Pilot Testing of a Hydraulic Bridge Exciter

RICHARD BORG AND OLIVIER DYMÉN
Abstract

In the design of a railway bridge that is meant for train speeds larger than 200 km/h, the Swedish Traffic Administration requires a dynamic analysis in addition to the conventional static design. The models used for static design may not always be suitable for dynamic analysis, which could lead to inaccurate estimations of the dynamic response.

The reason for this is a limited knowledge of the actual structural dynamic behaviour of bridges, which is why a hydraulic bridge exciter has been developed. With this device, smaller bridges can be excited in a load- or displacement-controlled manner under variable frequency and load. By having known inputs, the bridges’ dynamic properties can be evaluated using experimental modal analysis.

A finite element model of the double tracked railway bridge Pershagen was created in order to plan a pilot test of the bridge exciter. The influence of the load amplitude and sweep rate was evaluated. A theoretically optimal excitation position was also investigated in order to excite as many eigenmodes as possible from one position during the pilot test.

Based on these results, a pilot test was performed on the Pershagen Bridge. The dynamic properties of the bridge were evaluated based on the results from the test. From the pilot test it could be concluded that the load amplitude had a direct influence on the dynamic properties of the bridge, hence the dynamic behaviour is non-linear. The 1st vertical bending mode and its dynamic properties could also be identified.

Keywords— Hydraulic bridge exciter, controlled excitation, modal testing, experimental modal analysis, railway bridges, measurements, load shaker, FEM, FE-modelling, script
Referat

Pilottest av en hydraulisk broexciterare

Då en järnvägsbro som är avsedd för tåghastigheter över 200 km/h ska dimensioneras ställs det krav av Trafikverket att en dynamisk analys av bron skall utföras, utöver konventionell statisk dimensionering. De bromodeller som används för statisk analys är dock inte alltid lämpliga för dynamisk analys, vilket kan leda till felaktiga skattningar gällande den dynamiska inverkan.

Anledningen till detta är att kunskapen om broars dynamiska egenskaper är begränsade, och av denna anledning har en lastexciterare utvecklats. Med hjälp av denna anordning kan mindre broar exciteras med kontrollerad last eller förskjutning under variabel frekvens och last. Då dessa input-parametrar är kända kan broars dynamiska egenskaper utvärderas genom experimentell modal analys.

En finit element-modell av den tvåspåriga järnvägsbron vid Pershagen har skapats för att kunna planera ett pilottest av lastexciteraren. Svephastigheten och lastamplituden har analyserats. En teoretiskt optimal exciteringsposition har utvärderats för att excitera största möjliga antal moder från en och samma position under pilottestet.

Utifrån dessa resultat utfördes ett pilottest på bron vid Pershagen, där brons dynamiska egenskaper utvärderades utifrån resultaten. Från pilottestet kunde en slutsats dras om att lastamplituden hade en direkt inverkan på de dynamiska egenskaperna, vilket betyder att det dynamiska beteendet är olinjärt. Den första vertikala böjmoden och dess dynamiska egenskaper kunde också fastställas.
Acknowledgements

This master thesis was written for the department of Civil and Architectural Engineering at the Royal Institute of Technology (KTH), Division of Structural Engineering and Bridges. It was initiated as a merge project between KTH and Tyréns AB.

We would like to show our largest gratitude towards our two supervisors, Ph.D. Andreas Andersson and Ph.D. Mahir Ülker-Kaustell, who gave us this opportunity to write about and work with their newly crafted hydraulic bridge exciter. We would also like to thank them for their continuous help with our work. Considering their huge workload, we are grateful they always found time to answer our questions and motivate us throughout the project.

Large thanks to Tyréns AB and Martin Ålenius, the head of the Bridge Department, for giving us the possibility to work with our thesis at their office, and for providing us with coffee. We are also grateful to all employees of the Bridge Department at Tyréns AB for taking us in as their own and making our time here pleasantly enjoyable.

Furthermore, we want to thank Prof. Raid Karoumi, head of the Division for Structural Engineering and Bridges at KTH. Throughout our time as master students his courses and lectures were always an inspiration, which gave birth to our interest within the field of bridges and structural dynamics.

Finally, we want to thank our family, friends and loved ones for all their endless support while writing the final chapter of our five year long, but exciting, time at KTH.

Stockholm, June 2015

Richard Borg

Olivier Dymén
List of Abbreviations

| Abbreviation | Description                          |
|--------------|--------------------------------------|
| DOF          | Degree Of Freedom                    |
| EM           | Equation of Motion                    |
| EMA          | Experimental Modal Analysis          |
| FE           | Finite Element                        |
| FFT          | Fast Fourier Transform                |
| FRF          | Frequency-Response Function           |
| FVA          | Free Vibration Analysis               |
| MANABRIS     | Modal ANalysis of BRIdge Structures   |
| OMA          | Operational Modal Analysis            |
| REM          | Rotational Eccentric Mass             |
| SDF          | Single-Degree of Freedom              |
## List of Symbols

| Symbol | Description                                      | Unit               |
|--------|--------------------------------------------------|--------------------|
| ζ      | Damping ratio                                    | [%]               |
| θ      | Instantaneous phase                              | [rad]             |
| κ      | Sweep rate                                       | [rad/s]           |
| ν\(_c\) | Poisson’s ratio of concrete                      | [−]               |
| ρ\(_b\) | Density of ballast                               | [kg/m\(^3\)]     |
| ρ\(_c\) | Density of concrete                              | [kg/m\(^3\)]     |
| φ      | Phase                                            | [rad]             |
| ϕ      | Eigenvector                                      | [−]               |
| ω      | Angular frequency                                | [rad/s]           |
| ω\(_0\) | Initial angular frequency                        | [rad]             |
| ω\(_1\) | Final angular frequency                          | [rad]             |
| ω\(_n\) | Natural frequency                                | [rad/s]           |
| A      | Fourier transform of acceleration                 | [m/s\(^2\)]      |
| a      | Acceleration                                     | [m/s\(^2\)]      |
| a\(_0\) | Resonant amplitude times 1/√2                   | [−]               |
| a\(_1\) | Resonant amplitude                              | [−]               |
| c      | Damping coefficient                              | [N s/m]           |
| E\(_c\) | Dynamic Young’s modulus of concrete              | [Pa]              |
| F      | Force amplitude                                   | [N]               |
| f\(_a\) | Smaller forcing frequency corresponding to the resonant amplitude times 1/√2, a\(_0\) | [Hz]              |
| Symbol | Description                                                                 | Units     |
|--------|------------------------------------------------------------------------------|-----------|
| $f_b$  | Larger forcing frequency corresponding to the resonant amplitude times $1/\sqrt{2}$, $a_0$ | [Hz]      |
| $f_n$  | Natural frequency                                                             | [Hz]      |
| $f_s$  | Sampling frequency                                                            | [Hz]      |
| $G$    | Fourier transform of background noise                                         | [m/s²]    |
| $G_{FF}$ | Auto-spectrum                                                                 | [m²/s⁴]   |
| $G_{XF}$ | Cross-spectrum                                                                | [m³/Ns⁶] |
| $H$    | Complex frequency-response function                                           | [m/Ns²]   |
| $H_A$  | Frequency-response function for acceleration                                  | [m/Ns²]   |
| $k$    | Stiffness                                                                     | [N/m]     |
| $m$    | Mass                                                                          | [kg]      |
| $P$    | Fourier transform of force                                                    | [N]       |
| $p$    | Force                                                                         | [N]       |
| $r$    | Radius                                                                        | [m]       |
| $RI$   | Reference Indicator                                                           | [−]       |
| $T$    | Total sweep time                                                              | [s]       |
| $t$    | Time                                                                          | [s]       |
| $U$    | Fourier transform of displacement                                             | [m]       |
| $u$    | Displacement                                                                  | [m]       |
| $\dot{u}$ | Velocity                                                                    | [m/s]     |
| $\ddot{u}$ | Acceleration                                                                | [m/s²]    |
| $V$    | Fourier transform of velocity                                                 | [m/s]     |
# Table of Contents

Abstract .......................................................... i
Referat ................................................................ iii
Acknowledgement ....................................................... v
List of Abbreviations ................................................ vii
List of Symbols ........................................................ ix

## 1 Introduction ......................................................... 1
   1.1 Aim and Purpose .................................................. 2

## 2 Dynamic Testing of Railway Bridges ......................... 3
   2.1 Theoretical Background of Experimental Modal Analysis .... 4
      2.1.1 Half-Power Bandwidth Method ............................. 5
   2.2 Previously Performed Experimental Modal Analysis .......... 6
      2.2.1 Test Comparisons for the MANABRIS System ........... 7

## 3 Method .............................................................. 9
   3.1 Controlled Excitation ............................................ 9
      3.1.1 The Hydraulic Load Shaker ................................. 9
      3.1.2 The Pershagen Bridge ....................................... 10
      3.1.3 Routine Development ....................................... 11
   3.2 Finite Element Modelling ....................................... 12
      3.2.1 Geometry of the Pershagen Bridge ....................... 12
      3.2.2 Input Parameters ........................................... 12
      3.2.3 2D Model .................................................. 13
      3.2.4 3D Model .................................................. 14
      3.2.5 Convergence Analysis ..................................... 17
      3.2.6 Scripting in Python and MATLAB ....................... 19
   3.3 Evaluation in MATLAB ......................................... 20
      3.3.1 Linear Sweep Function ..................................... 20
      3.3.2 Implementation of Steady-State Theory and Sweep Function 22
Chapter 1

Introduction

In the design of railway bridges that are meant for train speeds larger than 200 km/h, dynamic analyses are required in addition to conventional static design (Banverket, 2008). These analyses are done in order to avoid bridges with e.g. resonance problems, uplift at bearings, as well as to limit the deflection and angle of rotation at the supports.

A set of serviceability criteria are stated in Eurocode SS-EN 1990 (SIS, 2006), related to passenger comfort and traffic safety. The main criterion considers the vertical accelerations of the bridge deck, which are not allowed to exceed a certain value. This is because large accelerations can cause ballast instability underneath the tracks, as well as unacceptable reduction in wheel-rail contact forces.

Model assumptions used in static design for bridges may not always be suitable in dynamic analysis. If assumptions are made that are too conservative, this may result in exceedance of the dynamic criteria for bridges that in reality are not at risk when it comes to resonance. Therefore a correct dynamic analysis is of large interest, since cost efficiency is an important factor in construction. More knowledge regarding the dynamic properties of bridges would also lead to more effective structures in the future and could save many existing bridges from being renovated or rebuilt when increasing the maximum allowed train speed on existing railways.

The dynamic response of bridges is difficult to estimate, mostly because of complicated boundary conditions, e.g. soil-structure interaction which may have a great influence on the dynamic properties, especially on short, stiff bridges. Numerical simulations are usually compared with experimental tests of passing trains. During these tests the free vibrations that occur after the train has passed are of large interest, but these are often small and do not give a lot of information regarding different modes. The problem is that the train only induces certain frequencies, from a combination of its axle distance, load magnitude and speed.

A solution to this would be to use controlled excitation, i.e. where both the load magnitude and frequency can be varied within useful ranges. For large structures, this is a difficult task to realize at a reasonable cost. However, the majority of the existing railway bridges in Sweden are small, with span lengths less than
30 m (Andersson et al., 2011), which enables the possibility for controlled excitation. Because of the large quantity of these bridges, it is also important to learn more about their dynamic capacity.

For these reasons a hydraulic bridge exciter (load shaker) has been developed by the Division of Structural Engineering and Bridges at KTH. A pilot test of this load shaker will be performed on the Pershagen Bridge.

1.1 Aim and Purpose

The purpose of this master thesis is to plan the pilot test of the hydraulic bridge exciter on the Pershagen Bridge. This will be done by computing theoretical dynamic responses of the bridge using a Finite Element (FE) model, which will serve as a foundation for the pilot test.

The aim of this master thesis is to

- Investigate an approach for evaluating the dynamic response from controlled excitation.
- Evaluate a theoretically optimal excitation position to be used during the pilot test.
- Evaluate the influence of the sweep rate and load amplitude from controlled excitation.
- Estimate the dynamic properties of the Pershagen Bridge from the pilot test.
Chapter 2

Dynamic Testing of Railway Bridges

There are mainly three different ways of testing the dynamic properties of a railway bridge:

- Free Vibration Analysis, (FVA)
- Operational Modal Analysis, (OMA)
- Experimental (Classical) Modal Analysis, (EMA)

The FVA method is common for railway bridges, where the free vibration after a train passage is measured and analysed while OMA methods use a continuous measurement of the ambient vibrations excited by e.g. traffic and wind. This continuous measurement is then typically analysed under the assumption that the forcing load is white noise. The EMA method is a method where the exciting load is pre-defined and applied by the experimenter, by using e.g. a rotating mass shaker or hydraulic load shaker. The amount of response from a structure when using controlled excitation depends on the size of the structure, which is why OMA is favourable for larger structures where controlled excitation is difficult to achieve. However, there are some drawbacks with OMA, e.g. that you cannot compute the Frequency-Response Function (FRF) from it (Grosel et al., 2014). In short terms:

- FVA and OMA: Unknown input signal - known output signal
- EMA: Known input signal - known output signal

From EMA information about the natural frequencies, mode shapes, modal damping coefficients and the FRF can be extracted. One advantage of EMA contra OMA is that the mean of excitation used does not affect the mass and stiffness of the structure of interest. This leads to results showing the properties of the structure alone rather than the properties of the structure-vehicle interaction (Zwolski and Bień, 2011), given that OMA is used on measurements from train passages.
CHAPTER 2. DYNAMIC TESTING OF RAILWAY BRIDGES

2.1 Theoretical Background of Experimental Modal Analysis

EMA works under the assumption that the structure is time invariant and linear, following Maxwell’s theory of reciprocity (Grosel et al., 2014). This means that a measured response at location X caused by an excitation at location Y gives the same response as the opposite scenario (Brandt, 2011). If these assumptions are made then an arbitrary excitation \( p(t) \) can, according to Chopra (2012), be represented by the Fourier integral:

\[
p(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} P(\omega) e^{i\omega t} \, d\omega
\]

Equation (2.2) represents the Fourier transform of the time function \( p(t) \), and Equation (2.1) is in turn the inverse Fourier transform of the frequency function \( P(\omega) \).

The response from the bridge will be measured in terms of acceleration, and the Fourier transform \( A(\omega) \) of the acceleration solution \( \ddot{u}(t) \) can be expressed as

\[
A(\omega) = H_A(\omega) P(\omega)
\]

where the complex Frequency-Response Function (FRF) \( H_A(\omega) \) describes the acceleration response of the system to harmonic excitation, and is yet to be determined.

For a linear Single Degree-of-Freedom (SDF) system subjected to an external force the Equation of Motion (EM) is defined as

\[
m\ddot{u} + c\dot{u} + ku = p(t)
\]

where \( m \) is the mass of the system, \( c \) is the damping coefficient of the system, and \( k \) is the stiffness of the system. The steady-state displacement of this system will be a harmonic motion at the forcing frequency \( \omega \):

\[
u(t) = U(\omega)e^{i\omega t}
\]

which can be differentiated and expressed in terms of velocity and acceleration:

\[
\dot{u}(t) = i\omega U(\omega)e^{i\omega t} = V(\omega)e^{i\omega t}
\]

\[
\ddot{u}(t) = -\omega^2 U(\omega)e^{i\omega t} = A(\omega)e^{i\omega t}
\]
2.1. THEORETICAL BACKGROUND OF EXPERIMENTAL MODAL ANALYSIS

Consider the external force in Equation (2.4) to have unit amplitude:

\[ p(t) = 1e^{i\omega t} \]  

(2.7)

Equations (2.5) and (2.6) are then substituted in Equation (2.4) and expressed for the acceleration response:

\[ A(\omega)e^{i\omega t}(-\frac{1}{\omega^2})(-\omega^2m + i\omega c + k) = e^{i\omega t} \]  

(2.8)

Cancelling \( e^{i\omega t} \) from both sides of this equation gives

\[ A(\omega) = \frac{-\omega^2}{-\omega^2m + i\omega c + k} \]  

(2.9)

and by substituting this in Equation (2.3), still under the assumption that the external force is unity, the FRF \( H_A(\omega) \) is determined:

\[ H_A(\omega) = \frac{-\omega^2}{-\omega^2m + i\omega c + k} \]  

(2.10)

By using the definition for the natural frequency of vibration, \( \omega_n = \sqrt{k/m} \), and the damping ratio, \( \zeta = c/2m\omega_n \), Equation (2.10) can be written as

\[ H_A(\omega) = \frac{1}{k \left[ 1 - \frac{\omega}{\omega_n} \right]^2 + i\frac{2\zeta}{\omega_n}} \]  

(2.11)

The FRF \( H_A(\omega) \) describes the steady-state acceleration response of the system to a harmonic unit force. By using the Fourier transform of the input signal from the excitation \( p(t) \), Equation (2.3) gives the relationship between the response from the controlled excitation and the FRF.

### 2.1.1 Half-Power Bandwidth Method

An effective way of evaluating the damping ratio \( \zeta \) from a structure is by using the half-power bandwidth method, which is defined using the FRF (Chopra, 2012). If \( f_a \) and \( f_b \) are the forcing frequencies on either side of the resonance frequency \( f_n \) at which the amplitude \( a_0 \) is \( 1/\sqrt{2} \) times the resonant amplitude \( a_1 \), then for small values of \( \zeta \)

\[ \frac{f_b - f_a}{f_n} = 2\zeta \]  

(2.12)

which can be rewritten as
CHAPTER 2. DYNAMIC TESTING OF RAILWAY BRIDGES

\[
\zeta = \frac{f_b - f_a}{2f_n}
\]  

(2.13)

This result means that it is possible to evaluate the damping ratio from forced vibration tests without knowing the applied force amplitude. The definition of the half-power bandwidth method is shown in Figure 2.1 for a natural frequency of 1 Hz and a damping ratio of 10 %.

![Figure 2.1. Definition of the half-power bandwidth.](image)

2.2 Previously Performed Experimental Modal Analysis

In 2011 Zwolski and Bień (Zwolski and Bień, 2011) studied and built an exciter. Their exciter device, called MANABRIS (Modal ANAlysis of BRIdge Structures), was designed for both road bridges and railway bridges and uses a Rotational Eccentric Mass (REM) exciter. The idea behind the MANABRIS system was to create a testing device which is highly portable with a comprehensive control by a computer.

The advantage of using the REM is that it is both portable and convenient. However, due to the rotational properties, the frequency range is 3 - 14 Hz and 3 - 30 Hz for the road bridge and railway bridge respectively. The wider range for the railway bridge is because the device can be clamped to the rails and prevent it
2.2. PREVIOUSLY PERFORMED EXPERIMENTAL MODAL ANALYSIS

from losing contact with the bridge at high frequencies. Another inconvenience is the difficulty to have total control of the output excitation force from the REM. This is due to proportionality of the force to the eccentricity and the angular frequency of the REM, according to the equation of the centripetal force \( F = mr\omega^2 \). This relationship gives rise to problems while exciting large structures at low frequencies since the force will not be sufficient (Zwolski and Bień, 2011). Zwolski and Bień computed FRFs for the tested structure according to Equation (2.14):

\[
H_1(\omega) = \frac{G_{XF}(\omega)}{G_{FF}(\omega)}
\]  

(2.14)

where \( H_1(\omega) \) is the FRF displacement estimator, \( G_{XF}(\omega) \) is the cross-spectrum of the measured displacement, velocity and acceleration and \( G_{FF}(\omega) \) is the autospectrum of the force at the frequency \( \omega \). These are calculated using FFT and averaged over a number of repetitions of tests. It can be seen that Equation (2.14) can be related to Equation (2.3) derived above. The modal parameters of the bridge are thereafter estimated with curve fitting algorithms.

Zwolski and Bień also describes a "Reference Indicator" (\( RI \)). This indicator provides a measure of the number of eigenmodes present in a specific region of the structure. A high value of \( RI \) means that the location is well suited for a reference point or as an excitation point. \( RI \) is derived from:

\[
RI_k = \prod_{i=1}^{r} |\phi_{k,i}|
\]  

(2.15)

where \( r \) is the number of modes taken into account, \( k \) the degree of freedom (DOF) number (direction) and \( \phi_k \) is the eigenvector calculated with a FE-model of the bridge.

2.2.1 Test Comparisons for the MANABRIS System

When the MANABRIS system was to be tested, Zwolski and Bień performed OMA, FVA and EMA tests on a railway bridge for comparison. The OMA and FVA tests were performed using freight trains and the EMA tests were performed with three different excitations. What could be seen from OMA and FVA tests was that when the autospectrum was calculated for the forced vibration and the free vibration separately, the resonance peaks where not coinciding, which could be explained by the added mass to the bridge (Figure 2.2).

The three different excitations used for the EMA tests where:

- Exponential sweep, 3 - 24 Hz, total of 11 min.
- Linear sweep, 3 - 24 Hz, total of 11 min.
- Harmonic excitations in the range of 3 - 13.2 Hz with a step increment of either 0.016 or 0.032 Hz, total of 2 hours.
What could be distinguished from the three EMA tests (Figure 2.2) is that harmonic excitation yielded less noise than the two sweep tests, especially at low frequencies which is believed to be a result from the low amount of excitation energy (Zwolski and Bień, 2011).

According to Zwolski and Bień (2011), the most effective excitation was the two sweep excitations while the harmonic gave good results with less noise but at the price of time. Clearly, the EMA tests gave a more precise FRF than the OMA tests. Furthermore, the OMA tests could not identify higher order modes of vibration.
Chapter 3

Method

In this chapter the method used for planning the pilot test on the Pershagen Bridge is presented. Section 3.1 gives a description of the hydraulic load shaker and the prerequisites for the pilot test. Section 3.2 describes the geometric definitions and assumptions made for the FE model of the bridge, and the method used for simulating the load shaker. Finally, Section 3.3 presents the evaluation process of a number of factors that affect the measurements from the pilot test:

- Excitation position
- Sweep rate
- Load amplitude

A method for computing the dynamic properties from both the FE model and the pilot test is also presented in Section 3.3.

3.1 Controlled Excitation

For smaller bridges controlled excitation is a favourable method in order to determine the dynamic properties of the structure, such as mode shapes, eigenfrequencies and damping ratio. In Sweden, most of the existing railway bridges have a span length less than 30 m (Andersson et al., 2011) and are situated in locations where background noise from different activities such as traffic or human impact is limited, and the wind buffeting loads are small or even negligible. Overall, this situation is unsuitable for output-only methods, which is why controlled excitation would be a better option.

3.1.1 The Hydraulic Load Shaker

The load shaker developed by the Division of Structural Engineering and Bridges at KTH is equipped with a hydraulic cylinder, which gives the ability to perform load or displacement controlled tests on structures at variable frequency. The hydraulic
cylinder has a control system which interacts with a load cell that has been installed at the contact point between the load shaker and the bridge deck. In turn, the load cell sends back real time data of the applied load on the bridge deck to the control system, in order to adjust the force in the hydraulic cylinder. This is done in order to take the effects from the ground on which the load shaker is placed into consideration, as well as the local dynamic behaviour of the load shaker. This in turn enables control of the input signal applied to the structure.

The load shaker can be applied to any structure for which the available load amplitude and frequency ranges are sufficient. It was designed for smaller railway bridges, with a maximum load amplitude of 50 kN and a frequency range of 0 - 70 Hz. However, for the pilot test the frequency range of interest has been limited to 0 - 50 Hz. A schematic drawing of the load shaker is shown in Figure 3.1.

![Schematic figure of the load shaker.](image)

### 3.1.2 The Pershagen Bridge

The bridge where the pilot test of the load shaker was performed is called the Pershagen Bridge, and it is located outside of Södertälje, Sweden. It is a 46.6 m long double track railway bridge with three spans and integrated abutments.

The abutments are surrounded by soil fill, which makes the dynamic response very hard to estimate. It is a difficult task to model soil-structure interaction behaviour using finite element software, and for this reason it is a field of large interest in dynamics of structures. The information needed in order to give a good estimate of the soil-structure interaction is usually vast and uncertain, and the included soil in the model needs to be of an extensive size when modelling the support conditions, leading to very long computational times (Grosel et al., 2014). By using controlled excitation the hopes are that the dynamic response will be well-defined, and that this in turn can give new information regarding the soil-structure interaction.

The load shaker is placed on the ground underneath the bridge and pre-stressed against the bridge deck, in order to make sure that the load shaker always stays connected to the bridge deck. However at the Pershagen Bridge, because of the soil
3.1. CONTROLLED EXCITATION

slopes below the side spans (Figure 3.2) it was only possible to place the load shaker somewhere underneath the middle span.

3.1.3 Routine Development

In order to obtain useful and well-defined results from the pilot tests of the load shaker on the Pershagen Bridge, the position of the load shaker and the accelerometers needed to be evaluated. The position of the load shaker was decided based on the theoretical FE model, and from which position on the bridge deck it was likely to excite as many eigenmodes as possible. The instrumentation of the accelerometers was also decided based on the model, and from where it was likely to detect as many eigenfrequencies and estimate as many mode shapes as possible.

As mentioned in Section 3.1.2, it was only possible to place the load shaker underneath the middle span of the bridge. Another aspect that needed to be taken into consideration was that traffic was running on the bridge during the tests, since the tests was done during daytime. This affected the sweep rate for the load shaker, since a train passage would disturb the response from the accelerometers and perhaps harm the load shaker.

A total of twelve accelerometers were to be used during the tests. For the accelerometers the main focus was to find out which layout gave as much information as possible regarding the eigenfrequencies and mode shapes, by using one accelerometer set-up. It would be optimal to use a reference accelerometer and perform several tests with different set-ups, but because the test was done over one day there were limitations to how many different sweeps it would be possible to carry out during the pilot test. For this reason it was decided to use one set-up.

Figure 3.2. The Pershagen Bridge.


3.2 Finite Element Modelling

Modelling a structure using FE software is an effective way of estimating a structure’s behaviour. For this project the software BRIGADE Plus was used, which is a software based on the Abaqus graphical user interface and solver technology, but with tools to facilitate bridge specific modelling. This software will be referred to as BRIGADE. The theory behind FE modelling is explained in Concepts and Applications of Finite Element Analysis (Cook et al., 2002).

3.2.1 Geometry of the Pershagen Bridge

The Pershagen Bridge is a slab bridge with three spans and end-shields (Figure 3.3). The total length of the bridge deck is 46.6 m with a width of 11.9 m. The middle span is 18.4 m long and the two side spans are each 11.1 m long. The ends of the bridge deck acts as cantilevers with the end-shields and are 3 m long. The bridge is supported by four support pairs. The two outer support pairs are roller bearings while the two middle support pairs are fixed bearings. The supports are columns of reinforced concrete with a diameter of 1.1 m and 1.35 m for the outer supports and the middle supports respectively. The bridge deck is constructed of reinforced concrete with an average thickness of 1.1 m and covered by a 0.6 m thick layer of ballast.

![Figure 3.3. Blueprint of the Pershagen Bridge (Banverket, 1991).](image)

3.2.2 Input Parameters

The input data was either gathered from the technical drawings or from Eurocodes. Table 3.1 shows all the input parameters used in the FE-models. A time increment of 0.001 s was used, which corresponds to a sampling frequency of $f_s = 1000 \text{ Hz}$. Furthermore, with the assumption that the concrete is cracked, the dynamic Young’s modulus should be reduced to approximately 60 % of its static value (Andersson et al., 2010). The concrete quality of the bridge is K40.
3.2. FINITE ELEMENT MODELLING

Table 3.1. Material parameters for the FE-models.

| Description                      | Denotation | Value     | Source                                      |
|----------------------------------|------------|-----------|---------------------------------------------|
| Density, reinforced concrete     | $\rho_c$  | 2500 kg/m$^3$ | Table A.1, EN 1991-1-1 (SIS, 2002)          |
| Density, ballast                 | $\rho_b$  | 2000 kg/m$^3$ | Table A.6, EN 1991-1-1 (SIS, 2002)          |
| Young’s modulus, concrete K40, dynamic | $E_c$   | 21 GPa    | Table 3.1, EN 1992-1-1 (SIS, 2005) and (Andersson et al., 2010) |
| Poisson’s ratio, concrete        | $\nu_c$   | 0.2       | 3.1.3(4), EN 1992-1-1 (SIS, 2005)           |
| Damping ratio                    | $\zeta$   | 1.5 %     | Table C.1, (Bachmann et al., 1995)          |

3.2.3 2D Model

In order to explore and learn BRIGADE the bridge was first modelled in 2D. The bridge was modelled as a rectangular beam with a total length of 46.6 m, width of 11.9 m and a height of 1.1 m. The middle supports are modelled with pinned conditions and the outer supports with roller conditions. The end-shields were taken into consideration by placing a point mass of 35 000 kg each at the ends of the beam. As shown in Figure 3.4 the columns of the bridge were not taken into consideration for the 2D model.

![Figure 3.4. 2D model of the Pershagen Bridge.](image)

Since the bridge is more or less aligned straight as well as symmetric in the lateral direction, a 2D model might be sufficient for a representation of the dynamic behaviour in the vertical and longitudinal direction. However, in the present context, the aim is to estimate all modes of vibration in the frequency range of the exciter.
3.2.4 3D Model

To be able to estimate the full response of the bridge, a 3D model was assembled in BRIGADE (Figure 3.5). In order to have a simple model with less computational time, a couple of assumptions were made:

- A linear behaviour for the entire structure.
- No soil-bridge interaction at the end-shields nor at the column foundations.
- The bridge deck and the ballast are one homogeneous layer.

![Figure 3.5. 3D model of the Pershagen Bridge.](image)

When modelling in BRIGADE the geometries of the structure are first defined as parts, and then assembled together to form a model (Figure 3.5). The different parts modelled for the 3D model are listed below:

- Bridge deck.
- Wing-walls.
- End-shields.
- 1\textsuperscript{st} support (left in Figure 3.5).
- 2\textsuperscript{nd} and 3\textsuperscript{rd} supports.
- 4\textsuperscript{th} support (right in Figure 3.5).

The bridge deck, end-shields and wing-walls were all modelled with four-noded shell elements (S4R) with respective thickness, while the columns were modelled with two-noded shear flexible beam elements (B31). See Appendix A for the geometric definitions of the parts.

When assembling the model, specific interaction conditions needs to be assigned to the parts to ensure that BRIGADE understands how they interact with each other. The end-shield, wing-wall and bridge deck are assumed to be a solid unit. To model this, tie interactions were used. The wing-walls are tied to the end-shields and the end-shields are tied to the deck. Since no soil-bridge interaction was taken into consideration the end-shields were considered to be unaffected by
3.2. FINITE ELEMENT MODELLING

the surroundings. For the same reason, the boundary conditions for the columns were assumed to be fixed at the bottom with corresponding support conditions at the top, i.e. pinned support conditions between the 2nd and 3rd columns and deck and roller conditions between the 1st and 4th columns and deck. For this, coupling conditions were used as interaction between the supports and the bridge deck.

In order to simulate the load shaker, a unit harmonic point load was applied at its suggested position. There were three different calculation methods to approach the simulation of the excitation force in BRIGADE: steady-state, modal dynamics, and dynamic implicit. In both modal dynamics and dynamic implicit the concentrated force was modelled with an amplitude sine-sweep function as the input signal according to Equation (3.11) in Section 3.3.1. When using the steady-state method the load input signal is implemented later on, which is explained in Section 3.3.2.

The difference between the results from these three different calculation methods are that with modal dynamics and dynamic implicit, BRIGADE calculates how the bridge will respond to the excitation in time domain, i.e. the acceleration for each time increment is given. With the steady-state method, BRIGADE calculates the response of the bridge in the frequency domain, i.e. the acceleration at each frequency increment (Figure 3.6). As explained in Section 2.1 with Equation (2.3) and an applied unit load in the model, the steady-state result from BRIGADE is the FRF, \( H_A(\omega) \), and with an inverse Fast Fourier Transform (FFT) the acceleration response in time domain can be obtained. This is further explained in Section 3.3.2.

![Figure 3.6](image.png)

**Figure 3.6.** The steady-state method result from BRIGADE for a sweep rate of 0.25 Hz/s over a frequency range of 0 - 15 Hz and an applied unit force excitation.

To be able to decide which method was the most suitable in terms of accuracy of results as well as computational time, a comparison was made. The comparison was done at the same position of the bridge deck with a sweep rate of 0.25 Hz/s over a frequency range of 0 - 15 Hz. The applied force amplitude was unity for all three different methods. The result from the comparison is shown in Figure 3.7 for the acceleration response in time domain and in Table 3.2 where the number of calculated increments and computational time are presented for the three different calculation methods.
Figure 3.7. Comparison of the acceleration response in time domain for three different evaluation methods in BRIGADE.

Table 3.2. Comparison of number of increments and computational time for three different evaluation methods in BRIGADE.

| Method          | Number of increments | Computational time |
|-----------------|----------------------|--------------------|
| Steady-state    | 900                  | 5 min              |
| Modal dynamics  | 60 000               | 70 min             |
| Dynamic implicit| 60 000               | 300 min            |

Since steady-state both has significantly less increments and computational time than both modal dynamics and dynamic implicit, and with more or less the same response (Figure 3.7), it was decided to be the best method for this project. However, as can be seen there is a slight disturbance at the beginning of the signal. This is because of the use of FFT when calculating the response in steady-state, but this will have smaller influence on the response when using a slower sweep rate.
and a larger frequency range. Also, as long as the disturbance does not affect any frequencies of interest it can be neglected.

### 3.2.5 Convergence Analysis

When evaluating structures using FE software, the model is divided into smaller elements, i.e. a mesh. To determine how fine the mesh should be in order to produce accurate results, a convergence analysis was performed. A comparison of different mesh sizes was performed with regard to how the element size affects the eigenfrequencies as well the dynamic response of the 3D model when modelling controlled excitation on the Pershagen Bridge.

The comparison regarding the eigenfrequency convergence of the model was performed over a frequency range of 0 - 50 Hz. Five different element sizes were evaluated for the bridge deck: 2 m, 1 m, 0.5 m, 0.25 m, and 0.1 m. The element size for the end-shields was defined as 0.7 times the deck element size, and for the wing-walls as 0.5 times the deck element size. The element size for the columns was the same as for the deck. The differing element sizes for the end-shields and the wing-walls were to ensure good tie-interactions conditions between wing-walls and end-shields to the bridge deck.

A number of 52 modes was found in the frequency range for all element sizes except for 2 m, which found 51 modes. As shown in Figure 3.8 the eigenfrequencies differ more for the higher mode numbers, but for an element size of 0.5 m a satisfactory convergence is reached.

In order to investigate the convergence regarding the dynamic response of the model, a comparison was done for the FRF $H_A(\omega)$ with a sweep rate of 0.167 Hz/s in a frequency range of 40 - 50 Hz. The response was evaluated at the same point of the bridge deck for all element sizes.

As presented in Figure 3.9 the general shape of the FRF reach satisfactory convergence for an element size of 0.5 m, though there is an offset of about 0.2 Hz for the frequencies. However, the purpose of the model is to give an estimation of the dynamic behaviour in order to have comparable data for the field test, it is not meant to be an exact representation of the actual bridge. For this reason, an element size of 0.5 m was considered good enough.
CHAPTER 3. METHOD

Figure 3.8. Convergence analysis for eigenfrequencies for five different element sizes over a frequency range of 0 - 50 Hz. The top figure shows the first 52 modes and the bottom figure is zoomed in over mode 40 to 52.

Figure 3.9. Convergence analysis for the FRF, $H_A(\omega)$, for five different element sizes over a frequency range of 40 - 50 Hz.
3.2. FINITE ELEMENT MODELLING

3.2.6 Scripting in Python and MATLAB

To be able to process and evaluate the output data from BRIGADE effectively, it needed to be exported to MATLAB. However, manual result extraction would be too time consuming since the amount of output data that was received from the dynamic analysis in BRIGADE was huge (around 3 GB).

This was solved by using Python based scripting. Since BRIGADE is based on Python code, it is possible to extract the output data from the result files. Another advantage of using scripts is that models can be assembled in BRIGADE by running user-defined scripts. This is a very effective way of being able to have control over and update the input parameters such as load position, Young’s modulus, frequency range, etc. It is also very effective for quickly assembling several models with different input parameters for a comparison of results.

However, the scripts needed to be written from scratch, since older scripts do not necessarily apply to the bridge used in this project. Two different Python scripts were written: one that assembled the model in BRIGADE, and one that extracted all the relevant output data from BRIGADE’s result file into structured tables in text files. All the relevant model input parameters were organized in a separate Python script, and later imported into the model assembly script as well as the result extraction script. This proved very helpful since parts in the model are identified by their names, which are defined in the variables script. This in turn gave full control to easily extract the correct result from the chosen part of interest in the model. The source code for the Python scripts can be found in Appendix C.

After the output data had been extracted to text files, they were imported and read with a MATLAB script. This script organized the results into MATLAB structure arrays and then saved them as MATLAB data files. This gave easy access and full control over the output data in MATLAB. The source code for this script can also be found in Appendix C.

The main reason why Python scripts were needed was that the result files written by BRIGADE are binary which MATLAB cannot read. To be able to read these binary files and translate the results written in those to ASCII characters (text format) the Python modules defined in the BRIGADE software were needed.
3.3 Evaluation in MATLAB

As described in Section 2.1 there is a relationship between the FRF $H_A(\omega)$, the applied excitation force $P(\omega)$, and the response from the structure $A(\omega)$, according to Equation (2.3). The FRF is calculated from the BRIGADE model. Since the expected response from excitation of the Pershagen Bridge is to be evaluated, the excitation force needs to be defined.

### 3.3.1 Linear Sweep Function

From the steady-state theory it is known that

$$A(\omega) = H_A(\omega)P(\omega)$$

where the FRF $H_A(\omega)$ describes the acceleration response of the system, and can be evaluated for all points of interest on the bridge by using steady-state in BRIGADE. This output data is then extracted into MATLAB structure arrays (Section 3.2.6).

$P(\omega)$ is the Fourier transform of an arbitrary excitation, which for this project is the sweep function of the load shaker. According to Viswanathan (2014), in order to generate a sweep function, consider a regular cosine signal with force amplitude $F$, angular frequency $\omega_0$, and initial phase $\varphi$:

$$p(t) = F\cos(\omega_0 t + \varphi)$$

This can be written as a function of instantaneous phase:

$$p(t) = F\cos(\theta(t))$$

where

$$\theta(t) = \omega_0 t + \varphi$$

For this project a linear sweep is used, which means that the frequency change is linear in time. The first time derivative of the instantaneous phase $\theta$ is equal to the angular frequency $\omega$:

$$\frac{d}{dt}\theta(t) = \omega(t)$$

which for Equation (3.4) would be constant. However, for a linear frequency change, the instantaneous phase must be in quadratic form:

$$\theta(t) = \alpha t^2 + \omega_0 t + \varphi$$
3.3. EVALUATION IN MATLAB

where \( \alpha \) is some constant. The first time derivative becomes

\[
\frac{d}{dt} \theta(t) = 2\alpha t + \omega_0
\]  

(3.7)

If the total sweep time is \( T \) and the final frequency is \( \omega_1 \), the sweep rate \( \kappa \) is given by

\[
\kappa = 2\alpha = \frac{\omega_1 - \omega_0}{T}
\]  

(3.8)

This is then substituted in Equation (3.7):

\[
\frac{d}{dt} \theta(t) = \omega(t) = \kappa t + \omega_0
\]  

(3.9)

Integrating over time:

\[
\theta(t) = \int \omega(t) dt = \int (\kappa t + \omega_0) dt = \frac{\kappa}{2} t^2 + \omega_0 t + \varphi = \left( \frac{\kappa}{2} t + \omega_0 \right) t + \varphi
\]  

(3.10)

where \( \varphi \) is the initial phase. By substituting in Equation (3.3) this gives the modified sweep signal:

\[
p(t) = F \cos \left[ \left( \frac{\kappa}{2} t + \omega_0 \right) t + \varphi \right]
\]  

(3.11)

In Figure 3.10 the result of the sweep signal \( p(t) \) can be seen with unit force amplitude, a frequency range of 0 - 10 Hz, a sweep rate of 1 Hz/s, and an initial phase of \(-\pi/2\).

![Figure 3.10. Linear sweep function, \( p(t) \) with a frequency range of 0 - 10 Hz over a time of 10 s.](image-url)
3.3.2 Implementation of Steady-State Theory and Sweep Function

Using the result from Section 3.2.4 and 3.3.1, the process of computing the time history of a load sweep can be summarized as follows:

1. Compute the sweep function \( p(t) \) using Equation (3.11).

2. Compute the FFT of the sweep function, \( P(\omega) \).

3. Compute the FRF \( H_A(\omega) \) from the BRIGADE model.

4. Multiply \( P(\omega) \) with \( H_A(\omega) \) in order to calculate the acceleration in frequency domain, \( A(\omega) \).

5. Compute the acceleration in time domain \( a(t) \) by using the inverse FFT of \( A(\omega) \).

This is illustrated in Figure 3.11, which shows the whole process for calculating the acceleration response in the time domain, \( a(t) \), from the BRIGADE model of the Pershagen Bridge. The first plot shows the excitation force \( p(t) \), defined for a frequency range of 0 - 50 Hz and a load amplitude of 1 N with a total sweep time of 300 s, resulting in a sweep rate of 0.167 Hz/s. According to Equation (3.1) the excitation force needs to be in the frequency domain, which is achieved by applying the FFT on \( p(t) \), resulting in the second plot showing \( P(\omega) \). Applying the FFT on the linear sweep function gives a spectrum with constant amplitude over the whole frequency range, with exception to the beginning and the end of the frequency range where there are some numerical disturbances. The third plot shows the FRF response from the BRIGADE model and the fourth plot the total response from the structure when excited. The expected response from the pilot test is consequently the inverse FFT of \( A(\omega) \), giving \( a(t) \) which is the acceleration in the time domain.

When measurements from the actual excitation with the load shaker are performed, the accelerations \( a(t) \) at the bridge deck as well as the excitation force \( p(t) \) from the load shaker will be recorded. With this data the FRF can be calculated by applying the FFT on the acceleration and the excitation force and rearranging Equation (3.1) to:

\[
H_A(\omega) = \frac{A(\omega)}{P(\omega)} \tag{3.12}
\]
Figure 3.11. Five plots showing the linear sweep function in both time and frequency domain, \( p(t) \) and \( P(\omega) \), the FRF from BRIGADE model, \( H_A(\omega) \), and the expected acceleration in frequency and time domain, \( A(\omega) \) and \( a(t) \).
3.3.3 Theoretical Excitation Position

Due to the load shaker’s size and weight it is inconvenient to move around during a field test. Therefore, it would be beneficial to find the theoretically optimal excitation position, which is the one position on the bridge deck from where it is ideal to excite as many modes as possible with a well-defined response. For this reason, in order to find the theoretically optimal excitation position, the mode shapes of the model needed to be evaluated.

As mentioned in Section 3.1.2, the soil slopes below the side spans limit the position of the load shaker to somewhere underneath the middle span. To find a good excitation position, the $RI$ factor presented in Section 2.2, Equation (2.15) was used:

$$RI_k = \prod_{i=1}^{r} |\phi_{k,i}|$$

where $r$ is the number of modes taken into account, $k$ the DOF number (direction) and $\phi_i$ is the $i$:th eigenvector for DOF $k$ calculated with a FE-model of the bridge. The result from this evaluation is presented in Section 4.2.1.

3.3.4 Sweep Rate

An important factor to investigate is the sweep rate of the hydraulic load shaker during testing. A couple of things needs to be considered when evaluating this:

- The response must be well-defined enough for evaluation of the dynamic properties.
- There are trains passing the Pershagen Bridge about every 5 min.
- The economical aspect.

A well-defined response means that the sweep rate can not compromise the results, which would lead to inaccurate outputs. Since there is a train passage at about every 5 min, it would be optimal if the total sweep time could be shorter than this. Otherwise, the sweep would have to be divided into several segments in order to avoid disturbance in the response.

In the beginning of this thesis the total sweep time was investigated instead of the sweep rate. It was discovered later on that the sweep rate was better to use for comparison, since the total sweep time would vary when testing different frequency ranges. Therefore, all earlier investigations were done using the total sweep time, but has been converted to sweep rate to keep consistency over the entire thesis.

As shown in Section 3.3.2, by applying FFT on the sweep signal $p(t)$ in MATLAB, the frequency spectrum $P(\omega)$ can be determined. As mentioned in Section 3.1.1, the frequency range of interest for this project is 0 - 50 Hz. The frequency spectrum $P(\omega)$ was compared for sweep rates of 0.028 Hz/s, 0.056 Hz/s, 0.083 Hz/s, 0.167 Hz/s, 0.417 Hz/s, and 0.833 Hz/s with a force amplitude of
3.3. EVALUATION IN MATLAB

10 kN. The comparisons are shown in Figure 3.12, and as can be seen there is some numerical disturbance at the beginning and end of the spectrum. When the sweep rate is slower the disturbance has a smaller frequency interval, and when the sweep rate is 0.167 Hz/s the disturbance only affects frequencies greater than 42 Hz, which would probably be good enough for this project. Also, since there is a train passage at about every 5 min and \((50/0.167)/60 = 5\) min, this means that a full sweep can be performed within that time.

![Figure 3.12. A comparison of sweep rates in frequency domain, \(P(\omega)\).](image)

In Figure 3.13 a comparison of the acceleration response for all the sweep rates that are less than 0.167 Hz/s is shown. It is apparent that a slower sweep rate gives higher accelerations and a higher energy content, but does not affect the eigenfrequency. This is because of the assumption that the system is linear. This will be discussed later in connection with the actual test results.

### 3.3.5 Load Amplitude and Noise

The load amplitude of the excitation must be large enough for three reasons:

- To overcome disturbing effects from background noise.
- To enable the detection of non-linear structures.
- To reach a sufficient magnitude of detected acceleration.

For this project, a goal has been set to reach accelerations of 1 m/s² in the bridge deck during the excitation. Usually when measuring accelerations from free
vibrations, the maximum acceleration are somewhere in the order of 0.1 m/s². Therefore, it would be a great improvement to reach 1 m/s².

As mentioned in Section 3.1.1, the maximum load amplitude of the load shaker is 50 kN. However, there is no way of fastening the top of the load shaker to the bridge deck, which means that the load shaker has to subject a compressive force throughout the sweep tests. In order to examine how this will affect the output response a comparison was made for two cases: one with a sweep load amplitude of 25 kN where the sweep oscillates around 0 kN, and one with a sweep load amplitude of 25 kN where the sweep oscillates around 25 kN (Figure 3.14). As can be seen, the response is exactly the same for both cases. However, this will limit the maximum load amplitude of the sweep to 25 kN.

In order to examine the acceleration response of the bridge deck a comparison was made around the 1st bending frequency with a sweep rate of 0.167 Hz/s and five different load amplitudes: 0.1 kN, 1 kN, 5 kN, 10 kN and 25 kN (Figure 3.15). The background noise was defined to be a normally distributed random white noise with a mean value of 0.001 m/s², and was added to the signal in frequency domain according to

\[ A(\omega) = H A(\omega) P(\omega) + G(\omega) \]  

(3.14)
3.3. EVALUATION IN MATLAB

where \( G(\omega) \) is the FFT of the background noise. The middle and bottom plot in Figure 3.15 show the response in frequency domain, \( A(\omega) \), for a load amplitude of 1 kN and 0.1 kN respectively. For 0.1 kN the background noise clearly affects the response, but for 1 kN it is barely detectable. Since the maximum load amplitude is 25 kN, according to theory, the background noise will not be a problem.

Theoretically, the acceleration is almost 1 m/s\(^2\) for a load amplitude of 25 kN, which is the maximum load amplitude. However, with a slower sweep rate the accelerations may exceed the pre-set goal of 1 m/s\(^2\).
Figure 3.15. Top figure: A comparison of the response in time domain, $a(t)$, with different load amplitudes and implemented background noise. Middle and bottom figure: The response in frequency domain, $A(\omega)$, for two different load amplitudes.
Chapter 4

Result

4.1 3D Model

4.1.1 Eigenfrequencies and Mode Shapes

The eigenfrequencies of the model were determined from the built-in eigenfrequency evaluation in BRIGADE, where the frequency range of interest was defined as 0 - 50 Hz. All eigenfrequencies are presented in Appendix B.

The corresponding mode shapes for the eigenfrequencies were investigated in BRIGADE. Mode shapes that lacked vertical movements in the bridge deck were considered irrelevant for this project (e.g. lateral and longitudinal mode shapes) and were not included for evaluation of the theoretically optimal excitation position. The modes of interest along with mode shape descriptions are presented in Table 4.1, while the 20 first mode shapes of interest can be found in Appendix B.

4.1.2 Excitation Response

Section 3.3.2 - 3.3.5 shows how the simulation of the load shaker was done using the FE model and MATLAB. With the FRF given by BRIGADE and a computed linear sweep excitation force the response from the Pershagen Bridge model was calculated. The excitation was supposed to give accelerations of 1 m/s² with a sweep rate as high as possible. As presented in Figure 4.1, a sweep rate of 0.167 Hz/s and a load amplitude of 25 kN is enough. The maximum acceleration reaches 1 m/s² with well defined resonance peaks. The FRF is found to have resonance peaks at an amplitude of about $5 \cdot 10^{-5}$ m/Ns².
Table 4.1. Modes of interest with mode shape descriptions.

| No. | Frequency | Description     | No.  | Frequency | Description     |
|-----|-----------|-----------------|-----|-----------|-----------------|
| 4   | 5.10 Hz   | Bending         | 28  | 25.23 Hz  | Torsion         |
| 5   | 8.01 Hz   | Torsion         | 29  | 26.03 Hz  | Bending/Torsion |
| 6   | 9.12 Hz   | Bending         | 30  | 26.23 Hz  | Bending/Torsion |
| 7   | 9.69 Hz   | Bending         | 31  | 29.02 Hz  | Torsion         |
| 10  | 12.38 Hz  | Torsion         | 32  | 31.01 Hz  | Bending         |
| 11  | 13.08 Hz  | Torsion         | 34  | 32.53 Hz  | Bending/Torsion |
| 12  | 14.04 Hz  | Torsion         | 35  | 33.50 Hz  | Torsion         |
| 13  | 14.32 Hz  | Torsion         | 37  | 34.65 Hz  | Bending/Torsion |
| 14  | 14.77 Hz  | Bending/Torsion | 41  | 39.10 Hz  | Torsion         |
| 15  | 15.19 Hz  | Bending/Torsion | 42  | 39.30 Hz  | Bending/Torsion |
| 22  | 17.96 Hz  | Torsion         | 43  | 39.44 Hz  | Bending/Torsion |
| 23  | 21.01 Hz  | Bending/Torsion | 45  | 44.00 Hz  | Torsion         |
| 25  | 24.48 Hz  | Bending/Longitudinal | 46  | 44.16 Hz  | Bending/Torsion |
| 26  | 24.60 Hz  | Bending/Torsion | 47  | 44.99 Hz  | Bending/Torsion |
| 27  | 25.20 Hz  | Bending/Torsion |

Figure 4.1. Simulated FRF and acceleration using a linear sweep excitation load of 25 kN, sweep rate of 0.167 Hz/s with frequency range of 0 - 50 Hz.
4.2 Full Scale Tests on the Pershagen Bridge

4.2.1 Planned Instrumentation

The method used for calculating the optimal excitation position (the \(RI\) factor) was presented in Section 3.3.3. The \(RI\) factor was calculated along a line positioned at \(y = 3.85\) m on the bridge deck. This line was chosen since it gave a good response for both bending and torsional modes. The \(RI\) factor was then calculated using 12 different mode combinations for the first 20 eigenmodes presented in Table 4.1. The results are shown in Figure 4.2.

After investigating the \(RI\) factor for the different mode combinations, it was discovered that the best position for excitation was \(x = 19.1\) m along the bridge deck (5.35 m from the left middle support). This position is marked in Figure 4.2.

![Figure 4.2. \(RI\) factor calculated for twelve different combinations of modes. Optimal excitation position is marked with a star at 19.1 m along the bridge deck.](image-url)
As can be seen it is a good position for more or less all combinations. When only taking the lower mode numbers into consideration it gives acceptable response, but not as good for larger mode numbers. However, since there were premonitions that the lower mode numbers would be easier to detect during the field test, it was chosen to be the best position. Also, as presented in Figure 4.2, no combination gives a zero value, which means that there will at least be some response regardless of mode number. In Figure 4.3 the planned excitation position can be seen in relation to the bridge.

In Section 3.1.3 the general idea behind the instrumentation for the pilot tests was described. The planned positioning of the accelerometers are presented in Figure 4.4. The general idea behind the positioning of the accelerometers was to detect as many mode shapes and eigenfrequencies as possible by using one set-up.

Several tests with different input parameters were planned (see Table 4.2). The first three tests were planned to be performed using the same load amplitude and frequency range, but with different sweep times, and a minimum pre-stress force amplitude of 5 kN. The results from these tests will be analysed on site, and the sweep time that gives a well-defined response with the shortest time will be the one used for the remaining tests. The starting frequency will be at 3 Hz in order to avoid the 1st longitudinal mode, which is expected to be somewhere around 1 - 2 Hz. The reason for avoiding this is that it could potentially harm the load shaker, and this mode is not of interest for the experiment.
4.2. FULL SCALE TESTS ON THE PERSHAGEN BRIDGE

Table 4.2. Planned input parameters for the different tests on the Pershagen Bridge.

| Test number | Sweep rate | Pre-stress load | Load amplitude | Frequency range |
|-------------|------------|----------------|----------------|-----------------|
| 1           | 0.05 Hz/s  | -6 kN          | 1 kN           | 3 - 20 Hz       |
| 2           | 0.03 Hz/s  | -6 kN          | 1 kN           | 3 - 20 Hz       |
| 3           | 0.01 Hz/s  | -6 kN          | 1 kN           | 3 - 20 Hz       |
| 4           | -10 kN     | 5 kN           | 3 - 50 Hz      |
| 5           | -25 kN     | 20 kN          | 30 - 50 Hz     |

The last two tests will be performed with a varying load amplitude in order to see what influence this has on the results. If the behaviour of the bridge is not linear, then this should affect the eigenfrequencies and the damping of the bridge. The frequency range will also be larger for these tests.

4.2.2 The Pilot Tests

The pilot tests of the load shaker on the Pershagen bridge were performed on June 3, 2015. The dirt road crossing underneath the bridge turned out to be quite narrow, with a ditch on one side of the road and sharp rocks on the other. Because of this it was difficult to use the theoretical optimal excitation position that had been determined. Instead, the load shaker was placed according to Figure 4.5, which also proved to be a good excitation position for the different mode combinations.

![Figure 4.5. Position of the load shaker during the pilot test at Pershagen Bridge.](image)

It also was demanding and time consuming to put the accelerometers in the exact positions that had been decided beforehand, and since this was the pilot test of the load shaker a lot of time and focus had to be put into assembling the device. Therefore, some compromises had to be done. Because of the time-limit nine accelerometers were used instead of twelve. They were placed according to Figure 4.6. The reason for not using a symmetric set-up was that the aerial work platform that was used for placing the accelerometers had limited range, and the narrow dirt road underneath the bridge made it difficult to position the vehicle.
Figure 4.6. The final instrumentation used at the pilot tests on the Pershagen Bridge. The accelerometers are marked as crossed circles labeled $a_i$ and the load shaker is marked as a crossed rectangle.

Because of the time limit and the compromises that had to be made, the planned tests were not performed. Instead, four different tests were performed with different input parameters. Test 1 was done before all accelerometers had been put in place in order to see that the device was working correctly, but it will still be considered here. The input parameters for the performed tests are presented in Table 4.3. The sampling frequency for all tests was 600 Hz.

Table 4.3. Input parameters for the different tests.

| Test number | Accelerometers used | Sweep increment | Pre-stress load | Load amplitude | Frequency range |
|-------------|---------------------|-----------------|-----------------|----------------|----------------|
| 1           | $a_1, a_6, a_7, a_9, a_{10}, a_{12}$ | 0.05 Hz/s | -10 kN | 5 kN | 3 - 20 Hz |
| 2           | $a_1, a_2, a_3, a_6, a_7, a_9, a_{10}, a_{12}$ | 0.05 Hz/s | -15 kN | 5 kN | 3 - 50 Hz |
| 3           | $a_1, a_2, a_3, a_6, a_7, a_8, a_9, a_{10}, a_{12}$ | 0.05 Hz/s | -15 kN | 10 kN | 3 - 50 Hz |
| 4           | $a_1, a_2, a_3, a_6, a_7, a_8, a_9, a_{10}, a_{12}$ | 0.01 Hz/s | -15 kN | 10 kN | 5 - 10 Hz |

4.2.3 Measured Response

In Figure 4.7 the response from accelerometer $a_9$ during test 3 is shown, both in the time domain, $a(t)$, and the frequency domain, $A(\omega)$. The three large peaks that are visible at around 600 s, 750 s, and 800 s are train passages that occurred during testing. Unfortunately, no test was performed without any train passages, which
of course affected the measured response. On a positive note the passages did not seem to harm the load shaker.

![Figure 4.7. The response from accelerometer a₉ during test 3 in the time domain, a(t), and frequency domain, A(ω).](image)

Also, during test 3 the foundation of the load shaker started vibrating extensively at around 30 Hz and continued to do so until the end of the test, which also affected the result. This is clear when looking at the frequency domain in Figure 4.7, where there is a greater extent of disturbance from 30 Hz and upwards. Because of this issue, and also because of the train passages, it was decided to limit the frequency range of interest to 3 - 20 Hz for all tests.

To be able to remove influence of the train passages on the response, they were removed from the signal using an "inverted window". In Figure 4.8 a comparison is made for the response in frequency domain around a peak value before limiting the frequency range to 3 - 20 Hz and removing the train passages, as well as after. It is obvious that by removing the train and limiting the frequency range, the response is much smoother. This smoother response will be used for the evaluations hereinafter.

### 4.2.4 Frequency Response Functions and Eigenfrequencies

By using the method presented in Section 3.3.2, the FRF $H_A(\omega)$ could be evaluated for the different tests. In Figure 4.9 the difference between the acceleration in the frequency domain $A(\omega)$ and the FRF $H_A(\omega)$ is shown for all tests from accelerometer a₉. Since $A(\omega)$ depends on the input load it will scaled with regard to it, but $H_A(\omega)$ should be the same for all tests if the system is linear. As can be seen this is not the case, which will be discussed further later on.
Figure 4.8. Comparison of the response from accelerometer $a_9$ during test 3 in the frequency domain, $A(\omega)$, before (top) and after (bottom) removing the train passages and limiting the frequency range.

Figure 4.9. Comparison of the acceleration in the frequency domain $A(\omega)$ and the FRF $H_A(\omega)$ from accelerometer $a_9$. 
4.2. FULL SCALE TESTS ON THE PERSHAGEN BRIDGE

The eigenfrequencies were determined by looking at the peak values in the FRF from the tests. The accelerometers that were placed at the end-shields and close to the supports gave very little response, so the eigenfrequencies were determined from the other accelerometers.

In Figure 4.10 the FRF, $H_A(\omega)$, is shown for accelerometers $a_2$, $a_3$, $a_9$, and $a_{10}$. These accelerometers had the overall best response from the tests. As can be seen, two eigenfrequencies are clearly visible at around 7.8 Hz and 18.8 Hz for all FRF’s. A full estimation of eigenfrequencies is given in Table 4.4, and also listed are the accelerometers where the eigenfrequency in question could be detected. Note that at some frequencies in Figure 4.10 there are sudden drops in the response, which is where the train passages have been cut out.

Figure 4.10. FRF, $H_A(\omega)$, for four different accelerometers.
4.2.5 Damping Ratio

The damping ratio was evaluated from the full scale testing by using the half-power bandwidth method, presented in Section 2.1.1. Because the frequency range was limited to 3 - 20 Hz, it was decided to only evaluate the damping ratio for the 1st vertical bending frequency.

Since the response signal still contained some disturbance after limiting the frequency range, a smoothing was done in MATLAB by applying the built-in smooth function to the response. The smoothing method which was chosen for the response was local regression using weighted linear least squares and a 2nd degree polynomial model.

As shown in Figure 4.11 this gives a smooth response, but it also gives rise to a source of error. Because of the disturbance in the original signal there is no way to be completely sure that the actual response looks like the smoothed signal. However, a choice was made not to evaluate this source of error further within this master thesis.

The estimated damping ratios for all tests and all accelerometers have been gathered in Table 4.5. Figure 4.11 shows the FRF from accelerometer a9 for all tests around the 1st bending frequency, as well as the corresponding smoothed signal and the parameters used for evaluating the damping ratio. Tests 1 and 2 gave a similar damping ratio of \(\sim 1.25 \%\) and a natural frequency of 7.86 Hz, while tests 3 and 4 gave a damping ratio of \(\sim 1.55 \%\) and a natural frequency of 7.80 Hz. As can be seen in Table 4.3 tests 1 and 2 both had a load amplitude of 5 kN, while tests 3 and 4 both had an amplitude of 10 kN. This clearly shows that the bridge response is non-linear.
4.2. FULL SCALE TESTS ON THE PERSHAGEN BRIDGE

Table 4.5. Evaluated damping ratios for all tests and all accelerometers.

| Test number | Damping ratio, $\zeta$ [%] |
|-------------|---------------------------|
|             | $a_1$ | $a_2$ | $a_3$ | $a_6$ | $a_7$ | $a_8$ | $a_9$ | $a_{10}$ | $a_{12}$ |
| 1           | 1.27  | –     | –     | 1.28  | 1.28  | –     | 1.25  | 1.25     | 1.25     |
| 2           | 1.26  | 1.26  | 1.27  | 1.22  | 1.28  | 1.16  | 1.26  | 1.26     | 1.25     |
| 3           | 1.61  | 1.54  | 1.64  | 1.67  | 1.58  | 1.32  | 1.52  | 1.50     | 1.55     |
| 4           | 1.70  | 1.59  | 1.67  | 1.79  | 1.68  | 1.17  | 1.58  | 1.56     | 1.68     |

Figure 4.11. Evaluation of the damping ratio using accelerometer $a_9$.

4.2.6 Vertical Accelerations

Since test 3 and 4 both had a load amplitude of 10 kN, the accelerations were largest during those tests. The accelerations were also considerably larger for the middle span accelerometers $a_3$, $a_9$, and $a_{10}$, since the response was largest at the 1st bending mode frequency. Because of this, only these accelerometers were considered when evaluating the maximum acceleration. The acceleration response from
these accelerometers are presented in Figure 4.12.

The largest accelerations that appeared were 0.681 m/s$^2$, which happened for accelerometer $a_{10}$ in test 4. For test 3 the largest accelerations were 0.612 m/s$^2$, which shows that a slower sweep rate gives a larger response. The accelerations did not reach our pre-defined goal of 1 m/s$^2$, but then again only 10 kN were used as maximum load amplitude for these tests.

![Acceleration response with corresponding sweep frequency from three different accelerometers.](image)

**Figure 4.12.** The acceleration response with the corresponding sweep frequency from three different accelerometers.

### 4.2.7 Mode Shapes

The mode shapes were also evaluated in the frequency range of 3 - 20 Hz. By using the response in the time domain combined with the position of the accelerometers, it was possible to plot the movements in the bridge deck in real time. By comparing with the evaluated eigenfrequencies the mode shapes could be estimated. Since there was a limited number of accelerometers and their positions were not entirely optimal, it proved difficult to evaluate the actual mode shapes.
4.2. FULL SCALE TESTS ON THE PERSHAGEN BRIDGE

Figure 4.13 shows the mode shapes for four of the estimated eigenfrequencies in test 3. It is apparent that the response is quite good, and it would be possible to receive very good estimates of the mode shapes with a larger number of accelerometers, or by using reference accelerometers and performing tests with several accelerometer set-ups. However, from the information at hand it is difficult to study any of the mode shapes, except for the mode shape at 7.8 Hz which is obviously the 1st vertical bending mode. This can be said with certainty, since the response is large and the edge beams are moving in phase with each other.

![Figure 4.13. Four mode shapes evaluated from test 3 in the frequency range of 3 - 20 Hz.](image)

4.2.8 Comparison to Free Vibration Test

The main reason for the development of the load shaker was to be able to analyse the dynamic properties of a bridge in a reliable manner. As mentioned in Chapter 2, the common way of analysing the dynamic properties of railway bridges is by analysing the free vibrations after a passing train. In Figure 4.14 the measured acceleration for
train passages are presented from an earlier measurement of the Pershagen Bridge. As can be seen, the free vibration after these passages are only one second long with an acceleration magnitude near the noise. In the frequency domain (Figure 4.15) the 1\textsuperscript{st} eigenfrequency can be identified at \(~8\) Hz, but disturbing effects from noise and internal vibrations of the train are present. The damping of the 1\textsuperscript{st} eigenfrequency would be hard to estimate correctly with this kind of data. Looking at the four controlled excitation tests it is evident that the disturbing effects are gone, and the peak of the 1\textsuperscript{st} eigenfrequency is well suited for evaluating the damping, as shown in Section 4.2.5.

Another advantage with the load shaker is that the applied excitation load is pre-defined and measured, giving the possibility to scale the response with regard to the applied load, according to Equation (3.1), which in turn gives the FRF of the structure. This is not possible to do when analysing train passages without any simulation of the resulting load from a train.

**Figure 4.14.** Top figure: Entire response from two train passages. Bottom figure: Free vibration response from the same two train passages.
Figure 4.15. Top figure: $A(\omega)$ from the free vibrations from six different train passages. Bottom figure: $H_A(\omega)$ from the four controlled excitation tests.
Chapter 5

Discussion

Using the hydraulic load shaker in order to evaluate the dynamic properties of the Pershagen Bridge proved to be a very effective method. As presented in Section 4.2.8, the response from controlled excitation is much more well-defined compared to measurements from free vibration tests. Controlled excitation gives the possibility to evaluate the FRFs, mode shapes and the damping ratios.

The information used to determine all the eigenfrequencies in the frequency range of interest was insufficient. Since only one excitation position was used, the influence of this position could therefore not be evaluated. Furthermore, the positioning of the accelerometers also have influence on which eigenfrequencies are possible to detect. If the test could be performed over a longer time period, it would be possible to perform several tests with different excitation positions and accelerometer set-ups.

The FRFs obtained from these tests were nevertheless well-defined, and from them it was obvious that the eigenfrequencies as well as the damping ratio vary with the applied load amplitude. This is a very interesting result, since it means that the bridge behaves in a non-linear manner. According to Ülker-Kaustell and Karoumi (2013) non-linear effects are mainly caused by the hysteretic effects induced by the friction in bridge bearings, which could be the case for this bridge.

In Figure 5.1 the effects from the load amplitude as well as the sweep rate are clearly visible. In the linear model, the accelerations are linearly dependent of the load amplitude. A doubling of the load amplitude with the same sweep rate results in a doubling of the acceleration. However, when observing the acceleration from the pilot tests around the 1st bending frequency the acceleration are not strictly linear to the load amplitude. A doubling of the load does not result in a doubling of the acceleration and the resonance peak is slightly offset. This is even more clear in Figure 5.2 where in the model a change of the load amplitude and sweep rate only affects the amplitude which is clearly not the case for the tests. However, a change in the sweep rate does not result in a change of the eigenfrequency, which can be seen when comparing test 3 and 4.

A way of getting even better response would be to first perform sweeps with a
sufficient sweep rate in order to receive an estimate of the eigenfrequencies, and then do a sweep with a very slow sweep rate over small intervals close to the estimated eigenfrequencies. This would give a very fine result, and would also help when determining the damping ratio. Also, since smoothing of the FRF was used in order to evaluate the damping ratio, there is a source of error in the results. With a slower sweep rate, the response would be more well-defined and then the damping ratio could be evaluated with a greater certainty.

Furthermore, by performing more tests with different load amplitudes, more information regarding the non-linear behaviour could be evaluated. Figure 5.3, which shows the FRF for all tests, clearly shows this non-linear behaviour. When augmenting the load amplitude the amplitude of the FRF gets smaller which indicates that there are energy dispersing the system. As presented in Section 4.2.5 the damping ratio increases with the load amplitude, which results in a lower energy content. This is probably due to an increased amount of kinetic energy in the bridge, especially at the supports, which in turn increases the friction and results in heat energy losses in the system.

Figure 5.1. Comparison of the accelerations in time domain $a(t)$ for accelerometer $a_9$ and the corresponding position in the 3D model.
The mode shapes that were evaluated from the tests were not very useful. Since the number of accelerometers were limited and since the positioning was not optimal for this purpose, the only mode shape that could be determined with certainty was the 1st vertical bending mode. However, if new tests were performed with more accelerometers, the mode shapes could probably be determined. An alternative is to perform several tests with reference accelerometers, which would make it possible to move the other accelerometers around in different set-ups. Also, to be able to distinguish the bending and torsional mode shapes from each other, accelerometers should be placed along the middle of the bridge deck. This would however mean that the accelerometers either need to be glued to the bottom of the bridge deck.
or placed on top of the bridge deck. This method would require a track guard to oversee the operation, and because of the ballast on the bridge deck this might not give a good response on many bridges. However, this will have to be decided from case to case.

5.1 Conclusion

From the FE model estimations and measurements from the pilot test it can be concluded that

- The dynamic response is effectively evaluated with the steady-state method.
- The dynamic behaviour of the Pershagen Bridge is non-linear. An increase of the load amplitude results in a larger damping ratio and a lower eigenfrequency.
- The sweep rate does not appear to have any influence on the dynamic properties, but does have influence on the quality of the measurements.
- Performing sweeps with a slow sweep rate over short frequency ranges close to estimated eigenfrequencies appears to be a good method in order to obtain accurate dynamic properties.
- The 1st vertical bending mode and its dynamic properties of the Pershagen Bridge could be identified.
Bibliography

Andersson, A., Karoumi, R., Pacoste, C., Johansson, C., Wiberg, J., and Ülker Kaustell, M. (2010). Höghastighetsprojekt-bro delrapport 1: Befintliga krav och erfarenheter samt parameterstudier avseende dimensionering av järnvägsbroar för farter över 200 km/h. Report 139, Brobyggnad KTH Byggvetenskap.

Andersson, A., Karoumi, R., Pacoste, C., Solat Yavari, M., Martino, D., Arvidsson, T., and Johansson, C. (2011). Höghastighetsprojekt-bro inventering av järnvägsbroar för ökad hastighet på befintliga banor. Report 141, Brobyggnad KTH Byggvetenskap.

Bachmann, H., Ammann, W. J., Deischl, F., Eisenmann, J., Floegl, I., Hirsch, G. H., Klein, G. K., Lande, G. J., Mahrenholtz, O., Natke, H. G., Nussbaumer, H., Pretlove, A. J., Rainer, J. H., Saemann, E.-U., and Steinbeisser, L. (1995). Vibration Problems in Structures.

Banverket (1991). Relationsritning - Bro vid Pershagen.

Banverket (2008). Broreregler för nybyggnad - BV-Bro.

Brandt, A. (2011). Noise and Vibration Analysis : Signal Analysis and Experimental Procedures. John Wiley & Sons, Hoboken, NJ, USA.

Chopra, A. K. (2012). Dynamics of Structures - Theory and Applications to Earthquake Engineering. 4 edition.

Cook, R. D., Malkus, D. S., Plesha, M. E., and Witt, R. J. (2002). Concepts and Applications of Finite Element Analysis. John Wiley & Sons, 4th edition.

Grosel, J., Sawicki, W., and Pakos, W. (2014). Application of classical and operational modal analysis for examination of engineering structures. Procedia Engineering, 91(0):136–141.

Ülker-Kaustell, M. and Karoumi, R. (2013). Influence of rate-independent hysteresis on the dynamic response of a railway bridge. Internation Journal of Rail Transportation, 1(4):237–257.

SIS (2002). Eurocode 1: Actions on structures.
SIS (2005). Eurocode 2: Design of concrete structures.

SIS (2006). Eurocode - Basis of structural design.

Viswanathan, M. (2014). Chirp signal - Frequency sweeping - FFT and power spectral density.

Zwolski, J. and Bień, J. (2011). Modal analysis of bridge structures by means of forced vibration tests. *Journal of Civil Engineering and Management*, 17(4):590–599.
Appendix A

3D Model Part Geometries

Figure A.1. The geometric part of the bridge deck seen as an elevation.

Figure A.2. The geometric part of the bridge deck seen as a section.

Figure A.3. The geometric part of the end-shield and the wing-wall seen in elevation and plane.
Figure A.4. The geometric part of the four support pairs seen in elevation and plane.
# Appendix B

## Eigenfrequencies and Mode Shapes

Table B.1. Evaluated eigenfrequencies in BRIGADE.

| No. | Frequency | No. | Frequency | No. | Frequency | No. | Frequency |
|-----|-----------|-----|-----------|-----|-----------|-----|-----------|
| 1   | 1.06 Hz   | 14  | 14.77 Hz  | 27  | 25.20 Hz  | 40  | 38.89 Hz  |
| 2   | 3.02 Hz   | 15  | 15.19 Hz  | 28  | 25.23 Hz  | 41  | 39.10 Hz  |
| 3   | 4.36 Hz   | 16  | 15.68 Hz  | 29  | 26.03 Hz  | 42  | 39.30 Hz  |
| 4   | 5.10 Hz   | 17  | 15.90 Hz  | 30  | 26.23 Hz  | 43  | 39.44 Hz  |
| 5   | 8.01 Hz   | 18  | 15.92 Hz  | 31  | 29.02 Hz  | 44  | 40.78 Hz  |
| 6   | 9.12 Hz   | 19  | 15.99 Hz  | 32  | 31.01 Hz  | 45  | 44.00 Hz  |
| 7   | 9.69 Hz   | 20  | 16.34 Hz  | 33  | 31.85 Hz  | 46  | 44.16 Hz  |
| 8   | 10.14 Hz  | 21  | 17.29 Hz  | 34  | 32.53 Hz  | 47  | 44.99 Hz  |
| 9   | 10.99 Hz  | 22  | 17.96 Hz  | 35  | 33.50 Hz  | 48  | 45.61 Hz  |
| 10  | 12.38 Hz  | 23  | 21.01 Hz  | 36  | 33.63 Hz  | 49  | 46.10 Hz  |
| 11  | 13.08 Hz  | 24  | 23.92 Hz  | 37  | 34.65 Hz  | 50  | 49.04 Hz  |
| 12  | 14.04 Hz  | 25  | 24.48 Hz  | 38  | 37.53 Hz  | 51  | 49.39 Hz  |
| 13  | 14.32 Hz  | 26  | 24.60 Hz  | 39  | 37.60 Hz  | 52  | 49.51 Hz  |
APPENDIX B. EIGENFREQUENCIES AND MODE SHAPES

Figure B.1. Mode shape of interest 1 - 10, which were used to find the optimal excitation position.
Figure B.2. Mode shape of interest 10 - 20, which were used to find the optimal excitation position.
Appendix C

Python and MATLAB scripts

C.1 3D Model Variables

The variables imported to the Python script used for building the 3D model in BRIGADE.

```python
from math import *
# Input variables for the 3D-model of the
# Pershagen bridge.

completedProcess = 'BrigadeCompleted'

# model Names
T = [60]
loadPos = [(19.1, 3.85, 0)]
loads = [1]
seedSize = [0.5]

modelNames = []
for i in loadPos:
    modelNames.append('3D-x=' + str(int(i[0])) + ' -y=' + str(int(i[1])) + ' - ' + str(int(seedSize[0]*1000)) + ' mm - ' + str(T[0]) + 's')

# The lenght of the bridges spans
spans = [2.5, 11.1, 18.4, 11.1, 2.5]

# The X-coordinations of end-points and supports
coordX = [0]
i = 0
for span in spans:
    b = coordX[i]
    coordX.append(b + span)
i += 1
```

# Bridge deck

# The bridge deck is composed of reinforced-concrete and ballasted track. The chosen densities are:
# - Concrete : 2500kg/m^3
# - Ballast : 2000kg/m^3 (The thickness of the
APPENDIX C. PYTHON AND MATLAB SCRIPTS

# ballast is assumed to be uniformly 0.6m over
# the deck.(in 21.122 i BV-bro utgava 9))
# The density of the deck geometry is the average of the
# concrete and the ballast, though only the geometric
# properties of the concrete are modeled.

# Names
namepartD='part_Deck'
namesecD='section_Deck' # Deck section name
namesetD='set_Deck' # Deck set name
nameinstD='inst_Deck'

# Geometry
hB =0.6 # Ballast height
hD = 1.1 # Deck height
wD = 11.9 # Deck width
lD = 45.6 # Deck length

# Offset distance from bridge deck to supports 2 and 3
offD =0.4

# Height of bearings
hBear = 0.1

# The support radius
rSupport =0.4 # [m]

# Set names for the connections
namesetConnectionD=['EndShieldConnection_1', 'EndShieldConnection_2']
cooFindAtConnectionD=[(0, wD/2, 0), (coordX[-1], wD/2, 0)]

##############################################################################
# End-Shields
##############################################################################

namepartEND=['part_EndShield', 'part_WingWall']
namesecEND=['sec_EndShield', 'sec_WingWall']
namesetEND=['set_EndShield', 'set_WingWall']
nameinstEND=['inst_EndShield', 'inst_WingWall']

hEndShield =2.6
wEndShield=wD

tEndShield=[0.5, 0.35]
cooWingWall=[(0,0),(0,2.3),(3.15,2.3),(3.15,1.3),(0,0)]

# Coordinates to translate the End-Shields
cooTransEND=[(-tEndShield[0]/2, 0, hD/2-hEndShield), (lD+tEndShield[0]/2, 0, hD/2-hEndShield)]

# Coordinates to translate the wing-walls
cooTransWingWall=[(-tEndShield[0], tEndShield[1]/2, -cooWingWall[1][1]+hD/2), (-tEndShield[0], wD-tEndShield[1]/2, -cooWingWall[1][1]+hD/2), (1D+tEndShield[0], tEndShield[1]/2, -cooWingWall[1][1]+hD/2), (1D+tEndShield[0], wD-tEndShield[1]/2, -cooWingWall[1][1]+hD/2)]

# Partitioning coordinates and new set names (End Shields)
namesetConnectionEND=['DeckConnection', 'WingWallConnection_1', 'WingWallConnection_2']
cooConnectionEND=[(0, hEndShield), (wEndShield, hEndShield-hD), (0, 0.3), (tEndShield[1], hEndShield), (wEndShield, 0.3),

VIII
### C.1. 3D Model Variables

$$(v_{EndShield} - t_{EndShield}[1], h_{EndShield})$$

```python
cooFindAtConnectionEND = [(cooConnectionEND[0][0]+0.001, cooConnectionEND[0][1]-0.001, 0),
(cooConnectionEND[1][0]/2, cooConnectionEND[1][1]+0.001, 0),
(cooConnectionEND[2][0]+0.001, cooConnectionEND[2][1]+0.001, 0),
(cooConnectionEND[4][0]-0.001, cooConnectionEND[4][1]+0.001, 0)]
```

# Set names for the wingwalls connection to the end shield
```python
namesetConnectionWingWall = ['EndShieldConnection_1', 'EndShieldConnection_2']
cooFindAtConnectionWingWall = (0, cooWingWall[1][1]/2, 0)
```

# Supports

```python
# Local geometry of the supports
cooSup = 
[((0, 0), (0, 3.95), (5, 0), (5, 3.95), (0, 3.425), (5, 3.425)),
((0, 0), (0, 7.2), (5, 0), (5, 7.2), (6, 0), (6, 6)),
((0, 0), (5.4), (5, 0), (5, 5.4), (0, 4.875), (5, 5.4875))]
```

namepartSup = ['part_FixedPinnedSupport_1', 'part_FixedPinnedSupport_4']

```
namepartBear = 'Bearing'
rColumnSup = [1.2/2, 1.35/2, 1.2/2]  # Radius of the columns
recBeamSup = [(1.05, 1.2), (1.2, 1.35), (1.05, 1.2)]  # Height and depth of horizontal beam
massBear = 100  # Mass of the bearing parts
namemassBear = 'pointMass_Bearing'
```

# Support 'Vectors'
```python
nameproSupport = ['profile_circPillarFP', 'profile_recPillarFP', 'profile_circPillarFF', 'profile_recPillarFF']
geoSupport = [rColumnSup[0], recBeamSup[0], rColumnSup[1], recBeamSup[1]]
namesecSupport = ['section_circPillarFP', 'section_recPillarFP', 'section_circPillarFF', 'section_recPillarFF']
```

# Vectors to help find corresponding values in for-loops
```python
forCooSup = []
forPartSup = []
for i in range(len(cooSup)):
    if i == 1:
        for j in range(2):
            forCooSup.append(cooSup[i])
            forPartSup.append(namepartSup[i])
    else:
        forCooSup.append(cooSup[i])
```
forPartSup.append(namepartSup[i])

# Instance vector
nameinstSup = ['inst_FixedPinnedSupport_1', 'inst_FixedFixedSupport_2', 'inst_FixedFixedSupport_3', 'inst_FixedFixedSupport_4']

# Set names for the support surfaces and nodes
namesetConnectionSup = ['SFP_1_1', 'SFP_1_2', 'SFF_2_1', 'SFF_2_2', 'SFF_3_1', 'SFF_3_2', 'SFP_4_1', 'SFP_4_2']

# Global Z-coordinates of pillar-slab position
coordZSup = [-(cooSup[0][1][1]+hD/2+hBear), -(cooSup[1][1][1]+hD/2+offD+hBear), -(cooSup[1][1][1]+hD/2+offD+hBear), -(cooSup[2][1][1]+hD/2+hBear)]

# Coordinates for the support surfaces and nodes
cooFindAtConnectionSup = []
for i in range(len(forCooSup)):
    cooFindAtConnectionSup.append((coordX[i+1], (wD-forCooSup[i][2][0])/2, 0))
    cooFindAtConnectionSup.append((coordX[i+1], (wD+forCooSup[i][2][0])/2, 0))

cooFindAtNodeConnectionSup = []
for coord in cooSup:
    for i in range(2):
        cooFindAtNodeConnectionSup.append((coord[i*2+1][0], coord[i*2+1][1], 0))

namesetNodeConnectionSup = ['nodeSFP_1_1', 'nodeSFP_1_2', 'nodeSFF_1', 'nodeSFF_2', 'nodeSFP_4_1', 'nodeSFP_4_2']

namePartSetNodesSup = []
for i in range(len(namepartSup)):
    if i == 1:
        for j in range(2):
            namePartSetNodesSup.append(namepartSup[i])
    else:
        namePartSetNodesSup.append(namepartSup[i])

# Coordinates for the bearings
cooBear = []
for i in range(4):
    for j in range(2):
        if i < 1 or i > 2:
            cooBear.append((cooFindAtConnectionSup[i*2+j][0], cooFindAtConnectionSup[i*2+j][1], -(hD/2)))
        else:
            cooBear.append((cooFindAtConnectionSup[i*2+j][0], cooFindAtConnectionSup[i*2+j][1], -(hD/2+offD)))

# Material definitions

# Names
namematD = 'material_Deck'
# Deck material name

X
C.1. 3D MODEL VARIABLES

matDdescription=['Averaged concrete-ballast','Only concrete']  # Deck material description

# Density
rhoB = 2000  # Ballast density
rhoC = 2500  # Reinforced concrete density

# Averaged density
mB=hB*wD*rhoB  # Ballast mass
mC=hD*wD*rhoC  # Concrete mass
mD=mB+mC  # Total deck mass
rhoD =mD /( hD*wD)  # Deck density

EC =210000000000.0  # E-module, concrete, 60% of E-module for K40, (Table 3.1 in EN1992-1-1).
PC =0.2  # Poisson ratio, concrete, (3.1.3 (4) in EN1992-1-1)
xic =0.015  # Damping ratio (Table C.1 in Vibration Problems in Structures, assumed value)

# Material 'Vectors'
nameMat=[namematD,namematC]  # All material names
rho=[rhoD,rhoC]  # All material densities

# Creating interaction, ties, between End-Shields, Wing-Walls and Bridge Deck

nameInstInteractionTie=[nameinstD, nameinstEND[0]+'_1', nameinstD, nameinstEND[0]+'_2', nameinstD, nameinstEND[1]+'_1', nameinstEND[0]+'_1', nameinstEND[1]+'_2', nameinstEND[0]+'_2', nameinstEND[1]+'_3', nameinstEND[0]+'_3', nameinstEND[1]+'_4', nameinstEND[0]+'_4']

nameSetInteractionTie=[namesetConnectionD[0], namesetConnectionEND[0], namesetConnectionD[1], namesetConnectionEND[1], namesetConnectionD[2], namesetConnectionEND[2], namesetConnectionWingWall[0], namesetConnectionWingWall[1], namesetConnectionWingWall[2], namesetConnectionWingWall[3]]

nameInteractionTie=['Deck-ES1','Deck-ES2','ES1-WW1','ES1-WW2','ES2-WW3','ES2-WW4']

# Creating interaction, Couplings, between supports and deck

nameInstInteractionCoupling1=[]
for i in range(len(nameinstSup)):
    for j in range(2):
        nameInstInteractionCoupling1.append(nameinstSup[i])
        nameInstInteractionCoupling1.append(nameinstBear+'_'+str(i+1)+'_'+str(j+1))

nameSetInteractionCoupling1=[]
for i in range(len(namesetConnectionSup)):
APPENDIX C. PYTHON AND MATLAB SCRIPTS

```python
if i < 4:
    nameSetInteractionCoupling1.append(namesetNodeConnectionSup[i])
else:
    nameSetInteractionCoupling1.append(namesetNodeConnectionSup[i-2])
nameSetInteractionCoupling1.append(namesetBear)

nameInteractionCoupling1 = []
for name in namesetConnectionSup:
    nameInteractionCoupling1.append('Coupling_1_ '+name)

nameInstInteractionCoupling2 = []
for i in range(len(nameinstSup)):
    for j in range(2):
        nameInstInteractionCoupling2.append(nameinstBear+'_ '+str(i+1)+'_ '+str(j+1))
        nameInstInteractionCoupling2.append(nameinstD)

nameSetInteractionCoupling2 = []
for i in range(len(namesetConnectionSup)):
    nameSetInteractionCoupling2.append(namesetBear)
    nameSetInteractionCoupling2.append(namesetConnectionSup[i])

nameInteractionCoupling2 = []
for name in namesetConnectionSup:
    nameInteractionCoupling2.append('Coupling_2_ '+name)

# Partitioning the deck for node pos

PartitionsX = [4, 8, 16]  
splitPosX = [0]
i = 0
check = 0
for span in spans:
    if span == spans[0]:  
        noPartitions = PartitionsX[0]
        dx = (span-rSupport)/noPartitions
        for coordX[i] == 0:
            for j in range(noPartitions-1):
                posX = (j+1)*dx+coordX[i]
                splitPosX.append(posX)
                splitPosX.append(coordX[i+1]-rSupport)
        else:
            splitPosX.append(coordX[i]+rSupport)
            for j in range(noPartitions-1):
                posX = (j+1)*dx+coordX[i]+rSupport
                splitPosX.append(posX)
    elif span == spans[1]:
        splitPosX.append(coordX[i]+rSupport)
        noPartitions = PartitionsX[1]
        dx = (span-2*rSupport)/noPartitions
        for j in range(noPartitions-1):
            posX = (j+1)*dx+coordX[i]+rSupport
            splitPosX.append(posX)
        else:
            splitPosX.append(coordX[i+1]-rSupport)
    else:
        splitPosX.append(coordX[i]+rSupport)
```
C.1. 3D MODEL VARIABLES

```
noPartitions = PartitionsX[2]
dx = (span-2*rSupport)/noPartitions
for j in range(noPartitions-1):
posX = (j+1)*dx+coordX[i]+rSupport
if posX > loadPos[0][0] and check == 0:
splitPosX.append(loadPos[0][0])
splitPosX.append(posX)
check += 1
else:
splitPosX.append(posX)
splitPosX.append(coordX[i+1]-rSupport)
i += 1
splitPosX.append(lD)
spansY = [(wD-cooSup[0][2][0])/2, cooSup[0][2][0], (wD-cooSup[0][2][0])/2]
coordY = [0]
i = 0
for span in spansY:
b = coordY[i]
coordY.append(b+span)
i += 1
PartitionsY = [1, 2]
splitPosY = [0]
i = 0
for span in spansY:
    if span == (wD-cooSup[0][2][0])/2:
        noPartitions = PartitionsY[0]
dy = (span-rSupport)/noPartitions
        if coordY[i] == 0:
            for j in range(noPartitions-1):
posY = (j+1)*dy+coordY[i]
splitPosY.append(posY)
splitPosY.append(coordY[i+1]-rSupport)
        else:
splitPosY.append(coordY[i]+rSupport)
        for j in range(noPartitions-1):
posY = (j+1)*dy+coordY[i]+rSupport
            splitPosY.append(posY)
        else:
splitPosY.append(coordY[i]+rSupport)
        noPartitions = PartitionsY[1]
dy = (span-2*rSupport)/noPartitions
        for j in range(noPartitions-1):
posY = (j+1)*dy+coordY[i]+rSupport
            splitPosY.append(posY)
splitPosY.append(coordY[i+1]-rSupport)
        i += 1
splitPosY.append(wD)
nameSetResult = ['resultSouth', 'resultMid', 'resultNorth']
```

```
# Step definitions
steps = ['Initial', 'Eigenfrequencies', 'Steady-state-direct', 'Dynamic']
nameFieldOutputRequest = ['F-Output-1', 'F-Output-2', 'F-Output-3']
nameLoad = ['SteadyState Load', 'Dynamic Load']
namesetLoad = 'Load Position'
```
# Eigenfrequencies, variables
minFreq = 0
maxFreq = 50

# Steady-state, variables
lowLimit = 0
upLimit = 50
noSteps = []
for i in T:
    noSteps.append(i * (upLimit - lowLimit) + 1)
loadSize = loads

# Mesh definitions

namesAllParts = [namepartD]
for name in namepartEND:
    namesAllParts.append(name)
for name in namepartSup:
    namesAllParts.append(name)

seedSize = [0.5]
seedFactor_1 = []
seedFactor_2 = []
for seed in seedSize:
    seedFactor_1.append(seed * 0.7)
    seedFactor_2.append(seed * 0.5)
devFactor = 0.01
minSizeFactor = 0.1

C.2 3D Model Script

The Python script used for building the 3D model in BRIGADE.

# Obligatory imports for Brigade
from abaqus import *
from abaqusConstants import *
import bpCustomData
import createStepsModule
import printStepInfoModule
from caeModules import *
import dataManagement.registerKernelCommands
executeOnCaeStartup()
#: Executing "onCaeStartup()" in the site directory ...
bpCustomData.createBpCustomData()
mdb.customData.fileEvent = 0
bpCustomData.updateBpCustomData(
    guiDataStr='"inputReportData#GUIDATASEPARATOR#(dp0' +
    '#GUIDATASEPARATOR#")"
)
import assembly
import sketchLaneModule
import LL_axlePos.liveLoadPosUtils
from LL_axlePos.liveLoadPosSymConsts import *
import fbcAxesDisplayUtils
import Mdb()
#: A new model database has been created.
#: The model "Model-1" has been created.
C.2. 3D MODEL SCRIPT

```python
session.graphicsOptions.setValue('backgroundColor', '#000000 ')
session.journalOptions.setValue('replayGeometry', 'COORDINATE',
                                'recoverGeometry', 'COORDINATE')
```

Profiles are shown
```python
session.viewports['Viewport: 1'].partDisplay.setValue('renderBeamProfiles', 'ON',
                                                      'renderShellThickness', 'ON')
```

The Variables .py contains all the variables describing the model
```python
from sys import path
path.append('C:/Users/rbg/Dropbox/Exjobb_R&O/3 - Modell/3D_Model/Model_script')
from Variables import *
```

Changing the name of the first model
```python
mdb.models.changeKey(fromName='Model-1', toName=modelNames[0])
```

Creating new models for convergence analysis of different mesh sizes
```python
for i in range(len(modelNames) - 1):
    mdb.Model(name=modelNames[i + 1])
    bpCustomData.updateBpCustomData(guiDataStr='inputReportData
                                """GUIDATASEPARATOR#(dp0."
                                """GUIDATASEPARATOR#""
    mdb.customData.modelEvent=None
```

Creating the geometry of the model
```python
# Creates the bridge deck part
i = 0
for model in modelNames:
    s = mdb.models[model].ConstrainedSketch(name='__profile__',
                                              sheetSize=200.0)
    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
    s.setPrimaryObject(option='STANDALONE')
    s.rectangle(point1=(0.0, 0.0), point2=(1D, wD))
    p = mdb.models[model].Part(name=namepartD, dimensionality=THREE_D,
                                type=DEFORMABLE_BODY)
    p = mdb.models[model].parts[namepartD]
    p.BaseShell(sketch=s)
    s.unsetPrimaryObject()
    del mdb.models[model].sketches['__profile__']
```

Creates the different support parts
```python
for model in modelNames:
```
```python
```
APPENDIX C. PYTHON AND MATLAB SCRIPTS

i=0
for name in namepartSup:
s = mdb.models[model].ConstrainedSketch(name='__profile__',
   sheetSize=200.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
s.Line(point1=cooSup[i][0], point2=cooSup[i][1])
s.VerticalConstraint(entity=g.findAt((cooSup[i][0][0],
   cooSup[i][0][1]+0.001)),
   addUndoState=False)
s.Line(point1=cooSup[i][2], point2=cooSup[i][3])
s.VerticalConstraint(entity=g.findAt((cooSup[i][2][0],
   cooSup[i][2][1]+0.001)),
   addUndoState=False)
s.Line(point1=cooSup[i][4], point2=cooSup[i][5])
s.HorizontalConstraint(entity=g.findAt((cooSup[i][4][0]+0.001,
   cooSup[i][4][1])),
   addUndoState=False)
p = mdb.models[model].Part(name=name, dimensionality=THREE_D,
   type=DEFORMABLE_BODY)
s.unsetPrimaryObject()
del mdb.models[model].sketches['__profile__']
i+=1

# Creating the End-Shield Parts

# End-shield wall
for model in modelNames:
s = mdb.models[model].ConstrainedSketch(name='__profile__',
   sheetSize=200.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
s.rectangle(point1=(0.0, 0.0), point2=(wEndShield, hEndShield))
p = mdb.models[model].Part(name=namepartEND[0], dimensionality=THREE_D,
   type=DEFORMABLE_BODY)
p.BaseShell(sketch=s)
s.unsetPrimaryObject()
del mdb.models[model].sketches['__profile__']

# Wing wall
for model in modelNames:
s = mdb.models[model].ConstrainedSketch(name='__profile__',
   sheetSize=200.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
s.Line(point1=cooWingWall[0], point2=cooWingWall[1])
s.VerticalConstraint(entity=g.findAt((cooWingWall[0][0],
   cooWingWall[0][1]+0.001)),
   addUndoState=False)
s.Line(point1=cooWingWall[1], point2=cooWingWall[2])
s.HorizontalConstraint(entity=g.findAt((cooWingWall[1][0]+0.001,
   cooWingWall[1][1])),
   addUndoState=False)
s.PerpendicularConstraint(entity1=g.findAt((cooWingWall[0][0],
   cooWingWall[0][1]+0.001)),
   entity2=g.findAt((cooWingWall[1][0]+0.001,
   cooWingWall[1][1])),
   addUndoState=False)
C.2. 3D MODEL SCRIPT

```python
addUndoState=False)

s.Line(point1=cooWingWall[2], point2=cooWingWall[3])
s.VerticalConstraint(entity=g.findAt((cooWingWall[2][0],
cooWingWall[2][1]-0.001)),
addUndoState=False)

s.PerpendicularConstraint(entity1=g.findAt((cooWingWall[1][0]+0.001,
cooWingWall[1][1]),
entity2=g.findAt((cooWingWall[2][0],
cooWingWall[2][1]-0.001)),
addUndoState=False)

s.Line(point1=cooWingWall[3], point2=cooWingWall[4])
p = mdb.models[model].Part(name=namepartEND[1], dimensionality=THREE_D,
type=DEFORMABLE_BODY)
p = mdb.models[model].parts[namepartEND[1]]
p.BaseShell(sketch=s)
s.unsetPrimaryObject()
del mdb.models[model].sketches['__profile__']

# Creating the bearing points
for model in modelNames:
p = mdb.models[model].Part(name=namepartBear, dimensionality=THREE_D,
type=DEFORMABLE_BODY)
p.ReferencePoint(point=(0.0, 0.0, 0.0))
```

# Creating materials
```python
j=0
for model in modelNames:
i=0
for name in nameMat:
    mdb.models[model].Material(name=name,
description=matDdescription[i])
    mdb.models[model].materials[name].Density(table=((rho[i], ), ))
    mdb.models[model].materials[name].Elastic(table=((EC, PC), ))
    mdb.models[model].materials[name].Damping(structural=2*xiC)
i+=1
j+=1
```

# Creating the Deck shell section
```python
for model in modelNames:
    mdb.models[model].HomogeneousShellSection(name=namesecD, preIntegrate=OFF,
material=namematD, thicknessType=UNIFORM,
thickness=hD, thicknessField='',
idealization=NO_IDEALIZATION,
poissonDefinition=DEFAULT,
thicknessModulus=Hooke,
temperature=GRADIENT,
useDensity=OFF,
```
 integrationRule=SIMPSON,
 numIntPts=5)

# Creating the End-Shield wall section
# The material in the end-shield can be discussed about =)
for model in modelNames:
  i=0
  for name in namesecEND:
    (mdb.models[model].HomogeneousShellSection(name=name, preIntegrate=OFF,
      material=namematD, thicknessType=UNIFORM,
      thickness=tEndShield[i], thicknessField='',
      idealization=NO_IDEALIZATION,
      poissonDefinition=DEFAULT,
      thicknessModulus=None, temperature=GRADIENT,
      useDensity=OFF, integrationRule=SIMPSON,
      numIntPts=5))
    i+=1

# Creating the profile of the supports
for model in modelNames:
  for i in range(len(nameproSupport)/2):
    mdb.models[model].CircularProfile(name=nameproSupport[i*2],
      r=geoSupport[i*2])
    mdb.models[model].RectangularProfile(name=nameproSupport[i*2+1],
      a=geoSupport[i*2+1][1],
      b=geoSupport[i*2+1][0])

# Creating sections of the supports
for model in modelNames:
  i=0
  for name in namesecSupport:
    mdb.models[model].BeamSection(name=name, integration=DURING_ANALYSIS,
      poissonRatio=PC,
      profile=nameproSupport[i],
      material=namematC, temperatureVar=LINEAR,
      consistentMassMatrix=False)
    i+=1

# Assigning the shell section to the Deck
for model in modelNames:
  p = mdb.models[model].parts[namepartD]
  f = p.faces
  faces = f.findAt(((1D/2, wD/2, 0.0), ))
  region = p.Set(faces=faces, name=namesetD)
  p.SectionAssignment(region=region, sectionName=namesecD, offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='%',
    thicknessAssignment=FROM_SECTION)

# Assigning the shell section to the End-Shield
for model in modelNames:
  p = mdb.models[model].parts[namepartEND[0]]
  f = p.faces
  faces = f.findAt(((wEndShield/2, bEndShield/2, 0.0), ))
  region = p.Set(faces=faces, name=namesetEND[0])
  p.SectionAssignment(region=region, sectionName=namesecEND[0], offset=0.0,
    offsetType=MIDDLE_SURFACE, offsetField='%',
    thicknessAssignment=FROM_SECTION)

  p = mdb.models[model].parts[namepartEND[1]]
  f = p.faces
C.2. 3D MODEL SCRIPT

faces = f.findAt(((cooWingWall[2][0]/2, cooWingWall[2][1]/2, 0.0), ))
region = p.Set(faces=faces, name=namesetEND[1])
p.SectionAssignment(region=region, sectionName=namesecEND[1], offset=0.0, 
offsetType=MIDDLE_SURFACE, offsetField='',
thicknessAssignment=FROM_SECTION)

# Assign the sections to the support geometries!
for model in modelNames:
    i=0
    for coord in cooSup:
        p = mdb.models[model].parts[namepartSup[i]]
        e = p.edges
        edges = e.findAt(((coord[0][0], coord[0][1]+0.001, 0.0), ),
                         ((coord[1][0], coord[1][1]-0.001, 0.0), ),
                         ((coord[2][0], coord[2][1]+0.001, 0.0), ),
                         ((coord[3][0], coord[3][1]-0.001, 0.0), ))
        region = p.Set(edges=edges, name=namesetSupport[i*2])
        if i == 1:
            p.SectionAssignment(region=region, sectionName=namesecSupport[2],
                                offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='',
                                thicknessAssignment=FROM_SECTION)
        else:
            p.SectionAssignment(region=region, sectionName=namesecSupport[0],
                                offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='',
                                thicknessAssignment=FROM_SECTION)
p.assignBeamSectionOrientation(region=region, method=N1_COSINES,
                                   n1=(0.0,0.0,-1.0))
        edges = e.findAt(((coord[4][0]+0.001, coord[4][1], 0.0), ))
        region = p.Set(edges=edges, name=namesetSupport[i*2+1])
        if i == 1:
            p.SectionAssignment(region=region, sectionName=namesecSupport[3],
                                offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='',
                                thicknessAssignment=FROM_SECTION)
        else:
            p.SectionAssignment(region=region, sectionName=namesecSupport[1],
                                offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='',
                                thicknessAssignment=FROM_SECTION)
p.assignBeamSectionOrientation(region=region, method=N1_COSINES,
                                   n1=(0.0,0.0,-1.0))
i++

# Assigning a point mass to the bearings
for model in modelNames:
    p = mdb.models[model].parts[namepartBear]
    r = p.referencePoints
    refPoints=(r[r[1], ])
    region=p.Set(referencePoints=refPoints, name=namesetBear)
    (mdb.models[model].parts[namepartBear].engineeringFeatures
    .PointMassInertia(name=namemassBear, region=region, mass=massBear,
    alpha=0.0, composite=0.0))

# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
for model in modelNames:
    a = mdb.models[model].rootAssembly
    a.DatumCsysByDefault(CARTESIAN)
XIX
APPENDIX C. PYTHON AND MATLAB SCRIPTS

# Deck assembly
p = mdb.models[model].parts[namepartD]
a. Instance(name=nameinstD, part=p, dependent=ON)

# End-shield assembly
p = mdb.models[model].parts[namepartEND[0]]
for i in range(len(cooTransEND)):
    a. Instance(name='nameinstEND'+str(i+1), part=p, dependent=ON)
    a. rotate(instanceList=('nameinstEND'+str(i+1),
                            axisPoint=(0.0,0.0,0.0),
                            axisDirection=(1,0,0,0),
                            angle=90.0))
    a. rotate(instanceList=('nameinstEND'+str(i+1),
                            axisPoint=(0.0,0.0,0.0),
                            axisDirection=(0.0,0.0,1),
                            angle=90.0))
    a. translate(instanceList=('nameinstEND'+str(i+1),
                             vector=cooTransEND[i]))

# Wing-wall assembly
p = mdb.models[model].parts[namepartEND[1]]
for i in range(len(cooTransWingWall)/2):
    for j in range(2):
        a. Instance(name='nameinstEND'+str(i*2+j+1), part=p, dependent=ON)
        a. rotate(instanceList=('nameinstEND'+str(i*2+j+1),
                                axisPoint=(0.0,0.0,0.0),
                                axisDirection=(1.0,0.0,0.0),
                                angle=90.0))
        a. rotate(instanceList=('nameinstEND'+str(i*2+j+1),
                                axisPoint=(0.0,0.0,0.0),
                                axisDirection=(0.0,0.0,1.0),
                                angle=180.0*(i+1))
        a. translate(instanceList=('nameinstEND'+str(i*2+j+1),
                                 vector=cooTransWingWall[i*2+j]))

# Supports assembly
for i in range(len(forPartSup)):
    p = mdb.models[model].parts[forPartSup[i]]
    a. Instance(name=nameinstSup[i], part=p, dependent=ON)
    a. rotate(instanceList=('nameinstSup[i],
                            axisPoint=(0.0,0.0,0.0),
                            axisDirection=(1.0,0.0,0.0),
                            angle=90.0))
    a. rotate(instanceList=('nameinstSup[i],
                            axisPoint=(0.0,0.0,0.0),
                            axisDirection=(0.0,0.0,1.0),
                            angle=90.0))
    a. translate(instanceList=('nameinstSup[i],
                             vector=(coordX[i+1],(wD-forCooSup[i][2][0])/2,
                                     coordZSup[i]))

# Bearings assembly
p = mdb.models[model].parts[namepartBear]
for i in range(4):
    for j in range(2):
        a. Instance(name='nameinstBear'+str(i+1)+'str(j+1), part=p, dependent=ON)
        a. translate(instanceList=('nameinstBear'+str(i+1)+'str(j+1),
                                 vector=cooBear[i*2+j]))

# Defining set names to the connection edges of the deck
# Partitioning the deck according to pre-defined values
for model in modelNames:
    p = mdb.models[model].parts[namepartD]
e = p.edges
i=0
for name in namesetConnectionD:
    edges = e.findAt((cooFindAtConnectionD[i], ))
    p.Set(edges=edges, name=name)
    i += 1

for model in modelNames:
    p = mdb.models[model].parts[namepartD]
    f, e, d = p.faces, p.edges, p.datums
    t = p.MakeSketchTransform(sketchPlane=f.findAt(coordinates=(lD/2, wD/2, 0.0),
                       normal=(0.0, 0.0, 1.0)),
                       sketchUpEdge=e.findAt(coordinates=(0.0, wD/2, 0.0)),
                       sketchPlaneSide=SIDE1, sketchOrientation=LEFT,
                       origin=(0.0, 0.0, 0.0))
    s = mdb.models[model].ConstrainedSketch(name='__profile__',
                                             sheetSize=94.25,
                                             gridSpacing=2.35, transform=t)
    g, v, d1, c = s.geometry, s.vertices, s.dimensions, s.constraints
    s.setPrimaryObject(option=SUPERIMPOSE)
    p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
    for X in splitPosX:
        s.Line(point1=(X, 0), point2=(X, wD))
    for Y in splitPosY:
        s.Line(point1=(0, Y), point2=(lD, Y))
    f = p.faces
    pickedFaces = f.findAt(((lD/2, wD/2, 0.0), ))
    p.PartitionFaceBySketch(sketchUpEdge=e.findAt(coordinates=(0.0, wD/2, 0.0)),
                            faces=pickedFaces, sketchOrientation=LEFT,
                            sketch=s)
    s.unsetPrimaryObject()
    del mdb.models[model].sketches['__profile__']

# Creating sets of the new partitions (for the support surfaces)
i=0
for coord in cooFindAtConnectionSup:
    faces = f.findAt(((coord[0]+0.001, coord[1]+0.001, 0.0), ),
                     ((coord[0]+0.001, coord[1]-0.001, 0.0), ),
                     ((coord[0]-0.001, coord[1]-0.001, 0.0), ),
                     ((coord[0]-0.001, coord[1]+0.001, 0.0), ))
    p.Set(faces=faces, name=namesetConnectionSup[i])
    i += 1

# Partitioning the Shield wall to make a connection surface
# and giving them a set name

for model in modelNames:
    p = mdb.models[model].parts[namepartEND[0]]
    f, e, d = p.faces, p.edges, p.datums
    t = p.MakeSketchTransform(sketchPlane=f.findAt(coordinates=(wEndShield/2, hEndShield/2, 0.0), normal=(0.0, 0.0, 1.0)),
                       sketchUpEdge=e.findAt( coordinates=(0.0, hEndShield/4, 0.0),
                       sketchPlaneSide=SIDE1,
                       sketchOrientation=LEFT, origin=(0.0, 0.0, 0.0))
    s = mdb.models[model].ConstrainedSketch(name='__profile__',
                                             sheetSize=24.36,
                                             gridSpacing=0.6, transform=t)
    g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
    s.setPrimaryObject(option=SUPERIMPOSE)
APPENDIX C. PYTHON AND MATLAB SCRIPTS

```python
p.projectReferencesOntoSketch(sketch=s, filter=COPLANAR_EDGES)
s.rectangle(point1=cooConnectionEND[0], point2=cooConnectionEND[1])
s.rectangle(point1=cooConnectionEND[2], point2=cooConnectionEND[3])
s.rectangle(point1=cooConnectionEND[4], point2=cooConnectionEND[5])
f = p.faces
pickedFaces = f.findAt(((wEndShield/2, hEndShield/2, 0.0),))
e, d1 = p.edges, p.datums
p.PartitionFaceBySketch(sketchUpEdge=e.findAt(coordinates=(0.0, hEndShield/4, 0.0)),
faces=pickedFaces, sketchOrientation=LEFT, sketch=s)
s.unsetPrimaryObject()
del mdb.models[model].sketches['__profile__']
# Creating sets of the new partitions
f = p.faces
# Deck connection
tfaces = f.findAt((cooFindAtConnectionEND[0],),
(cooFindAtConnectionEND[1],),
(cooFindAtConnectionEND[2],))
p.Set(faces=tfaces, name=namesetConnectionEND[0])
# WingWall connection 1
tfaces = f.findAt((cooFindAtConnectionEND[0],),
(cooFindAtConnectionEND[3],))
p.Set(faces=tfaces, name=namesetConnectionEND[1])
# WingWall connection 2
tfaces = f.findAt((cooFindAtConnectionEND[2],),
(cooFindAtConnectionEND[4],))
p.Set(faces=tfaces, name=namesetConnectionEND[2])

# Defining set names for the wingwall connection to the endshield

for model in modelNames:
    for name in namesetConnectionWingWall:
        p = mdb.models[model].parts[namepartEND[1]]
e = p.edges
edges = e.findAt((cooFindAtConnectionWingWall,))
p.Set(edges=edges, name=name)

# Defining set names for support nodes

for model in modelNames:
    for i in range(len(namepartSup)):
        for j in range(2):
            p = mdb.models[model].parts[namepartSup[i]]
v = p.vertices
verts = v.findAt((cooFindAtNodeConnectionSup[i*2+j],))
p.Set(verts=verts, name=namesetNodeConnectionSup[i*2+j])

# Defining tie interactions between the bridge deck and the End Shield
# Defining coupling interactions between the bridge deck and the columns

for model in modelNames:
    XXXX
```
C.2. 3D MODEL SCRIPT

```python
i=0
for name in nameInteractionTie:
a = mdb.models[model].rootAssembly
region1=(a.instances[nameInstInteractionTie[i*2]].sets[nameSetInteractionTie[i*2]])
region2=(a.instances[nameInstInteractionTie[i*2+1]].sets[nameSetInteractionTie[i*2+1]])
 mdb.models[model].Tie(name=name, master=region1, slave=region2,
positionToleranceMethod=SPECIFIED,
positionTolerance=1, adjust=OFF, tieRotations=ON,
thickness=ON)
i+=1

i=0
for name in nameInteractionCoupling1:
a = mdb.models[model].rootAssembly
region1=(a.instances[nameInstInteractionCoupling1[i*2]].sets[nameSetInteractionCoupling1[i*2]])
region2=(a.instances[nameInstInteractionCoupling1[i*2+1]].sets[nameSetInteractionCoupling1[i*2+1]])
if i < 2 or i > 5:
    mdb.models[model].Coupling(name=name, controlPoint=region1,
surface=region2,
influenceRadius=WHOLE_SURFACE,
couplingType=KINEMATIC, localCsys=None,
u1=OFF, u2=ON, u3=ON, url=OFF, ur2=OFF, ur3=ON)
else:
    mdb.models[model].Coupling(name=name, controlPoint=region1,
surface=region2,
influenceRadius=WHOLE_SURFACE,
couplingType=KINEMATIC, localCsys=None,
u1=ON, u2=ON, u3=ON, url=OFF, ur2=OFF, ur3=ON)
i+=1

i=0
for name in nameInteractionCoupling2:
a = mdb.models[model].rootAssembly
region1=(a.instances[nameInstInteractionCoupling2[i*2]].sets[nameSetInteractionCoupling2[i*2]])
region2=(a.instances[nameInstInteractionCoupling2[i*2+1]].sets[nameSetInteractionCoupling2[i*2+1]])
 mdb.models[model].Coupling(name=name, controlPoint=region1,
surface=region2,
influenceRadius=WHOLE_SURFACE,
couplingType=KINEMATIC, localCsys=None,
u1=ON, u2=ON, u3=ON, url=ON, ur2=ON, ur3=ON)
i+=1
```

#Creating different calculation steps
## - Eigenfrequency step
## - Steady-state step
## - Dynamics step

```bash
i=0
```
for model in modelNames:
    mdb.models[model].FrequencyStep(name=steps[1], previous=steps[0],
    maxEigen=maxFreq, minEigen=minFreq)

mdb.models[model].SteadyStateDirectStep(name=steps[2], previous=steps[1],
    factorization=COMPLEX,
    frequencyRange=((lowLimit, upLimit, noSteps[0],
    1.0),),
    scale=LINEAR, frictionDamping=ON)

i+=1

# Creating boundary conditions

for model in modelNames:
    a = mdb.models[model].rootAssembly
    verts = []
    for i in range(len(nameinstSup)):
        for j in range(2):
            v = a.instances[nameinstSup[i]].vertices
            verts.append(v.findAt(((cooFindAtConnectionSup[i*2+j][0],
            cooFindAtConnectionSup[i*2+j][1],
            coordZSup[i]),)))
    region = a.Set(vertices=verts, name='suppFound ')
    mdb.models[model].DisplacementBC(name='BC - Supports ',
    createStepName=steps[0], region=region,
    u1=SET, u2=SET, u3=SET, ur1=SET, ur2=SET,
    ur3=SET, amplitude=UNSET,
    distributionType=UNIFORM, fieldName='',
    localCsys=None)

# -Apply steady-state load
# -Define amplitude
# -Apply amplitude to dynamics step

i=0
for model in modelNames:
    a = mdb.models[model].rootAssembly
    v = a.instances[nameinstD].vertices
    # verts = v.findAt(((loadPos[i]),))
    verts = v.findAt(((loadPos[0]),))
    region = a.Set(vertices=verts, name=namesetLoad)
    mdb.models[model].ConcentratedForce(name=nameLoad[0],
    createStepName=steps[2], region=region,
    cf3=-loadSize[0]+0j,
    distributionType=UNIFORM, field='',
    localCsys=None)

i+=1

# Meshing the model
C.2. 3D MODEL SCRIPT

```python
i=0
for model in modelNames:
    j=0
    for part in namesAllParts:
        if j==1:
            size = seedFactor_1[i]
        elif j==2:
            size = seedFactor_2[i]
        else:
            size = seedSize[i]
        p = mdb.models[model].parts[part]
        p.seedPart(size=size, deviationFactor=devFactor,
                   minSizeFactor=minSizeFactor)
        p.generateMesh()
        j+=1
    i+=1
```

---

```python
for model in modelNames:
    p = mdb.models[model].parts[namepartD]
    n = p.nodes
    p.Set(nodes=n, name='All_Nodes')
    region = p.sets['All_Nodes']
    i=0
    for Y in splitPosY:
        if Y == 0 or Y < wD/2+0.001 and Y > wD/2-0.001 or Y == wD:
            Xmin=0-0.001
            Xmax=lD+0.001
            Ymin=Y-0.001
            Ymax=Y+0.001
            Zmin=-0.001
            Zmax=0.001
            myNodes = n.getByBoundingBox(Xmin, Ymin, Zmin, Xmax, Ymax, Zmax)
        p.Set(nodes=myNodes, name=nameSetResult[i])
        i+=1
```

---

```python
for model in modelNames:
    regionDef = (mdb.models[model].rootAssembly.instances[nameinstD]
                  .sets[nameSetResult[0]])
    (mdb.models[model].fieldOutputRequests[nameFieldOutputRequest[1]].
     setValues(variables=('A', 'U'), region=regionDef, sectionPoints=DEFAULT, rebar=EXCLUDE))
    mdb.models[model].fieldOutputRequests.changeKey(fromName='F-Output-2',
                                                   toName=nameSetResult[0])
    for i in range(len(nameSetResult)-1):
        regionDef = (mdb.models[model].rootAssembly.instances[nameinstD]
                     .sets[nameSetResult[i+1]])
        mdb.models[model].FieldOutputRequest(name=nameSetResult[i+1],
                                               createStepName=steps[2],
                                               variables=('A', 'U'), region=regionDef,
```

XXV
C.3 BRIGADE Result File to Text Files Script

The Python script used for extracting the results from BRIGADE to structured tables in text files.

```python
from sys import path
path.append('C:/Users/rbg/Dropbox/Exjobb_R&O/3 - Modell/3D_Model/'
             'Model_script')
from Variables import *
from abaqusConstants import *
import odbAccess
import os

# Deletes all existing files in result folder
resultpath = ('C:/Users/rbg/Documents/EXJOBB/Exjobb_R&O/3 - Modell/3D_Model/
              'Outputs/DATA_files/')
fileList = [f for f in os.listdir(resultpath) if f.endswith('.data')]
for f in fileList:
    os.remove(resultpath+f)

# Counter variables
i = 0
j = 0

# Function that writes frequency spectrum from steady-state calculations
def writeFrequencySpectrum(step, node, nodeFile, resultFile, i, region, j):
    nodeFile.write('%4s	 %8s	 %8s	 %8s
' % ('node ', 'x', 'y', 'z '))
    nodeFile.write('%4i	 %8.2 F	 %8.2 F	 %8.2 F
' % (node.label,
            node.coordinates[0],
            node.coordinates[1],
            node.coordinates[2]))
    resultFile.write('%11 s	 %15 s	 %15 s	 %15 s	 %15 s	 %15 s	 %15 s
' % ('f',
             'aR_x',
             'aR_y',
             'aR_z',
             'aI_x',
             'aI_y',
             'aI_z'))
    for frame in step.frames:
        resultData = frame.fieldOutputs['A'].getSubset(region=region,
                                                       position=NODAL,
                                                       readOnly=ON)
        value = resultData.values[i]
        freq = frame.frequency
        frameNumber = j
        if frameNumber > 1:
```

XXVI
C.3. BRIGADE RESULT FILE TO TEXT FILES SCRIPT

```python
resultFile.write('%11.8F	%15.8E	%15.8E	%15.8E
' % (freq, value.data[0], value.data[1], value.data[2], value.conjugateData[0], value.conjugateData[1], value.conjugateData[2]))

elif frameNumber == 1:
    resultFile.write('%11.8F	%15.8E	%15.8E	%15.8E	%15.8E
' % (0, 0, 0, 0, 0, 0, 0))

j += 1
nodeFile.write('
')
resultFile.write('
')
j = 0

def writeEigenFrequencies(frame, eigenFile, modeFile, j):
    eigenFile.write('%11s
' % ('fn'))
    eigenFile.write('%11.8F
' % (frame.frequency))
    modeFile.write('%4s	%8s	%8s	%8s	%15s	%15s	%15s
' % ('node', 'x', 'y', 'z', 'u_x', 'u_y', 'u_z'))
    for sets in nameSetResult:
        # Calling all nodes for evaluation of mode shapes
        region = (odb.rootAssembly.instances[instanceName.upper()]
                   .nodeSets[sets.upper()])
        resultData = frame.fieldOutputs['U'].getSubset(region=region,
                                                       position=NODAL,
                                                       readOnly=ON)

        for value in resultData.values:
            modeFile.write('%4i	%8.2F	%8.2F	%8.2F	%15.8F	%15.8F
' % (node[j].label, node[j].coordinates[0], node[j].coordinates[1], node[j].coordinates[2],
                                                       value.data[0], value.data[1], value.data[2],))
            j += 1
        j = 0
    eigenFile.write('
')
    modeFile.write('
')
j = 0

# Model names in BRIGADE+
odbNames = modelNames

# Name of result file
resultFileNames = odbNames

# Amount of models (-1)
seq_id = 0

# Path of odb-file for model
odbpath = 'C:/BRIGADE Plus Work Directory/'

# Name of steps, instances and node set for steady-state step
stepName = steps[2]
instanceName = nameinstD

# Node number counter
nodeNo = 0

# for loop that fetches data from steady-state
for models in odbNames:
    # Access the files

XXVII
APPENDIX C. PYTHON AND MATLAB SCRIPTS

```python
odb = odbAccess.openOdb(path=odbpath+odbNames[seq_id]+'.odb')
step = odb.steps[stepName]
for sets in nameSetResult:
    # Nodes of interest
    region = (odb.rootAssembly.instances[instanceName.upper()]
    .nodeSets[sets.upper()])
    # Write the frequency spectrum file
    for node in region.nodes:
        nodeFile = open(resultpath+resultFileNames[seq_id]+ '_'+
        stepName+'_'+str(nodeNo+1)+'.info.data','w+ ')
        resultFile = open(resultpath+resultFileNames[seq_id]+ '_'+
        stepName+'_'+str(nodeNo+1)+'.data','w+ ')
        writeFrequencySpectrum(step, node, nodeFile, resultFile, i,
        region, j)
        i += 1
        nodeNo += 1
    i = 0
    nodeNo = 0
    seq_id += 1
    odb.close()
    nodeFile.close()
    resultFile.close()

# Name of steps for eigenfrequencies
stepName = steps[1]
instanceName = nameinstD
for models in odbNames:
    # Access the file
    odb = odbAccess.openOdb(path=odbpath+odbNames[seq_id]+'.odb')
    step = odb.steps[stepName]
    # Write the frequency spectrum file
    for frame in step.frames:
        if i > 0:
            eigenFile = open(resultpath+resultFileNames[seq_id]+ '_'+
            stepName+'_'+str(i)+'.info.data','w+ ')
            modeFile = open(resultpath+resultFileNames[seq_id]+ '_'+
            stepName+'_'+str(i)+'.data','w+ ')
            writeEigenFrequencies(frame, eigenFile, modeFile, j)
            i += 1
        i = 0
        seq_id += 1
        odb.close()
        eigenFile.close()
        modeFile.close()
    seq_id = 0

# Creates a file for MATLAB to identify if process is completed
programCompleted=open('C:/Users/rbg/Dropbox/Exjobb_R&O/3 - Modell/'
    '3D_Model/Result_script/pyDATA_Completed.data','w+ ')
programCompleted.close()
```

XXVIII
The MATLAB script used for importing the results from text files to a MATLAB structure.

%% Running Python scripts

prompt = 'Enter your preferred filename extension: ';
ext = input(prompt, 's');
command = 'main.bat';
system(command);
command = 'runPython.bat';
system(command);

i = 0;
while i == 0
    i = exist(strcat([C:/Users/rbg/Documents/EXJOBB/Exjobb_R&O/3 - Modell/3 D_Model/Result_script/pyDATA_Completed.data]));
    pause(pauseLen)
end

pathName = strcat([C:/Users/rbg/Documents/EXJOBB/Exjobb_R&O/3 - Modell/3 D_Model/Outputs/DATA_files/]);

%% Defining structures m (models), s (steps), e (eigenfrequencies), % and n (nodes)

% For steady state direct
dModels = dir(strcat([C:/Users/rbg/Documents/EXJOBB/Exjobb_R&O/3 - Modell/3 D_Model/Outputs/DATA_files/* Eigenfrequencies_1.data]));
numModels = length(dModels);
for i = 1:numModels
    fileName = dModels(i).name;
    m(i).name = strrep(fileName, '_Eigenfrequencies_1.data', '');
    m(i).date = dModels(i).date;
end
dSteps = dir(char(strcat(pathName, m(1).name, '_*_1.data')));
numSteps = length(dSteps);
for i = 1:numModels
    for j = 1:numSteps
        partOne = dSteps(j).name;
        partTwo = strrep(partOne, strcat(m(i).name, '_'), '');
        m(i).s(j).name = strrep(partTwo, '_1.data', '');
    end
end

%% Gathering data

for i = 1:numModels
% Gathering eigenfrequencies
dFiles = dir(char(strcat(pathName, m(i).name,...
'Eigenfrequencies*.data')));
numEigen = length(dFiles)/2;
for j = 1:numEigen
    eigenSource = char(strcat(pathName, m(i).name,...
'Eigenfrequencies_', num2str(j), 'info.data'));
    fid = fopen(eigenSource, 'r');
    infoTitle = strsplit(fgets(fid));
    % The 1 for infoValues is the number of columns,
    % change if more data requested
    infoValues = (fscanf(fid, '%f', [1 inf]))';
    for k = 1:numSteps
        tf = strcmp(m(i).s(k).name, 'Eigenfrequencies');
        if tf == 1
            m(i).s(k).data.e(j).char(infoTitle(2))) = infoValues(:,1);
        end
    end
    fclose(fid);
end

% Gathering information from steady state step
nodeSource = char(strcat(pathName, m(i).name,...
'Steady-state-direct*.data'));
numNodes = length(dFiles)/2;
for j = 1:numNodes
    nodeSource = char(strcat(pathName, m(i).name,...
'Steady-state-direct_', num2str(j), 'info.data'));
    fid = fopen(nodeSource, 'r');
    infoSource = strsplit(fgets(fid));
    % The 4 for inforValues is the number of columns,
    % change if more data requested
    infoValues = (fscanf(fid, '%d %f %f %f %f %f %f', [4 inf]))';
    for k = 1:numSteps
        tf = strcmp(m(i).s(k).name, 'Steady-state-direct');
        if tf == 1
            for x = 1:4
                m(i).s(k).data.n(j).char(infoTitle(x)))... = infoValues(:,x);
            end
        end
    end
    fclose(fid);
end

XXX
fclose(fid);
dataSource = char(strcat(pathName, m(i).name,...
'_Steady-state-direct_ ', num2str(j), '_data'));
fid = fopen(dataSource, 'r');
dataTitle = strsplit(fgets(fid));

% The 7 for dataValues is the number of columns,
% change if more data requested
dataValues = (fscanf(fid, '%f %f %f %f %f %f %f', [7 inf]))';
for k = 1:numSteps
    tf = strcmp(m(i).s(k).name, 'Steady-state-direct');
    if tf == 1
        for x = 1:7
            m(i).s(k).data.n(j).(char(dataTitle(x+1))) = dataValues(:,x);
        end
    end
end
fclose(fid);
savePath = char(strcat(['C:/Users/rbg/Documents/EXJOBB/Exjobb_R&O/3-Modell/3D_Model/Outputs/MAT_files/matOutPut_ ', ext, '.mat']));
save(savePath, 'm');
delete('abaqus.rpy', 'Session.log.1', 'pyDATA_Completed.data',...
'MatLab_ResultToFindle_3D.asv')
