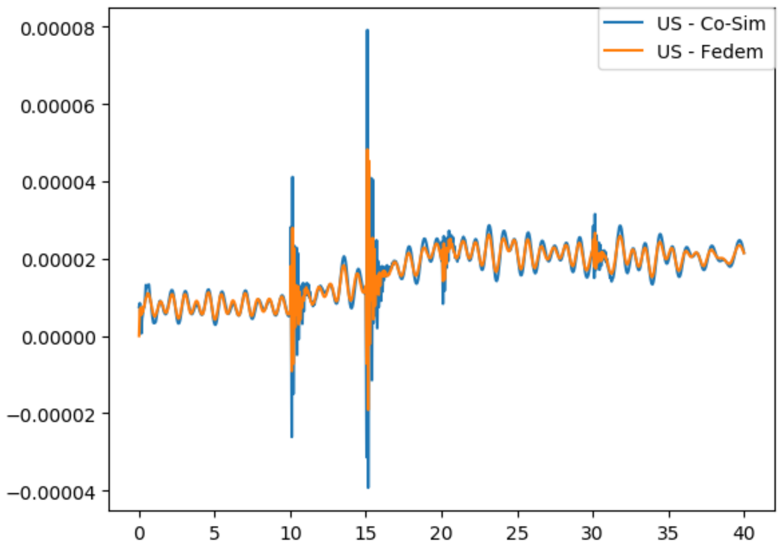


Figure 5.14: Force-position coupling, Independent strain gauge comparison

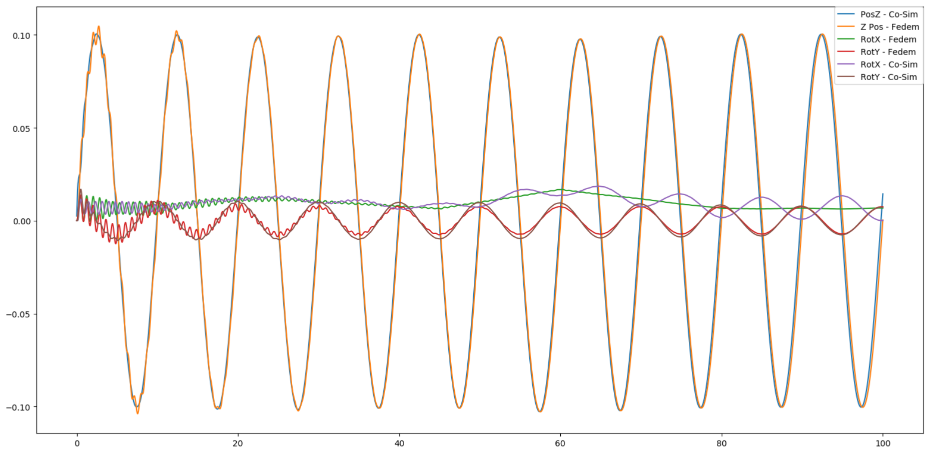


Figure 5.15: Force-position coupling, Position comparison in a regular wave

other runs only partly. Applying an 8kg mass to the outer end of the crane before moving the actuators is used as an example. As the inverse method applies loads to reproduce the

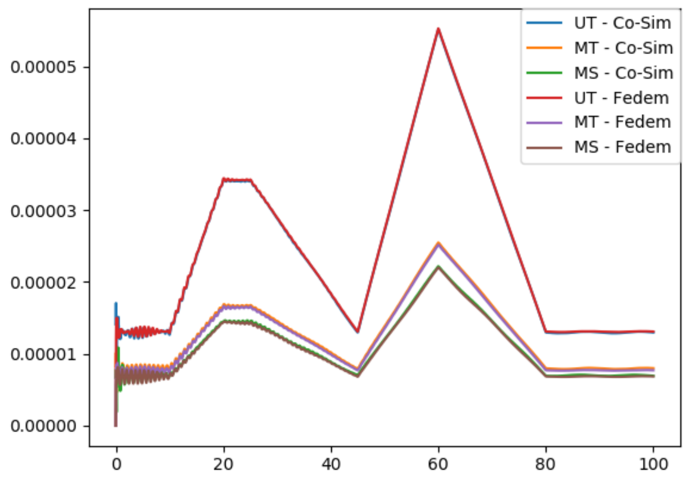


Figure 5.16: Force-position coupling, comparison of strains used in the inverse method

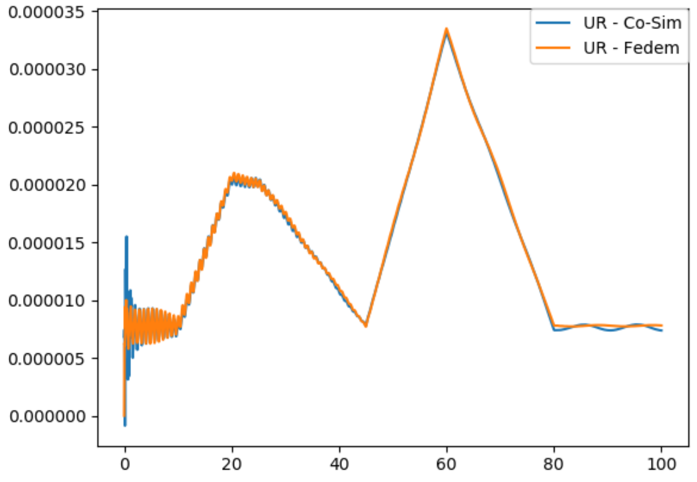


Figure 5.17: Force-position coupling, independent strain gauge comparison for simulation in regular wave

strains measured in the inverse method strain gauges, these will as good as always have a good fit with the real inverse strain gauges. Large drifts can be detected as the estimated loads become unrealistically large. The drifting can also be identified by comparing in-

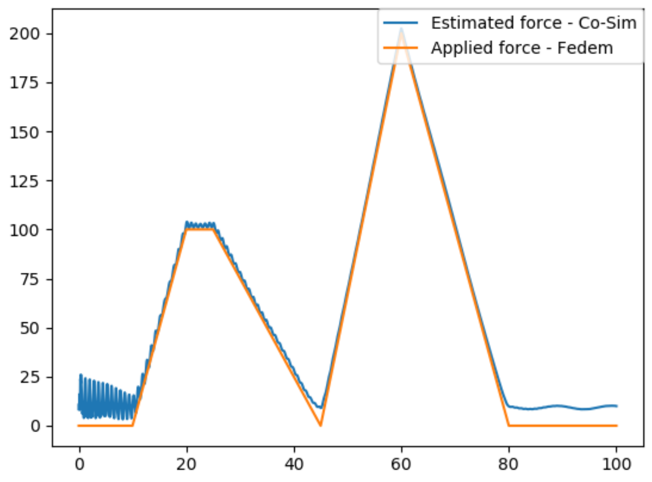


Figure 5.18: Force-position coupling, estimated vs applied load for simulation in regular wave

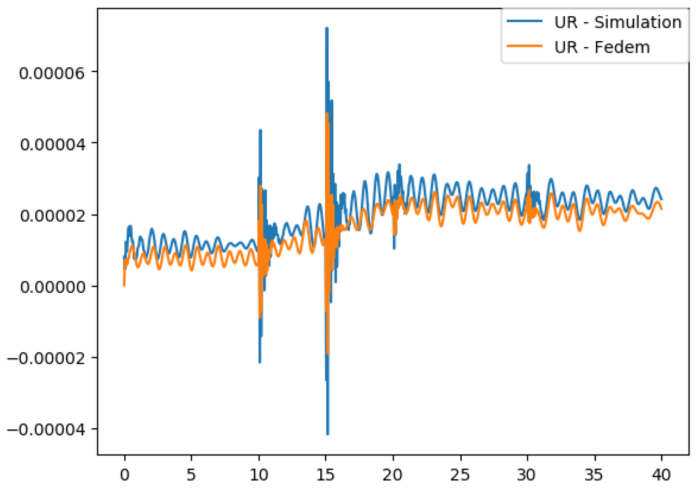
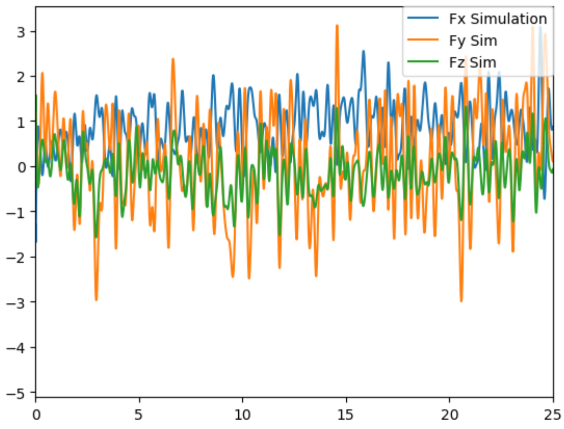
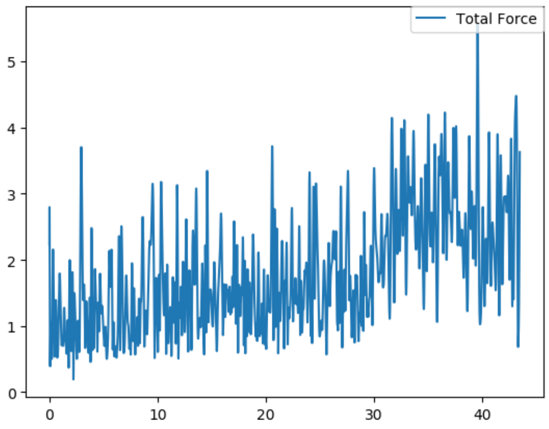


Figure 5.19: Assembly-FMU, independent strain gauge comparison

dependent strain gauges. If one has a seemingly stable physical independent strain gauge measurements and an accurate noise-free inverse mehthod, the effects of drifting can be seen comparing the two independent strain gauges. In Figure 5.22, the independent strain



(a) Decomposed



(b) Total Force

Figure 5.20: Forces estimated from static testing

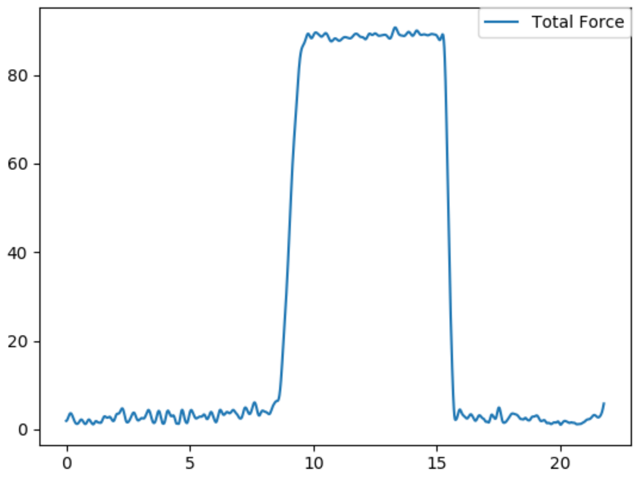


Figure 5.21: Force estimation from applying an 8kg mass

gauge values are estimated reasonably well in some regions, while drift is very apparent in others. Notice the "memory effect" after the run has ended, and the load removed after about 43 seconds. The strains are not returned to their original position. This illustrated independent strain gauge comparison is not the best, nor the worst measured but is a good illustration of the challenges with the instrumental setup as of today. This is untenable for a structurally valid digital twin.

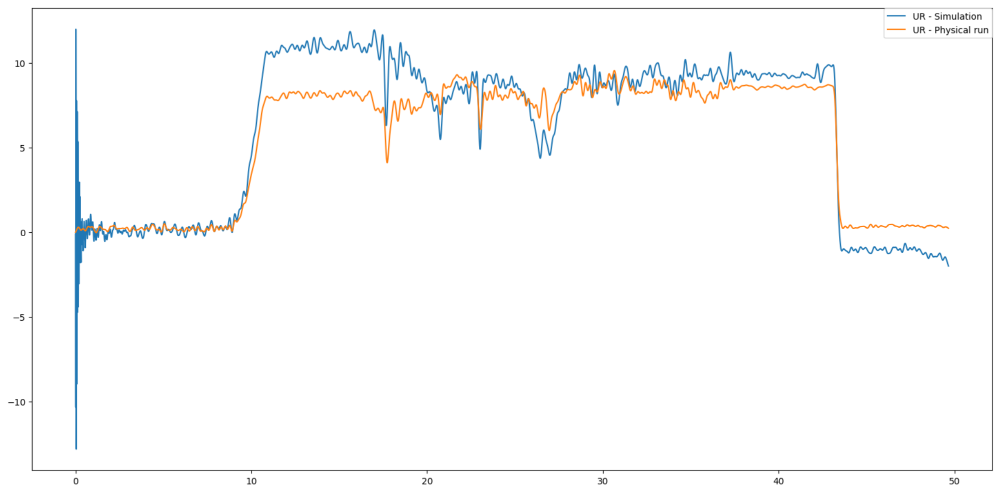


Figure 5.22: Independent strain gauge comparison for a run with an applied load and all actuators

The peaks in the estimated forces seen for abrupt actuator changes in Section 5.3 are not as evident in the run based on actual data, supporting the assumption of relatively low frequent loads and valid quasi-static assumptions.

Actuators Disabled and Manual Load Applied

Under the assumption that one can remove the devastating electrical system interference effects, the electrical noise left is that caused by the strain gauges themselves. This is a justifiable assumption to make, as these must be removed or reduced for the further use of the physical crane system. With no actuators enabled, the inverse method can be appropriately tested and validated. A manual frequently varied load was applied to the outer end of the upper crane arm. The load was mostly applied in the lifting plane of the arm to avoid errors from joint play.

In Figure 5.24, the strains for the independent strain gauges are compared, Figure 5.23 shows the estimated applied force. The correlation of the independent strain gauges for these type of tests are seen to give very satisfying results. The max absolute error from the physical data is seen to be in the regions of the error caused by static conditions phantom loads.

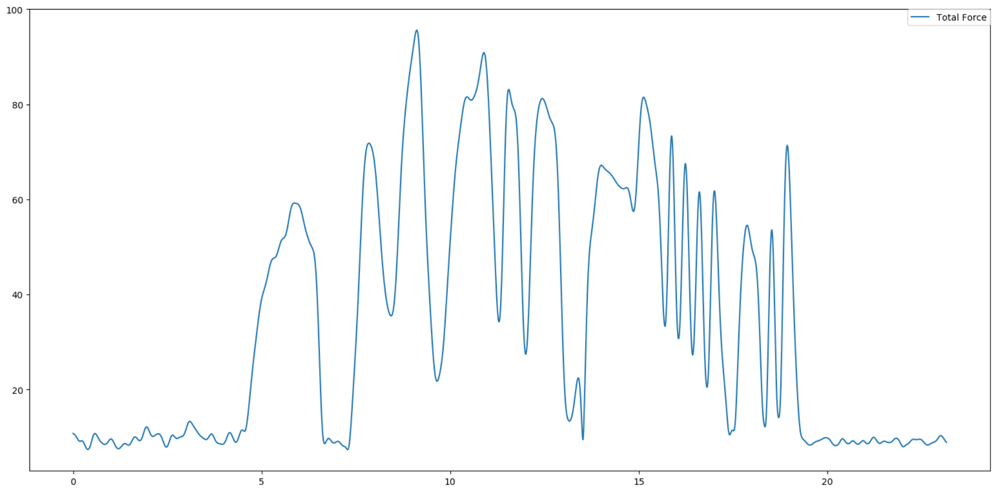


Figure 5.23: Total estimated force from manually loading the crane

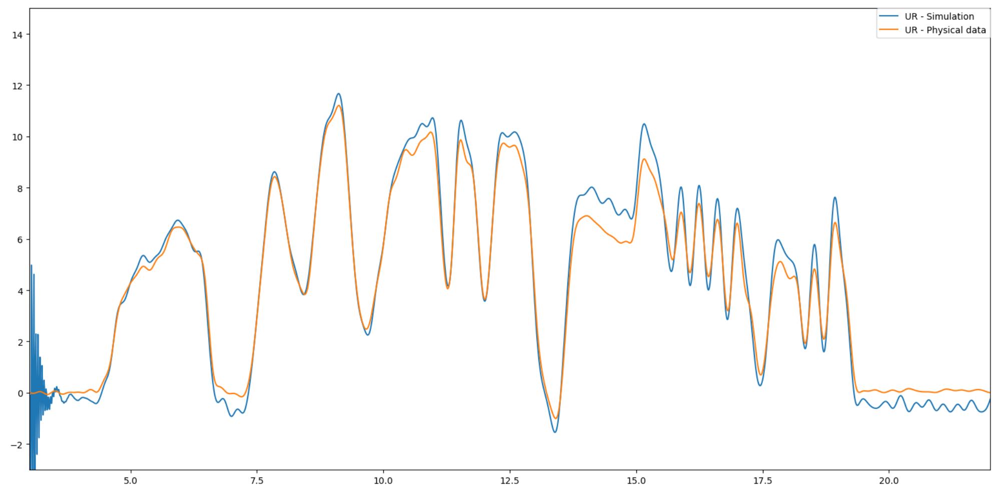
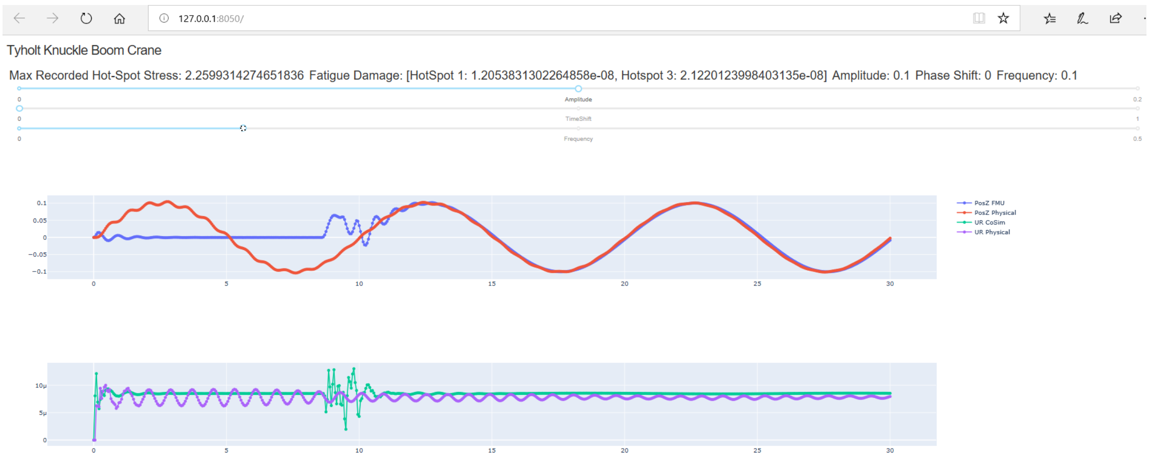


Figure 5.24: Simulated vs recorded independent strain gauges for the manual loading case

The phantom loads will never entirely be eliminated in the inverse method; thus, one has to determine what error in the results are small enough to be considered satisfactory. Once again, strains are seen to be slightly overestimated, giving a favorable conservative fatigue life estimation. It can be argued that the inverse is well formulated.

5.6 Cloud Based Monitoring system

As mentioned in Section 4.7, the alignment of the real-time co-simulation can be done by dragging the sliders to fit the simulated to the recorded Z positions. Figure 5.25 shows a typical web application layout after a simulation has been performed where alignment was attempted after about 8 seconds. Figure 5.26 shows a detailed view of the alignment during the simulation. The virtual and "physical" model is seen to be hydro-dynamically aligned after uses interaction, and the alignment method seems to work well for the sine wave.


Tyholt Knuckle Boom Crane

Max Recorded Hot-Spot Stress: 2.2599314274651836 Fatigue Damage: [HotSpot 1: 1.2053831302264858e-08, Hotspot 3: 2.1220123998403135e-08] Amplitude: 0.1 Phase Shift: 0 Frequency: 0.1

Figure 5.25: Alignment performed by dragging the sliders after about 5 seconds

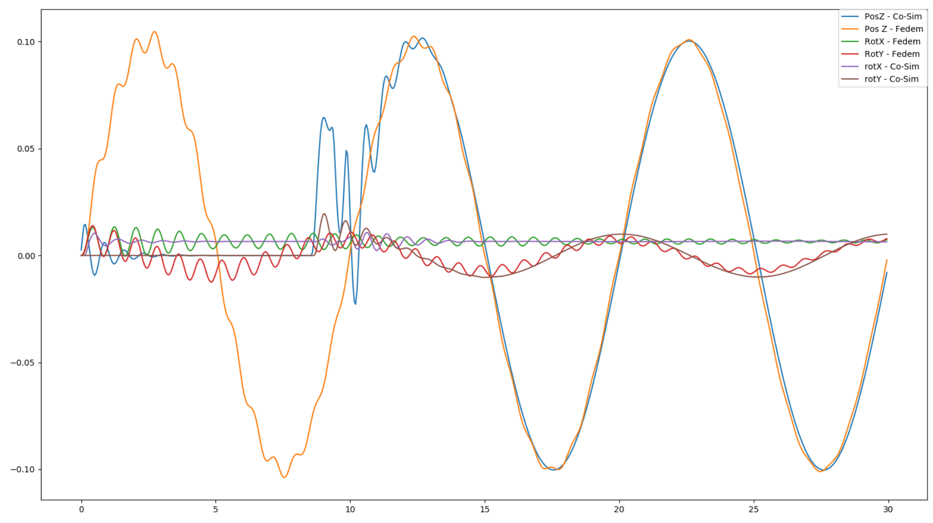


Figure 5.26: The slider alignment of Pos Z, Rot X and Rot Y, plotted from simulation

Chapter 6

Concluding Remarks

In this thesis, an inverse method based on finite element beam theory has been tested and validated for a standalone knuckle boom crane and a multi-domain co-simulation with a hydrodynamic barge. Simulation was performed using Fedem Functional Mock-Up Units in python. The decoupling of the crane and barge system to a co-simulation allows cross-platform simulation, an extremely powerful tool for system simulation. Physical data as well as data fabricated in a virtual environment has been used for validation. For the co-simulation, only virtually fabricated data is used.

The accuracy of the inverse method of the physical standalone knuckle boom crane was highly dependent on the environmental noise. Additionally, there are sources of error in the FE-model accuracy, wrongful data filtration, drifting and many more. The accuracy of the FE-model was tested by static loading cases as well as natural frequency tests. The most significant simplification was the rigid joints modeled. The fraction of the load and strain estimation error caused by these effects are hard to quantify. The inverse method concluded to be satisfactory accurate if reliable strain gauge readings are available. The improvements made to the inverse method since the author's project thesis (Hartmann (2019)) were seen to have huge impact and be essential for load estimation accuracy. In particular, conditioning inspection of the calculated inverse matrices. The quasi static assumptions were concluded to be valid for the standalone crane. In the current setup of the knuckle boom crane, additional efforts are needed to isolate the electrical noise originating from the actuators system to remove devastating unpredictable drifting effects.

The co-simulation of a structural and a hydrodynamic component was proven to replicate the overall behaviour of the two subsystems assembled inside Fedem. The most obvious deviation was additional damping effects in the co-simulation. While not significant to the overall behaviour of the system, the damping mismatch in the co-simulation and assembly systems will cause mismatch in simulated strains in the inverse method strain gauges, and hence estimated phantom loads. These phantom loads causes tilt in the co-simulation not present in the assembly simulation. Additionally, there is a varying degree of dynamic effects from the system moving in an ocean environment. What fraction of the phantom loads are coming from additional damping and dynamic motion has not been determined. These dynamic effects challenge the quasi-static assumptions of the inverse method. Despite the dynamic effects, the inverse method independent strain gauge comparisons are highly satisfying. The phantom loads are believed to encounter for "lost" strains to damping. Even though less dynamically accurate, this will provide accurate fatigue life estimations. Further testing and validation with physical data and additional simulation scenarios are needed to conclude on the final applicability of the method to a crane-barge co-simulation system.

The cloud based monitoring system set up for the crane-barge co-simulation digital twin has a simple graphical user interface, and is by no means a finalized monitoring application. The main outtake is the alignment procedure implemented to control the controllable sine wave sea-state of the virtual twin under operation. This method can be included in any future CBMS independent on the chosen back and front end construction. No irregular wave alignment method has been implemented, and thus no real-time digital twin of the crane-barge system in irregular waves is explained.

Chapter 7

Further work

Below is a few bullet-points suggesting focus areas for the further development of the digital twin described in the thesis.

- Compare co-simulation results with physical data. If representing the physical barge geometry by beam elements is inconvenient, the MSS toolbox could be used to import a model from Wamit or Ship X.
- Implement principal stress fatigue based calculation, as opposed to the current simple strains.
- Explore possible joint formulations in order to account for physical joint play.
- Isolate the electrical system of the strain gauges or reconfigure the actuators' electrical system, to remove the actuator induced electrical noise.
- Implement the digital twin to a CBMS or develop the CBMS in this thesis further.
- Research possible modifications to the inverse method including dynamic effects. This may reduce the estimated phantom loads under rough sea simulation.
- Implement an error estimation method calculating energy loss during co-simulation.

Bibliography

Christiansen, E. E., 2018. Digital twin of crane. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Engineering.com, April 2019. Ansys twin builder: The lowdown.
URL https://www.engineering.com/DesignSoftware/DesignSoftwareArticles/ArticleID/16974/ANSYS-Twin-Builder-The-Lowdown.aspx

Faltinsen, O., 1993. Sea Loads on Ships and Offshore Structures. International series of monographs on physics. Cambridge University Press.

FMI-standard, February 2019a. Fmi version 2.0.
URL https://fmi-standard.org/downloads/

FMI-standard, February 2019b. Functional mock-up interface for model exchange and co-simulation.
URL https://fmi-standard.org/docs/2.0.1-develop/

fmi standard.org, Desember 2019. Tools.
URL https://tml.jp/e/product/strain_gauge/wf_list.html

FMPy, April 2019. Fmpy - installation.
URL https://fmpy.readthedocs.io/en/latest/install/

Gyberg, F. A., 2017. Design, modeling and control of a generic crane for marine applications. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Hartmann, B., 2019. Digital twin modelling of offshore knuckle boom crane. Ph.D. thesis.

Haugen, B., 2017. TMM4250 Advanced Simulation: Dynamics topics.

Himanshu Mevadaa, D. P., May 2015. Experimental determination of structural damping of different materials.
URL https://core.ac.uk/download/pdf/82242060.pdf

HIVEMQ, Desember 2019. Mqtt topics best practices.
URL https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-bes

Johansen, C., 2019. Supervisor : Terje Rølvåg Digital Twin Of Knuckle Boom Crane (June).

Lab, T. M. I., April 2019. Wf series waterproof strain gauge.
URL https://tml.jp/e/product/strain_gauge/wf_list.html

Livieri, P., Lazzarin, P., 2005. Fatigue strength of steel and aluminium welded joints based on generalised stress intensity factors and local strain energy values. International Journal of Fracture 133 (3), 247–276.

LLC, P. N. A., April 2017. Global digital twin market 2017-2023.
URL https://search.proquest.com/docview/1938961009/A234837D5564692PQ/1?accountid=12870

Macdonald, K. A., Haagensen, P. J., 2009. Fatigue of welded aluminium hollow section profiles. Engineering Failure Analysis 16 (1), 254–261.
URL http://dx.doi.org/10.1016/j.engfailanal.2008.03.004

Moi, T., 2018. Inverse methods for digital twins using fedem vpm solver. Master's thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Moi, T., 2019. Digital Twin Based Structural Monitoring of a Knuckle Boom Crane. Ph.D. thesis.

Nicholas, J., 2016. Matrix Condition Numbers.

Niemi, E., Fricke, W., Maddox, S. J., 2018. The Structural Hot-Spot Stress Approach to Fatigue Analysis.

paho, Desember 2019. paho-mqtt 1.5.0.
URL https://pypi.org/project/paho-mqtt/

Perez, T., Smogelif, N., Fossenf, T. I., Sørensen, A. J., 2006. An overview of the marine systems simulator (MSS): A simulink® toolbox for marine control systems. Modeling, Identification and Control 27 (4), 259–275.

Sadjina, S., Kyllingstad, L. T., Rindarøy, M., Skjong, S., Esøy, V., Pedersen, E., 2019. Distributed co-simulation of maritime systems and operations. Journal of Offshore Mechanics and Arctic Engineering 141 (1).

Siemens, April 2019. Digital twin.
URL https://www.plm.automation.siemens.com/global/en/our-story/glossary/digital-twin/24465

Skjong, S., 2017. Modeling and Simulation of Maritime Systems and Operations for Virtual Prototyping using Co-Simulations. Vol. 2017.

Specialties, M., April 2019. Compact string pot - voltage divider output.
URL `https://docs-emea.rs-online.com/webdocs/142c/0900766b8142cde8.pdf`

Technology, F., 2018. Fedem Release 7.3 Theory Guide. Fedem Technology.

Thule, C., Broman, D., Larsen, P. G., Vangheluwe, H., 2017. Co-simulation : State of the art.

Viel, A., Minimes, P., 2014. Implementing stabilized co-simulation of strongly coupled systems using the Functional Mock-up Interface 2 . 0.

vpnMentor, April 2019. Tcp vs udp: Understanding the difference.
URL `https://www.vpnmentor.com/blog/tcp-vs-udp/`

Appendix A

Source code

A.1 Simple Co-Simulation of Crane and Barge

```

# This script plots simulated results in Fedem compared to those from a
    simple co-simulation setup with FMPy.

from fmpy import read_model_description, extract
from fmpy.fmi2 import FMU2Slave
from fmpy.util import plot_result, fmu_info
import os
import matplotlib.pyplot as plt
import numpy as np

PosZF = np.loadtxt(r'C:\Users\bjorn\OneDrive - NTNU\master\FEDEM\
    ComparisonFEDEMFMU\newSineWave\New Folder\
    G_10_C_36_Triad__146___Position_Z_vs_Time.asc', skiprows=6)
PosZF = [x[1] for x in PosZF]

rotXF = np.loadtxt(r'C:\Users\bjorn\OneDrive - NTNU\master\FEDEM\
    ComparisonFEDEMFMU\newSineWave\New Folder\
    G_10_C_47_Triad__146___Rotation_Angle_X_vs_Time.asc', skiprows=6)
rotXF = [x[1] for x in rotXF]

rotYF = np.loadtxt(r'C:\Users\bjorn\OneDrive - NTNU\master\FEDEM\
    ComparisonFEDEMFMU\newSineWave\New Folder\
    G_10_C_48_Triad__146___Rotation_Angle_Y_vs_Time.asc', skiprows=6)
rotYF = [x[1] for x in rotYF]


CUR_DIR = os.path.abspath(os.path.dirname(os.path.realpath(__file__)))
fmu_filenameFleet = r'C:\Users\bjorn\OneDrive - NTNU\master\FEDEM\FMU\
    fleetalpha42.fmu'  #fleetUpdated.fmu <- siste      #
    fleetalpha4_0_01pyFMI.fmu fungerer, fleetWaveInput.fmu
fmu_filenameCrane = r'C:\Users\bjorn\OneDrive - NTNU\master\FEDEM\FMU\
    Crane_alpha4origo0_05New.fmu'  #Crane_alpha4origo_0_01pyFMI.fmu
```

```
print("------FMU model info--------------------")
info = fmu_info(fmu_filenameFleet)
print(info) #prints model info
print("----------------------------------------")

model_descriptionFleet = read_model_description(fmu_filenameFleet)
model_descriptionCrane = read_model_description(fmu_filenameCrane)

unzipdirFleet = extract(fmu_filenameFleet)
unzipdirCrane = extract(fmu_filenameCrane)

Fleet = FMU2Slave(guid=model_descriptionFleet.guid,   #now you have your FMU for use in simulation
                    unzipDirectory=unzipdirFleet,
                    modelIdentifier=model_descriptionFleet.coSimulation.modelIdentifier,
                    instanceName='Fleet_FMU1')

Crane = FMU2Slave(guid=model_descriptionCrane.guid,   #now you have your FMU for use in simulation
                    unzipDirectory=unzipdirCrane,
                    modelIdentifier=model_descriptionCrane.coSimulation.modelIdentifier,
                    instanceName='Crane_FMU1')


fleetVrs = {} #It can be good to store variable names and values in a dict for later use.
craneVrs = {}
for variable in model_descriptionFleet.modelVariables: #dict with variable name as key and number as value. All start with Input_ and output with Output_
   fleetVrs[variable.name] = variable.valueReference
for variable in model_descriptionCrane.modelVariables:
    craneVrs[variable.name] = variable.valueReference

#comment everything below if you just want vrs ////////////////////////////////////////////////////////////////////////////////////////////


print("------VariableDict-------------------")
print(fleetVrs)
print(craneVrs)
print("----------------------------------------")


Fleet.instantiate()
Fleet.enterInitializationMode()
Fleet.exitInitializationMode()

Crane.instantiate()
Crane.enterInitializationMode()
Crane.exitInitializationMode()
#now you can do whatever you want with this FMUs

fleetInputRefVals = [0, 1, 2, 3, 4, 5] #inFx, inFy, inFz, inFRx, inFRy, inFRz
```

```
fleetOutputRefVals = [11, 12, 13, 14, 15, 16] #outPosX, outPosY, outPosZ,
    outRotX, outRotY, outRotZ

craneInputRefVals = [6, 7, 8, 9, 10, 11] #inPosX, inPosY, inPosZ, inRotX,
    inRotY, inRotZ
craneOutputRefVals = [18, 19, 20, 21, 22, 23]  #outFx, outFy, outFz,
    outFRx, outFRy, outFRz



start_time = 0.0
stop_time = 100.0
step_size = 0.05 # As far as i know, this is often constant for an fmu.
time = 0
timeList = []
PosZ = []
rotXList = []
rotYList = []
while time <= stop_time:
    timeList.append(time)
    #get outputs of both FMUs
    fleetOutputs = Fleet.getReal(fleetOutputRefVals)
    craneOutputs = Crane.getReal(craneOutputRefVals)

    #set inputs of both FMUs
    Fleet.setReal(fleetInputRefVals, craneOutputs)
    Crane.setReal(craneInputRefVals, fleetOutputs)

    Fleet.doStep(currentCommunicationPoint=time, communicationStepSize=
    step_size) #Do step
    Crane.doStep(currentCommunicationPoint=time, communicationStepSize=
    step_size)
    PosZ.append(Fleet.getReal([fleetOutputRefVals[2]]))
    rotX = Crane.getReal([craneVrs["Input_inRotX"]])[0]
    rotY = Crane.getReal([craneVrs["Input_inRotY"]])[0]
    rotXList.append(rotX)
    rotYList.append(rotY)
    time += step_size

plt.plot(timeList, PosZ, label="Pos Z - Co-Sim")
plt.plot(timeList, rotXList, label="Rot X - Co-Sim")
plt.plot(timeList, rotYList, label="Rot Y - Co-Sim")
plt.plot(timeList, rotYF, label="Rot Y - Fedem")# [0:-1]
plt.plot(timeList, rotXF, label="Rot X - Fedem")
plt.plot(timeList, PosZF, label="Pos Z - Fedem")
plt.legend(loc='upper right', borderaxespad=0.)
plt.show()


plt.plot(timeList, PosZ, label="Pos Z - Co-Sim")
plt.plot(timeList, rotXList, label="Rot X - Co-Sim")
plt.plot(timeList, rotYList, label="Rot Y - Co-Sim")
plt.plot(timeList, rotYF, label="Rot Y - Fedem")# [0:-1]
plt.plot(timeList, rotXF, label="Rot X - Fedem")
plt.plot(timeList, PosZF, label="Pos Z - Fedem")
plt.legend(loc='upper right', borderaxespad=0.)
plt.xlim([10, 20])
plt.show()

plt.plot(timeList, PosZ, label="Pos Z - Co-Sim")
plt.plot(timeList, rotXList, label="Rot X - Co-Sim")
plt.plot(timeList, rotYList, label="Rot Y - Co-Sim")
plt.plot(timeList, rotYF, label="Rot Y - Fedem")# [0:-1]
plt.plot(timeList, rotXF, label="Rot X - Fedem")
plt.plot(timeList, PosZF, label="Pos Z - Fedem")
plt.legend(loc='upper right', borderaxespad=0.)
plt.xlim([60, 80])
plt.show()
```

A.2 Fatigue Calculation - MQTT Included

```
#All implemented fatigue calculations can be seen in this script. The MQTT
    connection to the paralell running co-simulation is also included



import os
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt, find_peaks
import rainflow
import csv
import time as t
import json
import paho.mqtt.client as paho
import os
CUR_DIR = os.path.abspath(os.path.dirname(os.path.realpath(__file__)))
import pandas as pd

# #Connect to MQTT
broker='broker.hivemq.com'
port=1883
client= paho.Client("control1")                          #create client
    object
client.connect(broker, port) #establish connection
client.loop_start()

USfedem = np.loadtxt(r'C:\Users\bjorn\OneDrive - NTNU\master\FEDEM\
    ComparisonFEDEMFMU\New40SekHydro\
    G_13_C_53_Strain_rosette__15__US_strain_gauge__Gage_1__Strain_vs_Time.
    asc', skiprows=6)
US = np.array([x[1]*10**6 for x in USfedem])
#t = np.array(x[0] for x in USfedem)

def doPeakValleyExtraction(array):
    array= np.array(array)
    peaks, _ = find_peaks(array)
    arrayInv = array*-1
    valleys, _ = find_peaks(arrayInv)
    both = np.concatenate((peaks, valleys))
    both = np.sort(both)
```

```python
    return array[both]

def calculateDamage(extractedCycles, SNcurve):
    fractions = []
    for low, high, multi in extractedCycles:
        # find mean, amplitude and number of cycles
        S_m = (low + high) * 0.5
        R = high - low
        S_a = (abs(high) - abs(low)) / 2
        ni = multi
    # correct for mean stress using Goodman correction
        S_eff = abs(S_a * (S_u / (S_u - S_m)))
        Ni = SNcurve(R)
        if high >= S_y:
            Yield = True
        if Ni < 10**4:
            send_warning = {}
            send_warning["Number of cycles"] = Ni
            send_warning["warning"] ="low numbers of cycles to failure, this may make high cycle fatigue assumption invalid"
            warning = json.dumps(send_warning)
            client.publish("Warning", warning, qos=0)
        fraction_to_add = ni / Ni
        fractions.append(fraction_to_add)
    totDamage = sum(fractions)
    return totDamage

def FAT40(stressRange):
    return (stressRange/8751)**(1/-0.39)

def standardSN(stressRange):
    return (stressRange)

S_u = 50 * 10**6 #ultimate tensile strength
S_y =  47* 10**6 # yield strength
time = 0
a = 0
E = 70e09
fractions = []
hotspots = ["HotSpot 1", "HotSpot 3"]
sendDict = {}
while True: #while true makes the loop run until interrupted
    #set start time
    if time == 0 and a == 0:
        initalClockTime = t.time()
    maxStress = 0
    # run damage computation every 5 seconds
    if (time + initalClockTime - t.time() <= 1):
        strainList = [0,0,0]
        data = []
        strains = pd.read_csv(r"C:\Users\bjorn\OneDrive - NTNU\master\python\simulatedStresses.csv", skiprows=0, header=None)
        strains = np.array(strains)
        for i in range(len(strains[0])):
            strainList[i] = [item[i] for item in strains]
        strainList = np.array(strainList)
        strainList = np.array([strainList[0], np.add((strainList[1].dot
    (1.5)), ((strainList[2]).dot(0.5)))])
        strainList = strainList.dot(10**-6)

        #strains = US
        print(strains)
        for hotspot in strainList:
            hotspot = hotspot.astype(float)
            peaksValleys = doPeakValleyExtraction(hotspot)
            extractedCycles = rainflow.extract_cycles(peaksValleys)  #
    returns, low, high, multiple
            dataentry = calculateDamage(extractedCycles, FAT40)
            data.append(dataentry)
            maxStress = float(max(maxStress, max(hotspot)))
            print("max: ", maxStress)
        print(data)
        for i in range(len(data)):
            print(hotspots[i])
            sendDict[hotspots[i]] = data[i]
        sendDict["maxStress"] = maxStress
        send = json.dumps(sendDict)
        print(send)
        client.publish("Damage1", send, qos=0)

        time += 5




#run_damage_calculation()
```

A.3 Fast Fourier Transform

```
#This script shows the simple FFT analysis performed to obtain the
    rayleigh damping coefficiets of the crane.


import pandas as pd
import matplotlib.pyplot as plt
from scipy.fftpack import fft, fftfreq
from scipy.signal import butter, filtfilt, welch
import numpy as np

data = pd.read_csv("endPulley_pos6_9_47Hz_hitpointOuterEnd.csv", skiprows
    =21, header=None)
print(type(data))
y1 = np.array(data[3])
print(type(y1))
fft1 = data[7]
print(y1)

y1 = y1 - np.mean(y1)
fs = 5e03
```

```
f, Pxx = welch(y1, fs, nperseg=1024)


#plt.subplot(3,1,1)
plt.plot(y1, label="Vibrations after impact")
plt.legend(loc='upper right', borderaxespad=0.)
plt.show()


plt.plot(f, Pxx)
plt.xlim([0, 100])
plt.xlabel('frequency [Hz]')
plt.ylabel('Amplitude')
plt.legend(loc='upper right', borderaxespad=0.)
plt.show()


#FFT
N = len(y1)
T = 200000 * (10**-9)
x = np.linspace(0, N*T, N)
y = y1


yf = fft(y, x.size)

nyq = 2500 # 0.5*fs, where fs=5000/s
low25 = 24.95/nyq
high25 = 25.05/nyq
low49_95 = 49.95/nyq
high50_45 = 50.05/nyq
B1, B2 = butter(1, [low25, high25], btype='bandpass', analog=False)
B3, B4 = butter(1, [low49_95, high50_45], btype='bandpass', analog=False)

filtered25 = filtfilt(B1, B2, y)
filtered49 = filtfilt(B3, B4, y)

#plt.subplot(2,2,1)

#plt.subplot(2,2,2)
plt.plot(filtered49, label="Oscillations at 49.2Hz")
plt.legend(loc='upper right', borderaxespad=0.)
plt.show()

plt.plot(filtered25, label="Oscillations at 24.9Hz")
plt.legend(loc='upper right', borderaxespad=0.)
plt.show()
```

A.4 Interpolation Operations

```
#This script shows the functions written to perform matrix and vector
    interpolations. These were used in the real time simulation to obtain
    position-specific stiffness matrices and rotation compensation vectors
    .

from math import sqrt
import numpy as np
import matplotlib.pyplot as plt
import csv
from scipy.interpolate import RegularGridInterpolator

def interpolator2D(x, y, z, act1, act2):
    I = RegularGridInterpolator((x, y), z)
    return I([act1, act2])


def interpolator4D(a, b, c, d, z, act1, act2, rotX, rotY):
    I = RegularGridInterpolator((a, b, c, d), z)
    return I([act1, act2, rotX, rotY])



def interpolateMatrix(x, y, SMs, act1, act2):
    M = np.zeros((3,3))
    index = 0
    for i in range(len(M)):
        for j in range(len(M)):
            M[i][j] = interpolator2D(x,y, SMs[i][j], act1, act2)
            index += 1
    return M

def interpolateVecs(x, y, VecsFleet, act1, act2):
    Vec = np.zeros(3)
    for i in range(len(VecsFleet)):
        Vec[i] = interpolator2D(x, y, VecsFleet[i], act1, act2)
    return Vec

def interpolateVecs4D(a, b, c, d, Vecs, act1, act2, rotX, rotY):
    Vec = np.zeros(3)
    for i in range(len(Vecs)):
        Vec[i] = interpolator4D(a, b, c, d, Vecs[i], act1, act2, rotX,
    rotY)
    return Vec
```

A.5 Web App

```
#This script is the web app of the monitoring system.


import paho.mqtt.client as paho
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly
from dash.dependencies import Input, Output
import json
```

```python
import numpy as np

recvMessage = [[0.0], [0.0], [0.0]]
recvList = [[0], [0]]
sendDict = {}
strains = [[0], [0], [0], [0]]
timeList = [0.0]
posZF = [0]
posZ = [0]
actURList = [0]
def on_message(client, userdata, msg):
    topic = msg.topic
    message = json.loads(msg.payload)
    #print(message)
    if topic == "Damage1":
        recvMessage[0] = message.get("HotSpot 1")
        recvMessage[1] = message.get("HotSpot 3")
        recvMessage[2] = message.get("maxStress")
    if topic == "Stress":
        recvList[1].append(message.get("Stress"))
        recvList[0].append(recvList[0][-1] + 0.05)
    if topic == "Strains":
        strains[0].append(message.get("UT"))
        strains[1].append(message.get("REF"))
        strains[2].append(message.get("US"))
        strains[3].append(message.get("UR"))
        posZF.append(message.get("PosZactual"))
        posZ.append(message.get("PosZ"))
        actURList.append(message.get("actUR"))
        timeList.append(float(timeList[-1]) + 0.05)
    #print(recvMessage)
    return

broker='broker.hivemq.com'
port=1883
client= paho.Client()                          #create client object
client.on_message = on_message
print("connecting to broker: ", broker)
client.connect(broker, port) #establish connection
client.subscribe("Damage1", qos=2)
client.subscribe("Stress", qos=2)
client.subscribe("Strains", qos=2)
client.loop_start()



external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
app.layout = html.Div(
    html.Div([
        html.H4('Tyholt Knuckle Boom Crane'),
        html.Div(id='live-update-text'),
        dcc.Interval(
            id='interval-component',
            interval=1*1000, # in milliseconds
```

```
                n_intervals=0
            ),
            dcc.Slider(
                id='slider1',
                min=0,
                max=0.2,
                step=0.01,
                value=0,
                marks={0:"0", 0.1: "Amplitude", 0.2:"0.2"},
                #style={}
            ),
            dcc.Slider(
                id='slider2',
                min=0,
                max=1,
                step=0.01,
                value=0,
                marks={0:"0", 0.5: "TimeShift", 1:"1"},
                #style={}
            ),
            dcc.Slider(
                id='slider3',
                min=0,
                max=0.5,
                step=0.01,
                value=0,
                marks={0:"0", 0.25: "Frequency", 0.5:"0.5"},
                #style={}
            ),
            dcc.Graph(id='live-update-graph'),
        ])
)




@app.callback(
                Output('live-update-text', 'children'),
                [Input('slider1', 'value'), Input('slider2', 'value'), Input
    ('slider3', 'value')]) #, Input('slider2', 'value'), Input('slider3',
    'value')

# def update_output(value):
#     return 'You have selected "{}"'.format(value)

def update_metrics(value1, value2, value3):
    style = {'padding': '5px', 'fontSize': '24px'}
    sendDict["Amplitude"] = value1
    sendDict["TimeShift"] = value2
    sendDict["Frequency"] = value3
    data = json.dumps(sendDict)
    client.publish("testAmplitude", data, qos=2)
    return [
        html.Span('Max Recorded Hot-Spot Stress: {0}'.format(recvMessage
    [2]), style=style),
        html.Span('Fatigue Damage: [HotSpot 1: {0}, Hotspot 3: {1}]'.
    format(recvMessage[0], recvMessage[1]), style=style),
        html.Span('Amplitude: {0}'.format(value1), style=style),
        html.Span('Phase Shift: {0}'.format(value2), style=style),
        html.Span('Frequency: {0}'.format(value3), style=style),
    ]


# Multiple components can update everytime interval gets fired.
@app.callback(Output('live-update-graph', 'figure', ),
              [Input('interval-component', 'n_intervals')])

def update_graph_live(n):
    fig = plotly.subplots.make_subplots(rows=3, cols=1, vertical_spacing
    =0.2)
    fig.update_layout(
        autosize=False,
        width=2000,
        height=1000,
        )
    fig.append_trace({
        'x': timeList,
        'y': posZ,
        'text': "PosZ FMU",
        'name': 'PosZ FMU',
        'mode': 'lines+markers',
        'type': 'scatter'
    }, 1, 1)
    fig.append_trace({
        'x': timeList,
        'y': posZF,
        'text': "PosZ Physical",
        'name': 'PosZ Physical',
        'mode': 'lines+markers',
        'type': 'scatter'
    }, 1, 1)
    fig.append_trace({
        'x': timeList,
        'y': strains[3],
        'text': "UR CoSim",
        'name': "UR CoSim",
        'mode': 'lines+markers',
        'type': 'scatter'
    }, 2, 1)
    fig.append_trace({
        'x': timeList,
        'y': actURList,
        'text': "UR Physical",
        'name': "UR Physical",
        'mode': 'lines+markers',
        'type': 'scatter'
    }, 2, 1)


    return fig

```

```
if __name__ == '__main__':
    app.run_server(debug=True)
```

A.6 Stiffness Matrices and Rotation Compensation Vectors generation

```

# This code takes a functional mock-up unit(FMU) representing a FEDEM FE-
    crane model and generates a stiffness matrix for the system.
#The procedure is done for every possible combination of the interpolation
     design points.



from fmpy import read_model_description, extract
from fmpy.fmi2 import FMU2Slave
from fmpy.util import plot_result, fmu_info
import numpy as np
import os
import itertools
import csv
CUR_DIR = os.path.abspath(os.path.dirname(os.path.realpath(__file__)))
def stiffnessandrotationMatrices(show_plot=True):
    q = open(CUR_DIR + "\\rotationVecsFleet.csv", "w")
    q.close()
    start_time = 0.0
    stop_time = 11.0
    step_size = 5e-2

    fmu_filename = r'C:\Users\bjorn\OneDrive - NTNU\master\FEDEM\FMU\
    Crane_alpha4origo0_05.fmu'
    model_description = read_model_description(fmu_filename)

    info = fmu_info(fmu_filename)
    print(info) #prints model info for convenience

    # make a dictionary with all variable names stored
    vrs = {}
    inputVrs = {}
    outputVrs = {}
    outputStrainVrs = {}

    for variable in model_description.modelVariables:
        if "Input" in variable.name:
            inputVrs[variable.name] = variable.valueReference
        elif "Output" in variable.name:
            outputVrs[variable.name] = variable.valueReference
            if "inv" in variable.name:
                outputStrainVrs[variable.name] = variable.valueReference

        vrs[variable.name] = variable.valueReference
        print(variable.name)
        print(variable.valueReference)
    combinations = list(itertools.combinations(outputStrainVrs, 3))
    print(combinations)
    # All inputs numbers to the FMU that will later become applied forces
    actLower = vrs['Input_Actuator_Translation_Lower']
    actUpper = vrs['Input_Actuator_Translation_Upper']
    actRot = vrs['Input_Rotational_input']
    XrotBase = vrs['Input_inRotX']
    YrotBase = vrs['Input_inRotY']

    FMU_Input_list = [actLower, actUpper, actRot]


    # extract the FMU
    unzipdir = extract(fmu_filename)
    print('\n', unzipdir, '\n')
    fmu = FMU2Slave(guid=model_description.guid,
                    unzipDirectory=unzipdir,
                    modelIdentifier=model_description.coSimulation.
    modelIdentifier,
                    instanceName='Crane_FMU1')

    #initialize

    fmu.instantiate()
    fmu.setupExperiment(startTime=start_time)
    fmu.enterInitializationMode()
    fmu.exitInitializationMode()

    #for FMU_load in FMU_Input_list simulation loop:
    direction_list = [' Rotation', ' Lower Actuator', ' Upper Actuator']
     #Load direction used for labeling of graphs



    for item in combinations:
        FMU_output_list = [outputStrainVrs[item[0]], outputStrainVrs[item
    [1]], outputStrainVrs[item[2]]]

#interpolation design points
        actPosLower = [-0.02, 0.03, 0.08, 0.12]
        actPosUpper = [-0.02, 0.03, 0.08, 0.12]
        rotX = [-0.06, 0.0, 0.06]
        rotY = [-0.06, 0.0, 0.06]

        actPosIncrement = 0.001
        rotationVecs = []
        rotationStrain = np.zeros((3, len(actPosLower), len(actPosUpper),
    len(rotX), len(rotY)))
        for i in range(len(actPosLower)):
            NewLowerAct = actPosLower[i]
            for j in range(len(actPosUpper)):
                NewUpperAct = actPosUpper[j]
                for k in range(len(rotX)):
                    NewRotX = rotX[k]
                    for l in range(len(rotY)):
                        NewRotY = rotY[l]
            time = start_time


            while fmu.getReal([actLower])[0] <= NewLowerAct - 0.00001:  # bigger than noise.
                sign = (NewLowerAct - fmu.getReal([actLower])[0])/abs(NewLowerAct - fmu.getReal([actLower])[0])
                # DO STEP
                fmu.setReal([actLower], [fmu.getReal([actLower])[0] + (actPosIncrement*sign)])  # SETS VALUE FOR GIVEN INPUT
                fmu.doStep(currentCommunicationPoint=time, communicationStepSize=step_size)
                time += step_size


            while abs(NewUpperAct - fmu.getReal([actUpper])[0]) >=  0.00001:
                sign = (NewUpperAct - fmu.getReal([actUpper])[0]) / abs(NewUpperAct - fmu.getReal([actUpper])[0])
                fmu.setReal([actUpper], [fmu.getReal([actUpper])[0] + (actPosIncrement*sign)])  # SETS VALUE FOR GIVEN INPUT
                fmu.doStep(currentCommunicationPoint=time, communicationStepSize=step_size)  # DO STEP
                time += step_size

            while abs(NewRotX - fmu.getReal([XrotBase])[0]) >=  0.00001:
                sign = (NewRotX - fmu.getReal([XrotBase])[0]) / abs(NewRotX - fmu.getReal([XrotBase])[0])
                fmu.setReal([XrotBase], [fmu.getReal([XrotBase])[0] + (actPosIncrement*sign)])  # SETS VALUE FOR GIVEN INPUT
                fmu.doStep(currentCommunicationPoint=time, communicationStepSize=step_size)  # DO STEP
                time += step_size

            while abs(NewRotY - fmu.getReal([YrotBase])[0]) >= 0.00001:
                sign = (NewRotY - fmu.getReal([YrotBase])[0]) / abs(NewRotY - fmu.getReal([YrotBase])[0])
                fmu.setReal([YrotBase], [
                    fmu.getReal([YrotBase])[0] + (actPosIncrement * sign)])  # SETS VALUE FOR GIVEN INPUT
                fmu.doStep(currentCommunicationPoint=time, communicationStepSize=step_size)  # DO STEP
                time += step_size


            wait = time + 3 # let the simulation stabilize, seen to be sufficient after 3 seconds.
            while time <= wait:
                fmu.doStep(currentCommunicationPoint=time, communicationStepSize=step_size)
                time+=step_size
            substractStrains = {}
            for strainOutput in outputStrainVrs.keys():
                substractStrains[strainOutput] = fmu.getReal([outputStrainVrs[strainOutput]])[0] #strains due to acutator movement
```

```
prior to stiffness matrix construction.

        #
    ----------------------------------------------------------------------------------------------------

                                Rotation_compensation_vec = substractStrains.
    values()
                            print(Rotation_compensation_vec)
                            print(type(Rotation_compensation_vec))
                            print(list(Rotation_compensation_vec))
                            rotationVecs.append(np.array(list(
    Rotation_compensation_vec)))
                            print(type(np.array(list(Rotation_compensation_vec
    ))[0]))

                            for p in range(len(rotationStrain)):
                                rotationStrain[p][i][j][k][l] = np.array(list(
    Rotation_compensation_vec))[p]

        #i += 1
    #clean up and save right data
    fmu.terminate()
    outVecs = CUR_DIR + "\\rotationVectorsWFleet"
    np.save(outVecs, rotationStrain)

    for i in rotationVecs:
        with open(CUR_DIR + "\\rotationVecsFleet.csv", "a") as Test:
            wr = csv.writer(Test, dialect='excel')
            wr.writerow(i)

    print(rotationVecs)



    return rotationVecs


stiffnessandrotationMatrices()
```