Assessment of the dynamic response of the Bergsøysund Bridge in the frequency domain

Knut Andreas Kvåle

Master of Science in Mechanical Engineering
Submission date: June 2013
Supervisor: Ole Andre Øiseth, KT

Norwegian University of Science and Technology
Department of Structural Engineering

NTNU

Faculty of Engineering Science
and Technology
Department of Structural Engineering

MASTER THESIS 2013

Subject area:	**Date:**	**No. of pages:**
Structural Dynamics	10.06.13	139 (22+81+36)

Title:
Assessment of the dynamic response of the Bergsøysund Bridge
in the frequency domain

Dynamisk analyse av Bergsøysundbrua i frekvensplanet

By:
Knut Andreas Kvåle

Responsible teacher: Associate Professor Ole Øiseth
Supervisor(s): Associate Professor Ole Øiseth
Carried out at: Department of Structural Engineering

Institutt for konstruksjonsteknikk

FAKULTET FOR INGENIØRVITENSKAP OG TEKNOLOGI
NTNU – Norges teknisk-naturvitenskapelige universitet

MASTEROPPGAVE 2013

for

Knut Andreas Kvåle

Dynamisk analyse av Bergsøysundbrua i frekvensplanet

Assessment of the dynamic response of the Bergsøysund Bridge in the frequency domain

I forbindelse med prosjektet ferjefri E39 skal Bergsøysundbrua instrumenteres for å kartlegge nøyaktigheten til de metodene som benyttes til å beregne dynamisk respons av slike konstruksjoner utsatt for naturlaster. Denne oppgaven dreier seg om å utvikle et program i Matlab som beregner den dynamiske responsen av flytebruer i frekvensplanet.

Oppgaven bør inneholde følgende temaer:

- Beregning av hydrodynamisk bølgelastlast, masse, demping og stivhet i DNV programmet Wadam.
- Hente svingeformer og konstruksjonsegenskaper i fra Abaqus.
- Beregne systemets egenfrekvenser, svingeformer og dempingsforhold i Matlab.
- Beregne den dynamiske responsen til systemet i Matlab.

Besvarelsen organiseres i henhold til gjeldende retningslinjer.

Veileder(e): Ole Andre Øiseth

Besvarelsen skal leveres til Institutt for konstruksjonsteknikk innen 10. juni 2013.

NTNU, 14. januar, 2013

Ole Andre Øiseth
faglærer

Sammendrag

Statens vegvesen planlegger og bygger ny E39 langs den fjordkledde norske vestkysten. I den forbindelse er det viktig med verktøy som kan beregne dynamisk respons av flytende konstruksjoner utsatt for stokastisk bølgelast. Denne typen problemstillinger kan løses både i tidsplanet og frekvensplanet. For å etablere tidsplansløsninger må tidsavhengige laster genereres. Disse genereres gjennom Monte Carlo-simuleringer basert på den stokastiske bølgelasten. Deretter må lange, og mange, tidsserier beregnes for å til slutt få konfidens i resultatet. På grunn av dette, er det ofte vel så gunstig med frekvensplansløsninger av lineære, stokastiske problemer. Nøyaktigheten til veletablerte metoder for beregning av stokastisk respons i frekvensplanet ble undersøkt. Et MATLAB-program til beregning av flytende konstruksjoner utsatt for stokastisk bølgelast ble utviklet, hvor DNV HYDROD WADAM-analyser av de flytende elementene og en ABAQUS/CAE-modell av konstruksjonen antas gitt. Ved hjelp av tilstandsromsrepresentasjon av systemet, og iterasjon, ble også det komplekse og ikke-lineære egenverdiproblemet til systemet løst. Resultatene stemmer bra overens med resultater fra tilsvarende beregninger utført i tidsplanet. 20 svingemoder ble vurdert som tilstrekkelig på Bergsøysundbrua, med tanke på konvergens, når lasten var definert som hvit støy. Med estimert bølgelastspekter var det nødvendig med omtrent 30 svingemoder for konvergert løsning. Funnene støtter troverdigheten til metodene som er brukt, og indikerer tilfredsstillende nøyaktighet på de anvendte metodene.

Abstract

The Norwegian Public Roads Administration is currently both planning and building a new coastal highway, along the fjord-dense Norwegian west coast. In that concern, dynamic response of floating structures exposed to wave loads is of high importance. This kind of problem can be solved in the frequency or the time domain. In time domain analyses, the stochastic load must be estimated using Monte Carlo simulations, and long time series run to find confident results for the stochastic response. For linear problems, calculations in the frequency domain are therefore often favourable. The accuracy of the well-established linear frequency domain methods was studied, for stochastic problems, in this thesis. In particular, the Bersøysund Bridge was considered. A program was developed in MATLAB for calculations of structures on floating elements, where DNV HYDROD WADAM analyses of the floating elements and an ABAQUS/CAE model of the frame of the structure are prerequisites. Using state space form and iteration, the complex and non-linear modal eigenvalue problem of the system was also solved. The findings of this thesis involve that the results agree well with similar analyses performed in the time domain. 20 modes resulted in near-converged results for the model of the Bergsøysund Bridge exposed to white noise loading. When estimated sea load was enforced, approximately 30 included modes were needed for convergence. The findings supported the credibility of the methods used, and indicated adequate accuracy.

Preface

This report is the result of 20 weeks work between January and June 2013, and constitutes my master thesis at the Department of Structural Engineering at the Norwegian University of Science and Technology (NTNU). It finalizes my study on the programme Mechanical Engineering, Applied Mechanics. The master thesis has been carried out under supervision of Associate Professor Ole Øiseth.

Acknowledgements

I would like to give big thanks to my supervisor Associate Professor Ole Øiseth for discussions and guidance during the work with this master thesis. His assistance has been essential for the completion of this project - and his office door always open when input was needed. His prompt response on emails and general eagerness to assist, also long past working hours, has been highly appreciated. The present work has been very educational, and I am grateful for the opportunity.

Also, I would like to thank Sindre M. Hermstad for his work on the ABAQUS/CAE model of the Bergsøysund bridge, and analyses performed in DNV HYDROD WADAM regarding pontoons of the bridge. Both have been necessities for my work. Thank you for valuable discussions regarding the comparison of results.

Big thanks are also given to Professor Ragnar Sigbjörnsson, for his supply of the load spectra used in the calculations, and valuable discussions regarding this. Thank you in particular for using precious time of your weekend to help solve some of the obstacles faced along the way.

I would also like to thank the guys at the office, in particular Lars Edvard Dæhli and Erik Grimsmo, for fruitful discussions and advice.

Knut Andreas Kvåle

Trondheim, 10.06.13

Contents

Notation V

List of figures X

List of tables XI

1 Introduction 1
- 1.1 State of the art 3
- 1.2 Problem description 3
- 1.3 Scope of work 3
- 1.4 Structure of the report 4

2 Theory 5
- 2.1 Dynamics of SDOF systems 5
- 2.2 Dynamics of MDOF systems 7
  - 2.2.1 Frequency response method 8
  - 2.2.2 Modal transformation 9
  - 2.2.3 Rayleigh damping 10
- 2.3 Complex eigenvalues 11
  - 2.3.1 Example: two-storey shear frame 13
- 2.4 Fourier transform 17
  - 2.4.1 Discrete and fast Fourier transforms 17
  - 2.4.2 Leakage, windowing and estimation of spectra 18
- 2.5 Random data 20
  - 2.5.1 Characteristic processes 21
  - 2.5.2 Covariance 22
  - 2.5.3 Variance spectrum 23
  - 2.5.4 Spectra in multi-variable systems 27
- 2.6 Frequency domain dynamics for a floating bridge 28
  - 2.6.1 Hydrodynamic loading 30
  - 2.6.2 Wave action in three dimensions 30
  - 2.6.3 Example: submerged two-storey shear frame 32
  - 2.6.4 Modal representation of response variance spectra 34
  - 2.6.5 Modal representation of load variance spectra 36

2.6.6 Modal transformations for pontoon action . . . 36
2.7 Rigid rotation transformation matrix . . . 38
2.7.1 Transformation of load vector . . . 39
2.7.2 Transformation of added system matrices . . . 40

3 Matlab program 41
3.1 Solid structure . . . 41
3.2 Pontoons . . . 43
3.2.1 DNV HydroD Wadam . . . 43
3.2.2 Establishing the pontoon arrangement . . . 43
3.2.3 Transformation to modal coordinates . . . 44
3.2.4 Wet contributions included in Abaqus model . . . 44
3.3 Important operations . . . 45
3.3.1 Transformation of load spectra . . . 45
3.3.2 Modal response spectra . . . 48
3.3.3 Transformation to physical coordinates . . . 48
3.4 Eigenfrequencies and modal damping coefficients . . . 49
3.5 Versatility . . . 50

4 Case: Simply supported beam 51
4.1 Simply supported beam . . . 51
4.1.1 Results and discussion . . . 51
4.2 Simply supported beam with pontoon on midspan . . . 55
4.2.1 Results and discussion . . . 55

5 Case: The Bersøysund Bridge 59
5.1 Frame . . . 60
5.2 Pontoons . . . 61
5.3 Load cases . . . 63
5.3.1 White noise . . . 63
5.3.2 Estimated load spectra . . . 63
5.4 Results and discussion . . . 64
5.4.1 Eigenfrequencies, mode shapes and damping ratios . . . 64
5.4.2 Response from white noise . . . 66
5.4.3 Response from load spectra . . . 74
5.4.4 Duration of calculations . . . 77

6 Concluding remarks 79
6.1 Future work . . . 80

References 81

A Supplementary theory 83
A.1 Engineering results from spectral values . . . 83
A.1.1 Threshold crossings . . . 83
A.1.2 Extremal values . . . 84

A.2 Establishing transformation matrix for pontoon 85
A.3 Aerodynamics 88
A.3.1 Basic fluid mechanics 88
A.3.2 Linear potential functions 89
A.3.3 Panel method 91

B Matlab code 95
B.1 floatingBridge.m 95
B.2 structuralMatrices.m 99
B.3 hydroMatrices.m 99
B.4 dat2eig.m 101
B.5 Tmat.m 102
B.6 iterateFreq.m 103
B.7 eigSolve.m 104
B.8 plotModes.m 105
B.9 pontoonGeometry.m 106
B.10 shearframe.m 111
B.11 interp1z.m 112

C Comprehensive plots 115

Notation

DFT	Discrete Fourier transform
DOF(s)	Degree(s) of freedom
FEA	Finite Element Analysis
FEM	Finite Element Method
FFT	Fast Fourier transform
GUI	Graphical user interface
MDOF	Multiple degree of freedom
NPRA	Norwegian Public Roads Administration
SDOF	Single degree of freedom

Symbols

$[\ \ ]$	Matrix
$\{\ \ \}$	Column vector
$[\ \ ]^T$	Matrix transpose
$[\ \ ]^*$	Matrix transpose and complex conjugate
$[\ \ ]^{-1}$	Matrix inverse
$\cdot$	Time derivative, $\frac{d}{dt}$
$Re(\cdot)$	Real part of complex number
$Im(\cdot)$	Imaginary part of complex number
$\mathcal{F}$	Fourier transform
$\mathcal{F}^{-1}$	Inverse Fourier transform
$\frac{\partial}{\partial x}$	Partial derivative with respect to x
∇	Del-operator, $[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}]^T$
∇^2	Laplacian, $\nabla \cdot \nabla$

Physical quantities

t	Time
u, $\dot{u}$, $\ddot{u}$	SDOF displacement, velocity and acceleration
$\{\mathbf{u}\}$, $\{\dot{\mathbf{u}}\}$, $\{\ddot{\mathbf{u}}\}$	MDOF displacement, velocity and acceleration vectors
k	Single stiffness, for one degree of freedom systems
$[\mathbf{K}]$	Stiffness matrix
c	Single damping, for one degree of freedom systems
$[\mathbf{C}]$	Damping matrix
m	Single mass, for one degree of freedom systems
$[\mathbf{M}]$	Mass matrix
p	Single degree of freedom load
$\{\mathbf{p}\}$	Load vector
ω	Load frequency $[\frac{rad}{sec}]$
ω_n	Natural frequency or eigenfrequency of system or mode $[\frac{rad}{sec}]$
β	Load frequency ratio, $\frac{\omega}{\omega_n}$
λ	Eigenvalue, in solution of differential equations, $\mathbf{q} \cdot e^{\lambda t}$
$\{\mathbf{q}\}$	Eigenvector, in solution of differential equations, $\mathbf{q} \cdot e^{\lambda t}$
$\{\phi_i\}$	Modal transformation vector for mode i
$[\mathbf{\Phi}]$	Modal transformation matrix, $[\phi_1, \phi_2 ... \phi_n]$
$\{\mathbf{y}\}$	Modal coordinates, representing the scaling of the mode shapes
c_{cr}	Critical damping, $2\sqrt{km}$
ξ	Damping ratio, $\frac{c}{c_{cr}}$
$H(\omega)$	Scalar frequency response function, relates load and response
$[\mathbf{H}(\omega)]$	MDOF frequency response function
α	Constant defining the mass proportional damping
β	Constant defining the stiffness proportional damping

$f(x,y)$	Joint probability distribution between the variables x and y
$f(x)$	Probability distribution of the variable x
$E(x)$	Statistical expectancy value of variable x, see also μ_x
μ_x	Statistical expectancy of variable x, see also $E(x)$
$\sigma_{x,y}$	Statistical covariance between variables x and y
σ_x	Statistical standard deviation of variable x
$Varx = \sigma_x^2$	Statistical variance of variable x
C_x	Auto-covariance of variable x
$C_{x,y}$	Covariance between variables x and y
S_x	Auto-covariance spectra of variable x
$S_{x,y}$	Covariance spectra of variables x and y
$G_{x,y}$	Single-sided covariance spectra of variables x and y
$\rho_{x,y}$	The correlation between variables x and y
$\gamma_{x,y}$	The spectral coherence between variables x and y
$[\mathbf{C_x}]$	Covariance matrix for variable vector $\{\mathbf{x}\}$
$[\mathbf{S_x}]$	Covariance spectra matrix for variable vector $\{\mathbf{x}\}$
α, β, γ, σ and ω_p	Parameters and peak frequency for 1D Jonswap spectrum

List of Figures

1.1 The grown dimensions of Sognefjorden at Lavik-Oppedal. 1
1.2 Floating bridge crossing Sognefjorden. Statens vegvesen. 2

2.1 SDOF-system. 5
2.2 Rayleigh damping for $\alpha = 5 \cdot 10^{-2}s^{-1}$ and $\beta = 3 \cdot 10^{-2}s^{-2}$. 11
2.3 Eigenvector plot without damping. 14
2.4 Eigenvector plot with damping. 14
2.5 Two-storey shear frame. 14
2.6 Vector $\mathbf{q}_j e^{\omega_{d,j} it}$ for shear frame. 16
2.7 Sampled part of sine wave $f(t)$ expanded to a periodic function $f_p(t)$. . . 19
2.8 FFT of 3.75 periods of a sine wave with frequency $2Hz$. 20
2.9 Different realizations of the same process. 21
2.10 Variance spectrum with real and imaginary parts. 25
2.11 Uniformly distributed white noise with bounds $\pm A_0$. 26
2.12 Jonswap load spectrum. 32
2.13 Two-storey shear frame exposed to random wave load. 33
2.14 Variance spectra for shear frame. 35
2.15 Decomposition of vectors in different coordinate systems. 39

3.1 Structure of the Matlab program. 42
3.2 Supporting software for correct arrangement of pontoons. 44
3.3 Transformation of load spectra to global coordinates. 48
3.4 Iteration procedure for frequency dependent eigenvalue problem. 49

4.1 Simply supported beam. 52
4.2 Response spectra for simply supported beam. 53
4.3 Selected covariance spectra for simply supported beam. 54
4.4 Simply supported beam with pontoon. 55
4.5 Response spectra for simply supported beam with pontoon. 56
4.6 Selected covariance spectra for simply supported beam with pontoon. . . 57
4.7 Response variances of the simply supported beam with one pontoon. . . . 58

5.1 The Bersøysund Bridge. Photo: Cato Edvardsen. 59
5.2 Model used for calculations of structural properties in Abaqus/CAE. . . 60

5.3 Detail below bridge, at pontoon connection. 61
5.4 Principal drawing of bridge from above. 62
5.5 DOFs of pontoon. 62
5.6 Difference between coordinate systems for load spectra and Abaqus model. 64
5.7 Comparison between eigenfrequencies. 65
5.8 Comparison between modal damping ratios. 66
5.9 Mode shapes from complex eigenvalue problem seen from above. 67
5.10 Mode shapes from complex eigenvalue problem seen from the side. 68
5.11 Mode shapes from real eigenvalue problem seen from above and the side. . 69
5.12 Response spectra for the Bersøysund Bridge. 70
5.13 Selected covariance spectra for the Bergsøysund Bridge. 71
5.14 Variance for heave and yaw of the Bersøysund Bridge. 72
5.15 Comparison between time domain and frequency domain solutions. 73
5.16 Heave response of pontoon 3, time domain vs. frequency domain. 74
5.17 Spectra for the heave of the Bersøysund Bridge. 75
5.18 Selected covariance spectra for the Bergsøysund Bridge. 76
5.19 Selected covariance spectra for the Bergsøysund Bridge (II). 76
5.20 Response spectra DOFs 1-3 on pontoon no. 1. 78

A.1 Two sets of DOFs; $\{\mathbf{v}\}$ and $\{\bar{\mathbf{v}}\}$. 86
A.2 The most important flows with potential functions. 90
A.3 Panel method discretization. 92
A.4 Construction and physical requirements for panels. 92

C.1 Response spectra for the heave of all pontoons, with white noise. 116
C.2 Auto-covariance spectra of load used as input. 117

List of Tables

2.1 Additions to system matrices from pontoons. 29
2.2 Parameters used with Jonswap spectrum. 32
2.3 Eigenfrequencies for the three different cases studied. 34

4.1 Stiffness and mass data for the simply supported beam studied. 51
4.2 Structural parameters for selected modes of the simply supported beam. . 53
4.3 Geometry data file for pontoon on midspan of simply supported beam. . . 56

5.1 Geometry data file for all pontoons of the Bersøysund Bridge. 62
5.2 Natural frequencies and damping ratios for modes 1 to 10. 65
5.3 Deviance of variances between time and frequency domain. 73
5.4 Approximate durations of calculations. 78

A.1 Potential functions for elementary flows in 2D. 91

1 Introduction

The Norwegian Public Roads Administration (NPRA) is currently working on the plans for the new *Coastal Highway E39* along the Norwegian west-coast. This route stretches 1100 km between the major cities Kristiansand and Trondheim, and incorporates multiple crossings of deep fjords, which today are crossed by eight ferry connections [1]. According to [1], about 50% of the traditional Norwegian export is generated by industry along this route. By eliminating the ferry connections, up to 7 hours of travel time can be saved [1].

Sognefjorden is the deepest and widest of the fjords along this route, and is used as a pioneer project. The dimensions of this strait where the crossing is planned, are shown in Figure 1.1. If the Sognefjord is possible to cross, the same can be assumed for the remainder of the straits along this challenging route. One of the technological solutions most likely to succeed in this, is a floating bridge. A concept model of a floating bridge over the Sognefjord is shown in Figure 1.2.

In connection with NPRAs project, verification of the accuracy of the methods used to calculate dynamic response of floating structures exposed to environmental loads is sought.

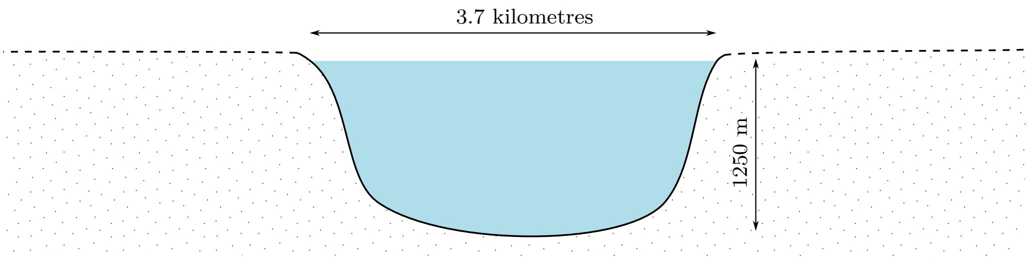


Figure 1.1: *The grown dimensions of Sognefjorden at Lavik-Oppedal.*

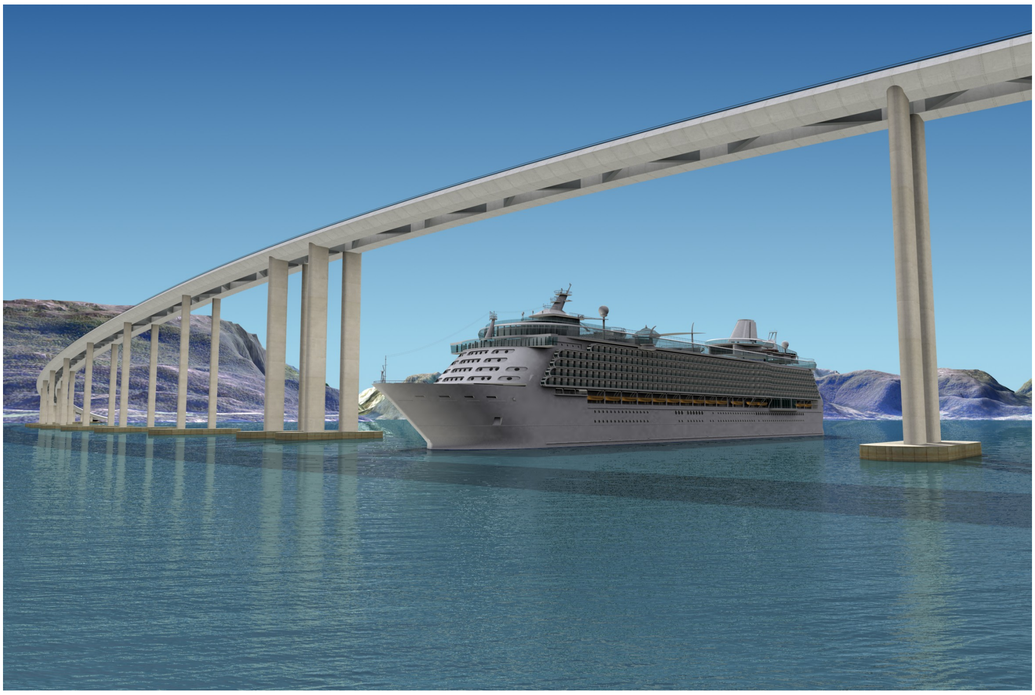


Figure 1.2: *Concept model of floating bridge for the crossing of the Sognefjord. Illustration: Norwegian Public Roads Administration.*

1.1 State of the art

For dynamic calculations in general, two main strategies are used; *time domain analysis* and *frequency domain analysis*. There exist vast amounts of literature concerning dynamics of structures exposed to stochastic frequency dependent loadings. Langen and Sigbjörnsson introduce many of the classical calculation methods in [2], both for frequency and time domain analyses.

Floating bridges would normally give rise to frequency dependent properties, such as mass and damping from the hydrodynamics between the water and the floating elements. Therefore, the frequency domain analysis tend to be more straightforward for this sort of problem. When non-linearities are introduced, frequency domain analyses complicate.

As computational power increase, time domain analyses become more practical. Shinozuka [3] reviews how Monte Carlo simulations can be used in many applications for time domain structural dynamics. Øiseth *et al.* describe a time domain procedure for prediction of wave induced dynamic response in [4], using Monte Carlo simulations. Preliminary results from the Bergsøysund Bridge are also presented in the mentioned paper.

1.2 Problem description

This master thesis concerns the development of a MATLAB program that calculates the linear dynamic response of floating bridges in the frequency domain.

In particular, the Bergsøysund Bridge is studied. The Bergsøysund Bridge is a 931 meters long floating bridge between the islands Aspøya and Bergsøya in the Norwegian county Møre og Romsdal. Seven lightweight concrete pontoons keep the bridge afloat, with a maximum span of 106 meters between them [4].

1.3 Scope of work

This master thesis aims at using MATLAB to calculate the dynamic response of the Bergsøysund Bridge. This should be done by using an already established ABAQUS/CAE model of the frame of the bridge, established stiffness, damping and mass contributions from pontoons based on DNV HYDROD WADAM analyses, and established estimated load spectra. The ABAQUS/CAE model should be used to retrieve modal properties, such as mass, mode shapes and eigenvalues. This will form the basis for stiffness, and by assuming Rayleigh damping, also the damping of the frame. The analyses in DNV HYDROD WADAM would further form the basis of the hydrodynamic contributions to the system, which should be summed and transformed to the modal space, using the mode shapes from the frame model results. Including additional stiffness, mass and damping, the eigenvalues and eigenvectors should also be recalculated.

The results found will be discussed regarding modal convergence, and compared with time series analyses. From this, some conclusions about the accuracy of the method are made.

1.4 Structure of the report

The report is structured in the way shown below.

Chapter 2 Basic theory needed for the problem at hand is presented. Some of the theory is not directly utilized, but serves as important building blocks for the subject.

Chapter 3 Presentation of the techniques used and assumptions made for the development of a program able to perform dynamic calculations on floating structures. Important aspects concerning the program are derived or explained.

Chapter 4 The program is used for two simplified cases; (a) a simply supported beam without any hydrodynamics, *i.e.* no pontoons, and (b) a simply supported beam with one pontoon at midspan. The simplified cases are loaded with *white noise*, and the resulting responses studied, among others with regard to number of modes included in the calculations.

Chapter 5 The Bergsøysund Bridge is studied using the same program. Important aspects regarding the modelling and the set-up are presented. The response is studied, both with regard to number of modes included, and compared with time series analysis performed by Sindre M. Hermstad. Both white noise load spectra and estimated load spectra established by Ragnar Sigbjörnsson are used.

Chapter 6 Main conclusions and some remarks about possible future work.

2 Theory

The theory presented constitute the basis for frequency domain analyses of floating structures. Some of the theory presented is not directly used, but still serves as important support and background theory.

2.1 Dynamics of SDOF systems

Assume a single degree of freedom (SDOF) system as shown in Figure 2.1. This gives, from dynamic equilibrium

$$m\ddot{u} + c\dot{u} + ku = p(t) \tag{2.1}$$

where u is the displacement, $p(t)$ is the applied load and m, c and k the mass, damping and stiffness of the system, respectively. $p(t)$ is assumed to be a simple harmonic load, and is expressed using the function $p(t) = p_0 cos(\omega t) = Re(p_0 e^{i\omega t})$, where p_0 is the amplitude, ω the load frequency and t the time.

This has the following solution form [5]:

$$u = e^{\lambda t} \tag{2.2}$$

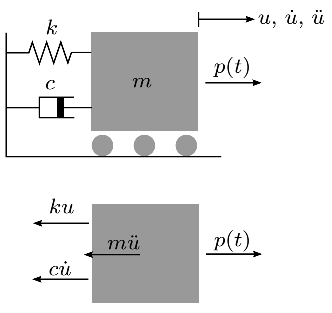


Figure 2.1: *SDOF-system.*

where λ is the eigenvalue for the problem. This can be found by arranging the characteristic equation;

$$\lambda^2 \cdot m + \lambda \cdot c + k = 0 \tag{2.3}$$

which by solving of roots is found to be

$$\lambda = -\frac{c}{2m} \pm \frac{\sqrt{(\frac{c}{m})^2 - 4 \cdot \frac{k}{m}}}{2} \tag{2.4}$$

If the second term is considered, $\pm\frac{\sqrt{\frac{c}{m}^2 - 4 \cdot \frac{k}{m}}}{2}$, the following can be concluded:

- Real for $\frac{c}{m}^2 - 4 \cdot \frac{k}{m} > 0$
- Zero for $\frac{c}{m}^2 - 4 \cdot \frac{k}{m} = 0$
- Imaginary for $\frac{c}{m}^2 - 4 \cdot \frac{k}{m} < 0$

These three cases define the three gradings of the damping; over-critical, critical and under-critical, respectively. The critical damping c_{cr} is therefore

$$c_{cr} = 2 \cdot \sqrt{km} \tag{2.5}$$

It is convenient to introduce these three basic definitions:

Natural frequency is defined as

$$\omega_n = \sqrt{\frac{k}{m}} \tag{2.6}$$

Damping ratio is defined as

$$\xi = \frac{c}{c_{cr}} = \frac{c}{2 \cdot \sqrt{km}} \tag{2.7}$$

Load frequency ratio is defined as

$$\beta = \frac{\omega}{\omega_n} \tag{2.8}$$

■

Using the definitions above, Equations 2.4 and 2.1 can be rewritten (assuming under-critical damping)

$$\lambda = -\xi\omega_n \pm \sqrt{1 - \xi^2}\omega_n i \tag{2.9}$$

and

$$\ddot{u} + (2\omega_n \xi)\dot{u} + \omega_n^2 u = \frac{p_0}{M} Re(e^{i\omega t}) \tag{2.10}$$

The solution to the differential equation shown in Equation 2.10 contains a homogeneous and a particular solution, $u = u_h + u_p$. The homogeneous solution has the form

$$u_h = Re[G_h \cdot e^{-\omega_n \xi t} \cdot e^{i\omega t\sqrt{1-\xi^2}}] \tag{2.11}$$

given that $\xi < 1$, *i.e.* under-critical damping. Here, G_h is a complex constant, not known at this point.

To obtain a particular solution, it is assumed that it resembles the function for the load, commonly referred to as *undetermined coefficients' method* [5]. This gives $u_p = Re[G_p e^{i\omega t}]$, where G_p is a complex unknown constant.

2.2 Dynamics of MDOF systems

Assume a multiple degree of freedom (MDOF) system

$$[\mathbf{M}]\{\ddot{\mathbf{u}}\} + [\mathbf{C}]\{\dot{\mathbf{u}}\} + [\mathbf{K}]\{\mathbf{u}\} = \{\mathbf{p}(t)\} \tag{2.12}$$

with N degrees of freedom (DOFs). This is assumed to have a homogeneous solution, *i.e.* $\{\mathbf{p}\} = \{\mathbf{0}\}$, on the form

$$\{\mathbf{u}\} = \{\mathbf{q}\}e^{\lambda t} \tag{2.13}$$

This gives

$$\{\dot{\mathbf{u}}\} = \lambda \cdot \{\mathbf{q}\}e^{\lambda t} \quad = \quad \lambda\{\mathbf{u}\} \tag{2.14}$$

$$\{\ddot{\mathbf{u}}\} = \lambda^2 \cdot \{\mathbf{q}\}e^{\lambda t} \quad = \quad \lambda^2\{\mathbf{u}\} \tag{2.15}$$

From Equations 2.14 and 2.15, and by assuming that the system is unloaded, Equation 2.12 can be rewritten

$$\left(\lambda^2 [\mathbf{M}] + \lambda [\mathbf{C}] + [\mathbf{K}]\right) \mathbf{q} = \{\mathbf{0}\} \tag{2.16}$$

For the special case of no damping, this gives

$$\left(\lambda^2 [\mathbf{M}] + [\mathbf{K}]\right) \{\mathbf{q}\} = \{\mathbf{0}\} \tag{2.17}$$

which results in

$$\lambda = \pm i\omega_n \tag{2.18}$$

For a one DOF system, this is seen easily;

$$[\lambda^2 M + K]q = 0 \Rightarrow \lambda^2 M + K = 0 \tag{2.19}$$

which leads to

$$\lambda^2 = -\frac{K}{M} = -\omega_n \Rightarrow \lambda = \pm i\omega_n \tag{2.20}$$

Using this, Equation 2.17 gives

$$det\left(-\omega_n^2\,[\mathbf{M}] + [\mathbf{K}]\right) = 0 \tag{2.21}$$

which is the classic eigenvalue problem of a dynamic system.

2.2.1 Frequency response method

The equation of motion, shown in Equation 2.12, can be used to find the response for a given applied harmonic load $\{\mathbf{p}_k(t)\} = \{\mathbf{p}_{0,k}\}e^{i\omega_k t}$;

$$[\mathbf{M}]\{\ddot{\mathbf{u}}_k\} + [\mathbf{C}]\{\dot{\mathbf{u}}_k\} + [\mathbf{K}]\{\mathbf{u}_k\} = \{\mathbf{p}_k(t)\} \tag{2.22}$$

The response from this single load is assumed

$$\{\mathbf{u}_k\} = \{\mathbf{G}_k\}e^{i\omega_k t} \tag{2.23}$$

which is the basis of the previously mentioned undetermined coefficients' method. Combining this with Equation 2.22, the following is obtained:

$$\{\mathbf{G}_k\}e^{i\omega_k t}\left(-\omega_k^2\,[\mathbf{M}] + i\omega_k\,[\mathbf{C}] + [\mathbf{K}]\right) = \{\mathbf{p}_k(t)\} \tag{2.24}$$

This can be written

$$\{\mathbf{u}_k(t)\} = [\mathbf{H}(\omega_k)]\{\mathbf{p}_k(t)\} \tag{2.25}$$

by introducing the *frequency response function*:

$$[\mathbf{H}(\omega)] = \left(-\omega^2\,[\mathbf{M}] + i\omega\,[\mathbf{C}] + [\mathbf{K}]\right)^{-1} \tag{2.26}$$

To expand to arbitrary loading, the total load is assumed a superposition of many loads:

$$\{\mathbf{p}(t)\} = \sum_{k=1}^{N_{tot}} \{\mathbf{p}_k(t)\} = \int_{-\infty}^{\infty} \{\mathbf{P}(\omega)\} e^{i\omega t} d\omega \tag{2.27}$$

where $\{\mathbf{P}(\omega)\}$ is the amplitude of the load at frequency ω. Using the same representation of the response vector, the following is found

$$\{\mathbf{p}(t)\} = \int_{-\infty}^{\infty} \{\mathbf{P}(\omega)\} e^{i\omega t} d\omega \tag{2.28}$$

$$\{\mathbf{u}(t)\} = \int_{-\infty}^{\infty} \{\mathbf{U}(\omega)\} e^{i\omega t} d\omega \tag{2.29}$$

By introducing Equation 2.25, Equation 2.29 gives

$$\int_{-\infty}^{\infty} \{\mathbf{U}(\omega)\} e^{i\omega t} d\omega = \int_{-\infty}^{\infty} [\mathbf{H}(\omega)] \{\mathbf{P}(\omega)\} e^{i\omega t} d\omega \tag{2.30}$$

yielding

$$\{\mathbf{U}(\omega)\} = [\mathbf{H}(\omega)] \{\mathbf{P}(\omega)\} \tag{2.31}$$

This represents the frequency dependent transformation from load to stationary response of an MDOF system.

2.2.2 Modal transformation

From Equation 2.17 the eigenvalues and eigenvectors of the system are found. This results in the total homogeneous solution

$$\{\mathbf{u}\} = a_1\{\phi\}_1 \cdot e^{\lambda_1 t} + a_2\{\phi\}_2 \cdot e^{\lambda_2 t} + ... + a_N\{\phi\}_N \cdot e^{\lambda_N t} \tag{2.32}$$

where $\{\mathbf{q}_j\} = a_j\{\phi\}_j$ represents eigenvector j. Further, this can be rewritten

$$\{\mathbf{u}\} = \{\phi\}_1 \cdot y_1 + \{\phi\}_2 \cdot y_2 + ... + \{\phi\}_N \cdot y_N \tag{2.33}$$

This can be written in matrix notation;

$$\{\mathbf{u}\} = \begin{bmatrix}\{\phi\}_1 & \{\phi\}_2 & ... & \{\phi\}_N\end{bmatrix}\begin{Bmatrix}y_1\\y_2\\...\\y_N\end{Bmatrix} \tag{2.34}$$

$$\{\mathbf{u}\} = [\mathbf{\Phi}]\{\mathbf{y}\} \tag{2.35}$$

It is assumed that the particular solution can be decomposed using the same shapes. Using this transform for the equation of motion, and pre-multiplying by $[\mathbf{\Phi}]^T$, the following is obtained:

$$[\mathbf{\Phi}]^T[\mathbf{M}][\mathbf{\Phi}]\{\ddot{\mathbf{y}}\} + [\mathbf{\Phi}]^T[\mathbf{C}][\mathbf{\Phi}]\{\dot{\mathbf{y}}\} + [\mathbf{\Phi}]^T[\mathbf{K}][\mathbf{\Phi}]\{\mathbf{y}\} = [\mathbf{\Phi}]^T\{\mathbf{p}(t)\} \tag{2.36}$$

Here, these modal sizes can be introduced:

$$[\tilde{\mathbf{M}}] = [\mathbf{\Phi}]^T[\mathbf{M}][\mathbf{\Phi}] \tag{2.37}$$

$$[\tilde{\mathbf{C}}] = [\mathbf{\Phi}]^T[\mathbf{C}][\mathbf{\Phi}] \tag{2.38}$$

$$[\tilde{\mathbf{K}}] = [\mathbf{\Phi}]^T[\mathbf{K}][\mathbf{\Phi}] \tag{2.39}$$

$$\{\tilde{\mathbf{p}}\} = [\mathbf{\Phi}]^T\{\mathbf{p}\} \tag{2.40}$$

2.2.3 Rayleigh damping

Rayleigh damping is a classical damping type, which for symmetrical mass and stiffness matrices gives a diagonal modal damping matrix. Mass and stiffness proportional damping are, in Chopra [6], defined

$$\mathbf{C_M} = \alpha\mathbf{M} \tag{2.41}$$

$$\mathbf{C_K} = \beta\mathbf{K} \tag{2.42}$$

Because the symmetric mass and stiffness matrices give orthogonal modal matrices, the two resulting damping matrices also must be orthogonal. Combining the two, the Rayleigh damping relation is established:

$$\mathbf{C} = \mathbf{C_M} + \mathbf{C_K} = \alpha\mathbf{M} + \beta\mathbf{K} \tag{2.43}$$

This yields this damping coefficient for mode n

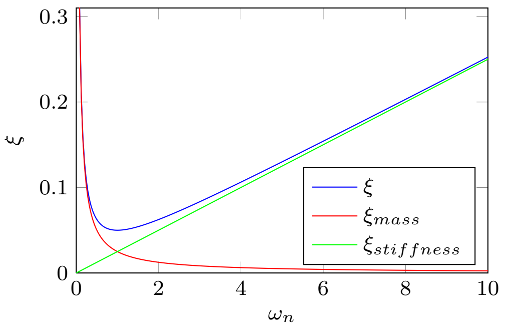


Figure 2.2: *Rayleigh damping for* $\alpha = 5 \cdot 10^{-2} s^{-1}$ *and* $\beta = 3 \cdot 10^{-2} s^{-2}$.

$$\xi_n = \alpha \cdot \frac{1}{2\omega_n} + \beta \cdot \frac{\omega_n}{2} \tag{2.44}$$

The damping ratio for mode n is plotted in Figure 2.2 using Equation 2.44.

2.3 Complex eigenvalues

The following is adapted from [7]. For the interested reader, more information about the subject is found in [8, 9, 10].

If damping is not excluded as in Section 2.2, the problem complicates. For the convenience of the reader, the eigenvalue problem given in Equation 2.16 is restated:

$$\left(\lambda^2 [\mathbf{M}] + \lambda [\mathbf{C}] + [\mathbf{K}]\right) \mathbf{q} = \{\mathbf{0}\}$$

By assuming the same form of the eigenvalue as for an SDOF system, Equation 2.9 gives

$$\lambda = -\xi\omega_n \pm \sqrt{1-\xi^2}\omega_n i \tag{2.45}$$

This gives the following relations:

$$\omega_n = |\lambda| \tag{2.46}$$

$$\xi = -\frac{Re(\lambda)}{|\lambda|} \tag{2.47}$$

Now remains the main problem; solving the eigenvalue problem for a second order differential equation. This can be done by rewriting the equation system in *state space* form [7]. First a new vector variable is introduced;

$$\{\mathbf{z}\} = \begin{Bmatrix} \mathbf{u} \\ \dot{\mathbf{u}} \end{Bmatrix} \tag{2.48}$$

Rewriting Equation 2.12 gives

$$\{\ddot{\mathbf{u}}\} + [\mathbf{M}]^{-1}[\mathbf{C}]\{\dot{\mathbf{u}}\} + [\mathbf{M}]^{-1}[\mathbf{K}]\{\mathbf{u}\} = [\mathbf{M}]^{-1}\{\mathbf{p}\} \tag{2.49}$$

From this a new equation of motion with variable $\{\mathbf{z}\}$ can be constructed;

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}} \\ \ddot{\mathbf{u}} \end{Bmatrix} + \begin{bmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{M}^{-1}\mathbf{K} & \mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \dot{\mathbf{u}} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{M^{-1}p} \end{Bmatrix} \tag{2.50}$$

which can be rewritten

$$\begin{Bmatrix} \dot{\mathbf{u}} \\ \ddot{\mathbf{u}} \end{Bmatrix} + \begin{bmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{M}^{-1}\mathbf{K} & \mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \dot{\mathbf{u}} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{M^{-1}p} \end{Bmatrix} \tag{2.51}$$

Using Equation 2.48, this can be rewritten on compact form

$$\{\dot{\mathbf{z}}\} + [\mathbf{A}]\{\mathbf{z}\} = \{\mathbf{Q}\} \tag{2.52}$$

where these definitions are introduced:

$$\{\mathbf{Q}\} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{M^{-1}p} \end{Bmatrix} \tag{2.53}$$

$$[\mathbf{A}] = \begin{bmatrix} \mathbf{0} & -\mathbf{I} \\ \mathbf{M}^{-1}\mathbf{K} & \mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \tag{2.54}$$

λ_j from Equation 2.45 is a complex conjugate pair, and each $\mathbf{q}_j$ seen in Equation 2.16 represents a complex-conjugated vector pair. Both conjugates have to be included in contribution of mode j to the overall solution:

$$\{\mathbf{u}_j\} = \{\bar{\mathbf{q}}_j\}e^{-\xi\omega_n - \sqrt{1-\xi^2}\omega_n it} + \{\mathbf{q}_j\}e^{-\xi\omega_n + \sqrt{1-\xi^2}\omega_n it} \tag{2.55}$$

By altering the exponential term, it is seen that

$$\{\mathbf{q}_j\}e^{-\xi\omega_n + \sqrt{1-\xi^2}\omega_n it} = \{\mathbf{q}_j\} \underbrace{e^{-\xi\omega_n}}_{\text{Damping}} \underbrace{e^{\sqrt{1-\xi^2}\omega_n it}}_{\text{Oscillation}} \tag{2.56}$$

When enforcing a complex eigenvalue problem, the different modes' contribution to the total solution experiences *spreading.* This can be seen from Figures 2.3 and 2.4. These figures show the vector $\{\mathbf{q}_j\}e^{\sqrt{1-\xi^2}\omega_n it}$ for a chosen mode, for real and complex eigenvalue solutions, respectively. In the figures, only the oscillation part of the exponential term is considered, such that the lengths of the vectors remain the same. The spreading effect is caused by the different complex values of the elements in $\{\mathbf{q}_j\}$, which gives rise to phase shifts. The exponential term only adds oscillation and damping, as shown in Equation 2.56.

The total solution, for all N pairs of complex conjugate modes, reads out

$$\{\mathbf{u}\} = \sum_{j=1}^{N} \left(\{\mathbf{q}_j\}e^{\lambda_j t} + \{\bar{\mathbf{q}}_j\}e^{\bar{\lambda}_j t} \right) \tag{2.57}$$

This can also be expressed on matrix form;

$$\{\mathbf{u}\} = \begin{bmatrix} \{\mathbf{q}_1\} & \{\bar{\mathbf{q}}_1\} & \{\mathbf{q}_2\} & \{\bar{\mathbf{q}}_2\} & ... & \{\mathbf{q}_N\} & \{\bar{\mathbf{q}}_N\} \end{bmatrix} \begin{Bmatrix} e^{\lambda_1 t} \\ e^{\bar{\lambda}_1 t} \\ e^{\lambda_2 t} \\ e^{\bar{\lambda}_2 t} \\ ... \\ e^{\lambda_N t} \\ e^{\bar{\lambda}_N t} \end{Bmatrix} \tag{2.58}$$

which on compact form can be written

$$\{\mathbf{u}\} = [\mathbf{\Psi}]\{\mathbf{g}\} \tag{2.59}$$

2.3.1 Example: two-storey shear frame

The state space form method described in the previous section was used together with the `eig` command in MATLAB to solve the damped eigenvalue problem of the shear frame shown in Figure 2.5.

From the standard beam stiffness matrix, each columns' contribution to the stiffness in the assigned DOFs becomes $12\frac{EI}{L^3}$. For simplicity this is denoted k. In addition, the mass of the columns are assumed negligible compared to the mass of the horizontal infinitely stiff beams, leading to a lumped appearance of the mass matrix. Using this, these global matrices are established

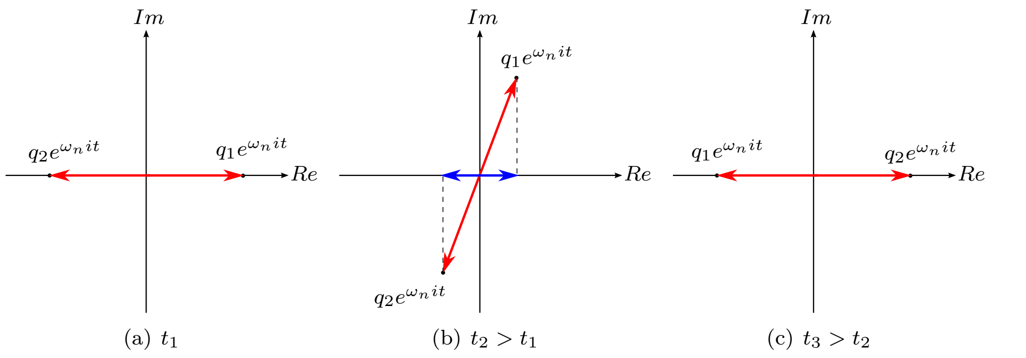


Figure 2.3: *Vector $\{\mathbf{q}\}e^{\omega_n it}$ schematically drawn at three different time instances, for $\omega_n = Im(\lambda)$, when damping is not included in calculation of eigenvectors and eigenvalues.*

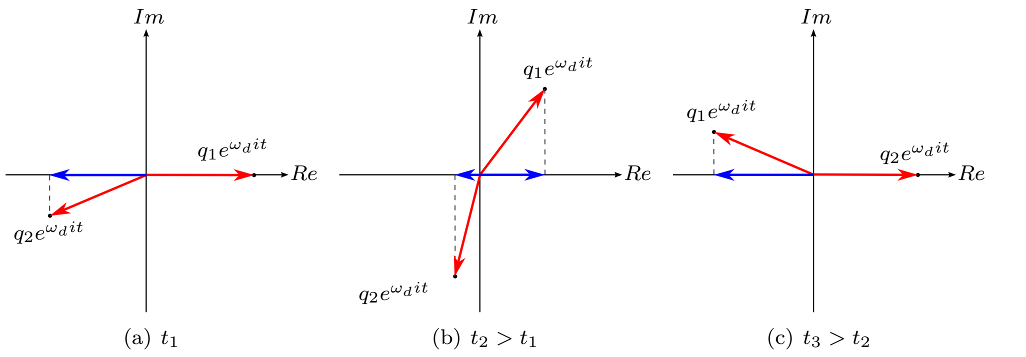


Figure 2.4: *Vector $\{\mathbf{q}\}e^{\omega_d it}$ schematically drawn at three different time instances, for $\omega_d = Im(\lambda) = \sqrt{1-\xi^2}\omega_n$, when damping is included in calculation of eigenvectors and eigenvalues. The damping introduces spreading between $q_1 e^{\omega_d it}$ and $q_2 e^{\omega_d it}$. This means that the maximum contributions for different DOFs in one mode do not occur at the same time.*

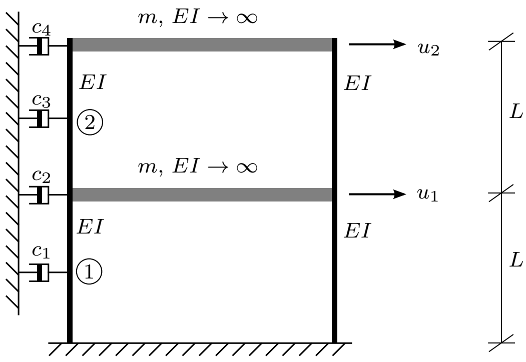


Figure 2.5: *Two-storey shear frame.*

$$[\mathbf{K}] = \begin{bmatrix} 4k & -2k \\ -2k & 2k \end{bmatrix} \tag{2.60}$$

$$[\mathbf{M}] = \begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix} \tag{2.61}$$

By placing dashpots as shown on Figure 2.5, a damping matrix can be established. The dashpots' given damping constants can be interpolated using the consistent interpolation formulas for beams. The following expression can be used to establish the damping matrix contribution from element i:

$$[\mathbf{C}]_{\textcircled{i}} = \sum_{j=1}^{n} \mathbf{N}(z_j)^T c_j \mathbf{N}(z_j) \tag{2.62}$$

where $\mathbf{N}$ is the interpolation vector, interpolating each element between its to end DOFs, evaluated at the position of dashpot j, z_j, and n is the total number of dashpots on element i.

Using this, the total damping matrix can be established as

$$[\mathbf{C}] = [\mathbf{C}]_{\textcircled{1}} + [\mathbf{C}]_{\textcircled{2}} = \begin{bmatrix} \frac{1}{4}c_1 + c_2 + \frac{1}{4}c_3 & \frac{1}{4}c_3 \\ \frac{1}{4}c_3 & \frac{1}{4}c_3 + c_4 \end{bmatrix} \tag{2.63}$$

For the special case of equal damping constant c for all dashpots, this gives

$$[\mathbf{C}] = \begin{bmatrix} \frac{6}{4}c & \frac{1}{4}c \\ \frac{1}{4}c & \frac{5}{4}c \end{bmatrix} \tag{2.64}$$

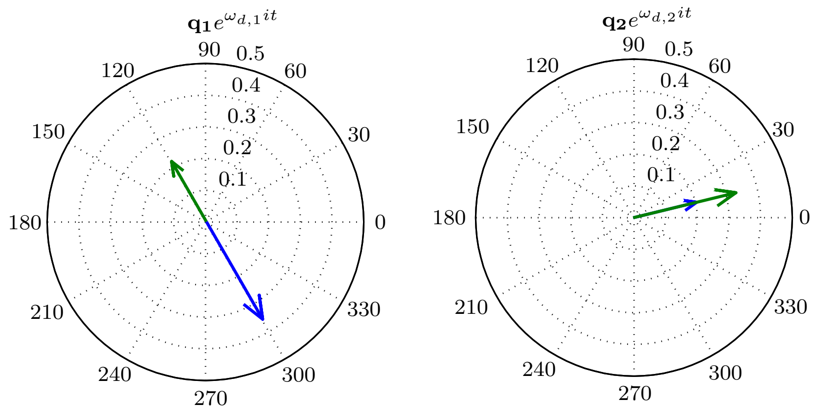
When introducing $c_1 = c_3 = 0$ and $c_2 = c_4 = c$, this further gives the simple diagonal matrix

$$[\mathbf{C}] = \begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix} \tag{2.65}$$

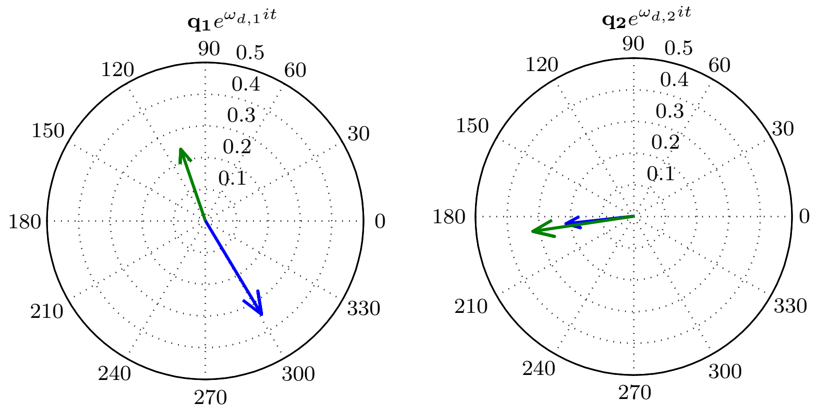
These two special cases of damping were studied using the MATLAB script `eigMain.m`. The script uses the function `eigSolve.m` to solve the eigenvalue problem, and the function `eigPlot.m` to plot the resulting solution. All these MATLAB files are found in Appendix B. The input used when running the calculations were

```
m = 1, k = 1, c = 1
```

The resulting complex vector plots are shown in Figure 2.6. For the case of diagonal damping, no spreading is observed, as shown in Figure 2.6(a). For the case of a symmetric non-diagonal damping matrix on the other hand, the complex eigenvalue results in spreading of the solution, as seen in Figure 2.6(b).



(a) $c_1 = c_3 = 0$ leads to a diagonal damping matrix, and no spreading.



(b) Equal damping coefficient c for all dashpots leads to a symmetric non-diagonal damping matrix, and spreading of solution.

Figure 2.6: *Vector* $\mathbf{q}_j e^{\omega_{d,j}it}$ *plotted at arbitrary time instance of both modes, for the shear frame shown in Figure 2.5.*

2.4 Fourier transform

The analytic *Fourier transform* of a function $f(t)$ can be defined [5]

$$\mathcal{F}(f(t)) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \tag{2.66}$$

where

$$f(t) = \int_{-\infty}^{\infty} \mathcal{F}(f(t))e^{i\omega t}d\omega \tag{2.67}$$

The factor $\frac{1}{2\pi}$ can appear in different forms, but what really matters, is the relation between the factors used with the Fourier transform and the inverse Fourier transform.

2.4.1 Discrete and fast Fourier transforms

When dealing with discrete signals, the analytical term defined above can be replaced with a *discrete Fourier transform* (DFT). According to Kreyszig [5], the vector with N discrete function values

$$\{\mathbf{f}\} = \begin{Bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{Bmatrix} \tag{2.68}$$

with a corresponding time vector

$$\{\mathbf{t}\} = \begin{Bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{N-1} \end{Bmatrix} \tag{2.69}$$

have the following DFT for component n:

$$\hat{f}_n = \sum_{k=0}^{N-1} f_k e^{-int_k} \tag{2.70}$$

This can be seen as the frequency spectrum of the signal. Rewritten in vector notation, this reads out

$$\{\hat{\mathbf{f}}\} = [\mathbf{F}]\{\mathbf{f}\} \tag{2.71}$$

where

$$[\mathbf{F}] = \begin{bmatrix} e^{-i\cdot 0\cdot t_0} & e^{-i\cdot 0\cdot t_1} & \cdots & e^{-i\cdot 0\cdot t_{N-1}} \\ e^{-i\cdot 1\cdot t_0} & e^{-i\cdot 1\cdot t_1} & \cdots & e^{-i\cdot 1\cdot t_{N-1}} \\ e^{-i\cdot 2\cdot t_0} & e^{-i\cdot 2\cdot t_1} & \cdots & e^{-i\cdot 2\cdot t_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-i\cdot (N-1)\cdot t_0} & e^{-i\cdot (N-1)\cdot t_1} & \cdots & e^{-i\cdot (N-1)\cdot t_{N-1}} \end{bmatrix} \tag{2.72}$$

Kreyszig [5] states that Equation 2.71 requires $\mathcal{O}(N^2)$ operations for all $n < \frac{N}{2}$. In most cases, many points have to be sampled in the time series for good results, thus leading to many operations. In an attempt to remedy this, the *fast Fourier Transform* (FFT) was introduced. By dividing the time series into smaller ones and adding these together, the FFT can perform the discrete transform using only $\mathcal{O}(N)log_2N$ operations [5].

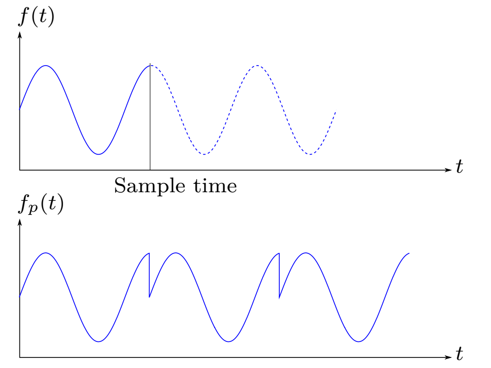
2.4.2 Leakage, windowing and estimation of spectra

FFT expands the sampled signal to a periodic signal. For a signal that is periodic in its nature, like the pure sine wave shown in Figure 2.7, sampling should be done with caution to the wave length of the signal. When this is not done, it results in unwanted artefacts, as shown in Figure 2.7(a). The same would always be the case for a signal that is not periodic in its nature, *e.g.* like a sampled random displacement. This is why *window functions* may have to be used when performing FFT on a non-periodic signal.

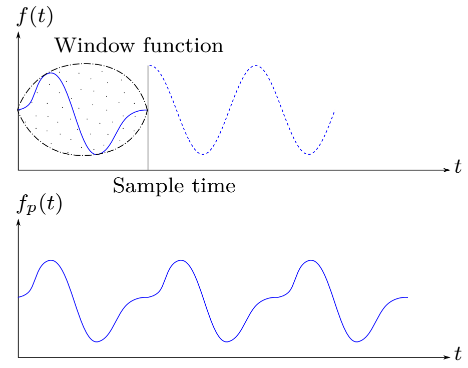
Harris [11] refers to window functions as weighting functions, which are used on the data set at hand, to reduce *spectral leakage*. Spectral leakage can, according to Harris, intuitively be understood as an effect caused by the fact that signals with other frequencies than the ones used in the basis set will be non-periodic in the window the signal is observed. For further insight it is referred to [11].

Window functions should nevertheless be used with great caution, as FFT itself should be. Figure 2.8 shows two estimations of the spectrum corresponding to a simple sine wave, 3.75 periods long, with and without applying a window function. The window function used here, is introduced in [12]:

$$W(n) = 1 - \left(\frac{n - \frac{N-1}{2}}{\frac{N+1}{2}} \right)^2 \tag{2.73}$$



(a) No window fucntion is introduced. This results in unwanted artefacts in the periodic expansion.

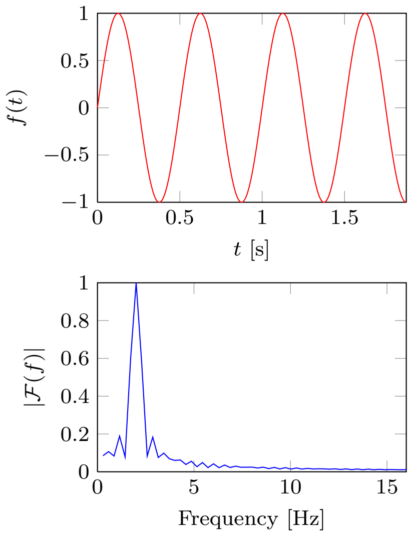


(b) A window function is introduced. This deals with the unwanted artefacts from the periodic expansion.

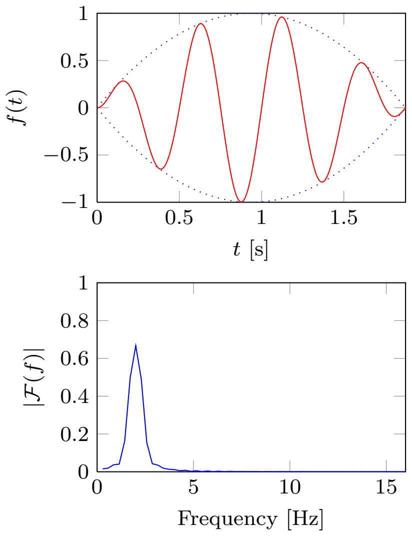
Figure 2.7: *Sampled part of sine wave $f(t)$ expanded to a periodic function $f_p(t)$.*

where n is the sample number and N the total number of samples. The windowed signal results in an FFT with a smaller amplitude. This is as expected, since some energy is lost due to the envelope in the windowing process.

Welch introduces a method of estimating the power spectra from time series in [12]. This method is based on averaging estimated spectra from short sections of the time series at hand, which results in lower computational time and improved results.



(a) No window function.



(b) Welch window function.

Figure 2.8: *Fast Fourier transform of* 3.75 *periods of a simple sine wave with frequency* $2Hz$.

2.5 Random data

Bendat and Piersol [13] define a random process as a process with different outcomes each realization. See Figure 2.9. Therefore, each observed behaviour only represents one of many possible outcomes. To mathematically describe these kind of processes, a statistical or stochastic description is needed.

Some important statistical sizes from Walpole *et al.* [14] are introduced below. The probability distribution of x is $f(x)$, such that $\int_{-\infty}^{\infty} f(x) \equiv 1$.

Expectancy-value for the variable $x(t)$ is defined

$$\mu_x = E(x) = \int_{-\infty}^{\infty} x f(x) dx \tag{2.74}$$

Variance of the variable $x(t)$ is defined

$$Var\ x = \sigma_x^2 = E[(x - \mu_x)^2] = \int_{-\infty}^{\infty} (x - \mu_x)^2 f(x) dx \tag{2.75}$$

Covariance between the variables $x(t)$ and $y(t)$ is defined

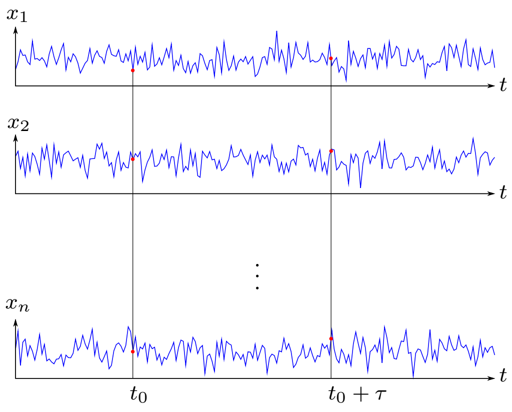


Figure 2.9: *Different realizations of the same process.*

$$\sigma_{xy} = E[(x(t)-\mu_x)(y(t)-\mu_y)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x(t)-\mu_x)(y(t)-\mu_y)f(x,y)dxdy \tag{2.76}$$

where $f(x,y)$ is introduced as the joint probability distribution between the variables x and y. By comparison with Equation 2.75 it is seen that $\sigma_{xx} = \sigma_x^2$.

■

2.5.1 Characteristic processes

Gaussian process

A process described by the variable $x(t)$ follows a Gaussian distribution if it is described by the following probability density function

$$f(x) = (c\sqrt{2\pi})^{-1}e^{-\frac{(x-a)^2}{2c^2}} \tag{2.77}$$

where a is a real constant and c is a positive constant [13]. The normal distribution is described by letting $a = \sigma$ (standard deviation) and $c = \mu$ (expectancy value). Physical processes can often be described successfully using Gaussian distribution. One important reason for this is the *central limit theorem.* According to this theorem, the sum of independent and random variables, can be described using a Gaussian distribution when many variables are included [5].

Stationary process

According to Naess [15], a process is *weakly stationary* if both $E[x(t)]$ and $E[x(t)x(t+\tau)]$ are independent of t. $E[x(t)x(t+\tau)]$ is called the *autocorrelation function* of x.

Ergodic process

Consider the realizations in Figure 2.9. In general, statistical estimates of the process studied can be determined at a time instance, by averaging the values of the different realizations at that time instance. According to Bendat and Piersol [13], a random process which is stationary and has values of autocorrelation and mean value independent of the realization studied, is said to be ergodic.

2.5.2 Covariance

The cross-covariance or covariance between two variables x and y is defined [13]

$$C_{x,y}(t,\tau) = E[(x_k(t) - \mu_x(t))(y_k(t+\tau) - \mu_y(t+\tau))] \tag{2.78}$$

where E[·] is the expected value of [·], μ_x and μ_y the expected values of the variables x and y, and τ a variable time shift. For a stationary and ergodic process this reduces to

$$C_{x,y}(\tau) = E[(x(t) - \mu_x)(y(t+\tau) - \mu_y)] \tag{2.79}$$

For the special case of $x = y$ we have *auto-covariance*, which becomes

$$C_x(\tau) = E[(x(t) - \mu_x)(x(t+\tau) - \mu_x)] \tag{2.80}$$

For loads and displacements used with linear Finite Element Method (FEM), it is reasonable to set the expectancy value equal to zero. This leads to the following expression for the covariance function:

$$C_{x,y}(\tau) = E[x(t)y(t+\tau)] = E[y(t)x(t+\tau)] \tag{2.81}$$

and the auto-covariance of the process x becomes

$$C_x(\tau) = E[x(t)x(t+\tau)] \tag{2.82}$$

Correlation

Langen and Sigbjörnsson [2] define the *cross-correlation coefficient function* as

$$\rho_{xy}(t_1, t_2) = \frac{C_{x,y}(t_1, t_2)}{\sigma_x(t_1)\sigma_y(t_2)} \tag{2.83}$$

with bounds

$$|\rho_{xy}(t_1, t_2)| \leq 1 \tag{2.84}$$

2.5.3 Variance spectrum

Bendat and Piersol [13] suggest three equivalent ways to establish the *covariance spectrum* between two processes:

1. Using correlation functions
2. Using Fourier transform directly on realizations
3. Using filtering-squaring-averaging operations

The two first methods will be briefly explained below. For more information about the subject, it is referred to [13].

Using correlation functions. The variance spectrum is defined [2] as the Fourier transform of the covariance;

$$S_{x,y}(\omega) = \mathcal{F}[C_{x,y}(\tau)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} C_{x,y}(\tau)e^{-i\omega\tau}d\tau \tag{2.85}$$

where the inverse Fourier transform becomes:

$$C_{x,y}(\tau) = \mathcal{F}^{-1}[S_{x,y}(\omega)] = \int_{-\infty}^{\infty} S_{x,y}(\omega)e^{i\omega\tau}d\omega \tag{2.86}$$

For the auto-covariance spectrum, or *auto-spectral density*, the same definition yields

$$S_x(\omega) = \mathcal{F}[C_x(\tau)] \tag{2.87}$$

Using Fourier transform directly on realizations. The spectral density of the variance can also be defined using the original signal data. Strømmen [16] introduces the following representation of the stationary variables x and y, both with length T:

$$x(t) = \lim_{N\to\infty} \sum_{m=-N}^{N} X_m(\omega_m, t) \tag{2.88}$$

$$y(t) = \lim_{N\to\infty} \sum_{m=-N}^{N} Y_m(\omega_m, t) \tag{2.89}$$

where $X_m(\omega_m, t)$ and $Y_m(\omega_m, t)$ are the mth harmonic components of x and y. These are defined [16]

$$X_m(\omega_m, t) = \frac{1}{T} c_{X_m} e^{i\omega_m t} \tag{2.90}$$

$$Y_m(\omega_m, t) = \frac{1}{T} c_{Y_m} e^{i\omega_m t} \tag{2.91}$$

Here, the coefficients c_{X_m} and c_{Y_m} are the complex Fourier coefficients. c_{X_m} is defined

$$c_{X_m} = \lim_{T\to\infty} \int_{-\frac{T}{2}}^{\frac{T}{2}} x(t) e^{-i\omega_m t} dt \tag{2.92}$$

and c_{Y_m} equivalently. [16] further defines the *cross-spectral density* as

$$S_{xy}(\omega_m) = \frac{E[X_m^* Y_m]}{\Delta\omega} \tag{2.93}$$

When using that

$$E[X_m^* Y_m] = \frac{1}{T} \int_0^T \frac{1}{T} c_{X_m}^* e^{-i\omega_m t} \cdot \frac{1}{T} c_{Y_m} e^{i\omega_m t} = \frac{1}{T^2} c_{X_m}^* c_{Y_m} \tag{2.94}$$

this can be written

$$S_{xy}(\omega_m) = \frac{c_{X_m}^* c_{Y_m}}{2\pi T} \tag{2.95}$$

A continuous representation is found by letting T and N go towards infinity:

$$S_{xy}(\omega) = \lim_{T\to\infty} \lim_{N\to\infty} \frac{c_X^*(\omega) c_Y(\omega)}{2\pi T} \tag{2.96}$$

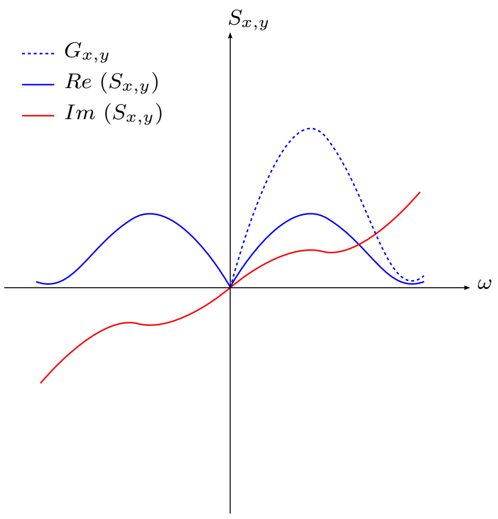


Figure 2.10: *Variance spectrum with real and imaginary parts, including a one-sided representation of the real parts.*

One-sided spectral density functions

According to Bendat and Piersol [13], the *one-sided* cross-spectral density is defined

$$G_{x,y}(\omega) = 2S_{x,y}(\omega) \qquad \text{for} \quad \omega \geq 0 \tag{2.97}$$

This is illustrated in Figure 2.10. The relations between load and response spectra defined later, are equivalent for one- and two-sided spectra. In the calculations in the project at hand, two-sided spectra are used. They are represented with positive frequencies only, due to the symmetry properties of the spectrum.

White noise spectrum

Bendat and Piersol [13] define *white noise* as a process with constant auto-spectral density. Assume a process with white noise, which has a uniformly distributed probability density function, and upper and lower bounds A_0 and $-A_0$. Such a process is presented in Figure 2.11. By studying this figure, and using the definition of variance given in Equation 2.75, it is easily seen that

$$\sigma_x^2 = \int_{-\infty}^{\infty} x^2 f(x) dx = \frac{1}{3} A_0^2 \tag{2.98}$$

White noise will be used as load input for the systems studied later in the report.

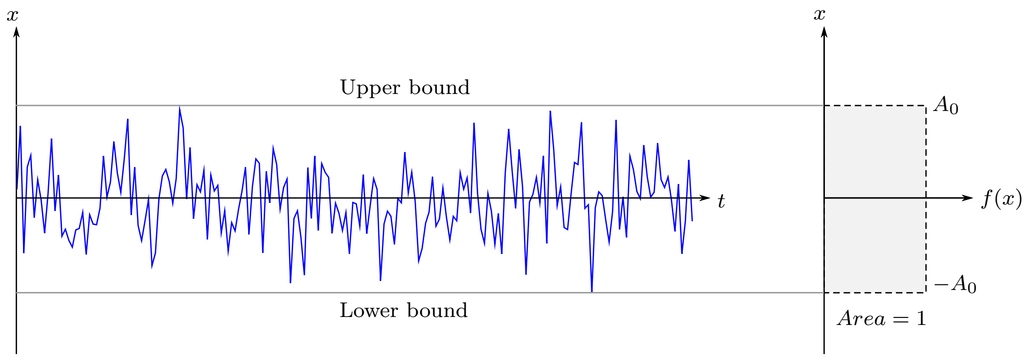


Figure 2.11: *Uniformly distributed white noise with bounds* $\pm A_0$*.*

Variance and covariance

When choosing $\tau = 0$, the cross-covariance function in Equation 2.78 reduces to

$$C_{x,y}(0) = E[x(t)y(t)] \tag{2.99}$$

This matches the definition of covariance in Equation 2.76 for $\mu_x = \mu_y = 0$. Thus, $C_{x,y}(0) = \sigma_{xy}$. Using $\tau = 0$ in Equation 2.86 yields

$$\sigma_{x,y} = C_{x,y}(0) = \int_{-\infty}^{\infty} S_{x,y}(\omega)d\omega \tag{2.100}$$

and therefore also

$$\sigma_x^2 = C_{x,x}(0) = \int_{-\infty}^{\infty} S_x(\omega)d\omega \tag{2.101}$$

Coherence spectrum

According to [2], the *coherence spectrum* is defined

$$\gamma_{xy}(\omega) = \frac{|S_{xy}(\omega)|^2}{S_x(\omega)S_y(\omega)} \tag{2.102}$$

and has the constraints

$$0 \le \gamma_{xy}(\omega) \le 1 \tag{2.103}$$

By comparison with Equation 2.83, this can be interpreted as the frequency domain counterpart of correlation.

2.5.4 Spectra in multi-variable systems

The covariance matrix [15] is defined

$$[\mathbf{C_x}] = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots \\ C_{2,1} & C_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{2.104}$$

where the elements of the matrix are defined as in Equation 2.78. When $\mu = 0$ for all variables, Equation 2.78 gives

$$[\mathbf{C_x}] = \begin{bmatrix} E[x_1(t) \cdot x_1(t+\tau)] & E[x_1(t) \cdot x_2(t+\tau)] & \cdots \\ E[x_2(t) \cdot x_1(t+\tau)] & E[x_2(t) \cdot x_2(t+\tau)] & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{2.105}$$

This can be rewritten

$$[\mathbf{C_x}] = E\left[\begin{Bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \end{Bmatrix} \begin{bmatrix} x_1(t+\tau) & x_2(t+\tau) & x_3(t+\tau) & \cdots \end{bmatrix}\right] \tag{2.106}$$

which leads to

$$[\mathbf{C_x}] = E\left[\begin{Bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \vdots \end{Bmatrix} \begin{Bmatrix} x_1(t+\tau) \\ x_2(t+\tau) \\ x_3(t+\tau) \\ \vdots \end{Bmatrix}^T\right] = E\left[\{\mathbf{x}(t)\}\{\mathbf{x}(t+\tau)\}^T\right] \tag{2.107}$$

Further, the covariance spectrum matrix is defined

$$[\mathbf{S_x}] = \begin{bmatrix} S_{1,1} & S_{1,2} & \cdots \\ S_{2,1} & S_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{2.108}$$

where the elements of the matrix are defined as in Equation 2.85. Using this gives

$$[\mathbf{S_x}] = \begin{bmatrix} \frac{1}{2\pi}\int_{-\infty}^{\infty} C_{1,1}(\tau)e^{-i\omega\tau}d\tau & \frac{1}{2\pi}\int_{-\infty}^{\infty} C_{1,2}(\tau)e^{-i\omega\tau}d\tau & \cdots \\ \frac{1}{2\pi}\int_{-\infty}^{\infty} C_{2,1}(\tau)e^{-i\omega\tau}d\tau & \frac{1}{2\pi}\int_{-\infty}^{\infty} C_{2,2}(\tau)e^{-i\omega\tau}d\tau & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{2.109}$$

which from factorization further gives

$$[\mathbf{S_x}] = \frac{1}{2\pi}\int_{-\infty}^{\infty} \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots \\ C_{2,1} & C_{2,2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} e^{-i\omega\tau}d\tau = \frac{1}{2\pi}\int_{-\infty}^{\infty} [\mathbf{C_x}]\, e^{-i\omega\tau}d\tau = \mathcal{F}\,[\mathbf{C_x}] \tag{2.110}$$

Using Equation 2.107, this gives

$$[\mathbf{S_x}] = \mathcal{F}\left(E\left[\{\mathbf{x}(t)\}\{\mathbf{x}(t+\tau)\}^T\right]\right) \tag{2.111}$$

2.6 Frequency domain dynamics for a floating bridge

Assume a bridge is made up of a structural part, *i.e.* the frame, and floating elements, hereafter called pontoons. By using FEM, the equation of motion for the structural part of the floating bridge can be written

$$[\mathbf{M}_s]\{\ddot{\mathbf{u}}(t)\} + [\mathbf{C}_s]\{\dot{\mathbf{u}}(t)\} + [\mathbf{K}_s]\{\mathbf{u}(t)\} = \{\mathbf{p}_h(t)\} \tag{2.112}$$

where $\{\mathbf{p}_h\}$ is the hydrodynamic load vector [17]. The floating elements contribute with forces from the interaction between the fluid and the structure. These forces are dependent of displacements, velocities and accelerations of the pontoons, which give rise to hydrodynamic and hydrostatic mass, damping and stiffness. The matrices are shown in Table 2.1. This results in the following total system matrices:

$$[\mathbf{M}(\omega)] = [\mathbf{M}_s] + [\mathbf{M}_h(\omega)] \tag{2.113}$$

$$[\mathbf{C}(\omega)] = [\mathbf{C}_s] + [\mathbf{C}_h(\omega)] \tag{2.114}$$

$$[\mathbf{K}] = [\mathbf{K}_s] + [\mathbf{K}_h] \tag{2.115}$$

This yields the following equations of motion for the coupled system

$$[\mathbf{M}(\omega)]\{\ddot{\mathbf{u}}(t)\} + [\mathbf{C}(\omega)]\{\dot{\mathbf{u}}(t)\} + [\mathbf{K}]\{\mathbf{u}(t)\} = \{\mathbf{p}(t)\} \tag{2.116}$$

Table 2.1: *Additions to system matrices from pontoons.*

	Stiffness	Damping	Mass
Frequency dependent		$\left[\mathbf{C}_{h,\omega}(\omega)\right]$	$\left[\mathbf{M}_{h,\omega}(\omega)\right]$
Frequency independent	$\left[\mathbf{K}_{h,0}\right]$		$\left[\mathbf{M}_{h,0}\right]$
Total	$[\mathbf{K}_h]$	$[\mathbf{C}_h(\omega)]$	$[\mathbf{M}_h(\omega)]$

where $\{\mathbf{p}(t)\}$ is the total load vector acting on the bridge. As stated in Langen and Sigbjörnsson [2], the response can be written as a harmonic decomposition:

$$\{\mathbf{u}(t)\} = \int_{-\infty}^{\infty} e^{i\omega t} d\{\mathbf{Z_u}(\omega)\} \tag{2.117}$$

where the spectral process corresponding to the displacement vector $\{\mathbf{u}(t)\}$ is introduced as $\{\mathbf{Z}_u(\omega)\}$. It follows directly from this that the velocity and acceleration vectors become

$$\{\dot{\mathbf{u}}(t)\} = \int_{-\infty}^{\infty} i\omega e^{i\omega t} d\{\mathbf{Z_u}(\omega)\} \tag{2.118}$$

$$\{\ddot{\mathbf{u}}(t)\} = \int_{-\infty}^{\infty} -\omega^2 e^{i\omega t} d\{\mathbf{Z_u}(\omega)\} \tag{2.119}$$

Similarly, the load vector $\{\mathbf{p}(t)\}$ can be written using the same technique

$$\{\mathbf{p}(t)\} = \int_{-\infty}^{\infty} e^{i\omega t} d\{\mathbf{Z_p}(\omega)\} \tag{2.120}$$

where $\{\mathbf{Z_p}(\omega)\}$ is the spectral process corresponding to the load vector $\{\mathbf{p}(t)\}$. Using Equations 2.117-2.120, Equation 2.116 can be written

$$\int_{-\infty}^{\infty} \left(-\omega^2 \left[\mathbf{M}(\omega)\right] + i\omega \left[\mathbf{C}(\omega)\right] + [\mathbf{K}]\right) e^{i\omega t} d\{\mathbf{Z}_u(\omega)\} = \int_{-\infty}^{\infty} e^{i\omega t} d\{\mathbf{Z_p}(\omega)\} \tag{2.121}$$

The frequency response function can be introduced into Equation 2.121, as described in Section 2.2.1. This yields

$$d\{\mathbf{Z_u}(\omega)\} = [\mathbf{H}(\omega)]\, d\{\mathbf{Z_p}(\omega)\} \tag{2.122}$$

For a MDOF system, Langen and Sigbjörnsson [2] express the spectral density of the displacement process $\{\mathbf{u}(t)\}$ as

$$[\mathbf{S_u}(\omega)] = E[d\{\mathbf{Z_u}(\omega)\}d\{\mathbf{Z_u}(\omega)\}^*] \tag{2.123}$$

where the operator $[\cdot]^*$ is used as complex conjugate and matrix transpose. Combining this with Equation 2.122 gives

$$[\mathbf{S_u}(\omega)] = [\mathbf{H}(\omega)]\,[\mathbf{S_p}(\omega)]\,[\mathbf{H}(\omega)]^* \tag{2.124}$$

where a definition analogous to the spectral density of the displacement, is introduced for the load spectral density. From this, it follows that the response spectra matrix $[\mathbf{S_u}(\omega)]$ is a *Hermittian symmetric* matrix [2]. This implies that the real parts of the matrix are symmetric, while the imaginary parts are negative symmetric. In other words, the element (i, j) is the complex conjugate of the element (j, i).

2.6.1 Hydrodynamic loading

The total hydrodynamic loading, from both waves and displacements of the construction itself can, according to [17], be written

$$\{\mathbf{p}_h\}(t) = \{\mathbf{p}_w\}(t) - \{\mathbf{p}_m\}(t) - \{\mathbf{p}_c\}(t) - \{\mathbf{p}_k\}(t) \tag{2.125}$$

in the time domain. Here, $\{\mathbf{p}_w\}$ is the force excited by the waves. The fluid-structure interaction gives rise to $\{\mathbf{p}_m\}$, $\{\mathbf{p}_c\}$ and $\{\mathbf{p}_k\}$, which are the hydrodynamic inertia forces, the hydrodynamic damping forces and the hydrostatic buoyancy forces, respectively. The parts of the hydrodynamic load can be written as the sum of harmonic components, as done for the development of Equation 2.117. Doing that, the total load vector acting on the structure can be written

$$\begin{aligned}\{\mathbf{p}_h(t)\} =&\{\mathbf{p}_w(t)\} - \int_{-\infty}^{\infty} -\omega^2\,[\mathbf{M}_h(\omega)]\,e^{i\omega t} d\{\mathbf{Z_u}\} \\ &- \int_{-\infty}^{\infty} i\omega\,[\mathbf{C}_h(\omega)]\,e^{i\omega t} d\{\mathbf{Z_u}\} - \int_{-\infty}^{\infty} [\mathbf{K}_h]\,e^{i\omega t} d\{\mathbf{Z_u}\}\end{aligned} \tag{2.126}$$

2.6.2 Wave action in three dimensions

The elevation of the sea surface is assumed to be a ergodic Gaussian process, with $\mu = 0$, in [2]. Assuming that the sea elevation is an ergodic and Gaussian process, the auto-covariance function of the loads due to the waves can expressed using Equation 2.82. This leads to

$$C_p(\tau) = E[p(t)p(t+\tau)] \tag{2.127}$$

where $p(t)$ is the amplitude of the wave load. Using Equation 2.85, this gives the following variance spectrum:

$$S_p(\omega) = \mathcal{F}[C_p(\tau)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} C_p(\tau)e^{-i\omega\tau}d\tau \tag{2.128}$$

For the full three dimensional case, [17] states that for a short-term analysis the wave situation is completely described by the cross-spectral density

$$S_{\eta_m\eta_n}(\omega) = S_{\eta\eta}(\omega)\int_{\theta} D(\theta)e^{B(\omega)}d\theta \tag{2.129}$$

where $S_{\eta\eta}$ is the 1D wave spectrum, η_n is corresponding to the wave amplitude in $x = x_n$ and $y = y_n$, $D(\theta,\omega)$ is the spreading function describing the spreading of the spectrum in angular direction and the function $B(\omega,\theta)$ is introduced for simplicity:

$$B(\omega,\theta) = -i\frac{\omega}{|\omega|}\frac{\omega^2}{g}\cdot[(y_m - y_n)cos(\theta) - (x_m - x_n)sin(\theta)] \tag{2.130}$$

The expression for the spreading function can be approached by the following frequency independent function [2]

$$D(\theta) = C\cdot cos^n(\theta-\theta_0) \tag{2.131}$$

Here, θ_0 corresponds to the mean direction of wave propagation, n defines the distribution width and C is a normalization factor. The total load vector spectra can further be expressed

$$[\mathbf{S}_p(\omega)] = [\mathbf{F}(\omega)]\,[\mathbf{S}_\eta(\omega)] \tag{2.132}$$

where $[\mathbf{S}_\eta(\omega)]$ is the matrix constructed by the elements defined in Equation 2.129 and $[\mathbf{F}(\omega)]$ is the *hydrodynamic transfer function.* No further attention is directed towards the hydrodynamic transfer function, and the reader is referred to [2] for further insight.

Jonswap spectrum

One commonly used model for one dimensional wave spectra is the *Jonswap* spectrum. The Jonswap spectrum is defined [2] as

$$S_{\eta\eta}(\omega) = \alpha g^2\omega^{-5}\cdot exp\left(-\beta\left(\frac{\omega}{\omega_n}\right)^{-4}\gamma\right)\cdot exp\left(\frac{-(\omega-\omega_{peak})^2}{2\sigma^2\omega_{peak}^2}\right) \tag{2.133}$$

where α, β, γ and σ are spectral parameters given specific values for the case being modelled, ω_{peak} is the natural frequency corresponding to the maximum amplitude and g is the gravitational constant. The Jonswap spectrum will be used on the example in

Table 2.2: *Parameters used with Jonswap spectrum.*

Parameter	Value	Comment
α	1.5	-
β	1.25	Value from [2].
γ	7.0	Spreading parameter. Value from [2].
σ	0.07	Value from [2].
ω_p	1.5 $\frac{rad}{s}$	Peak frequency of spectrum.

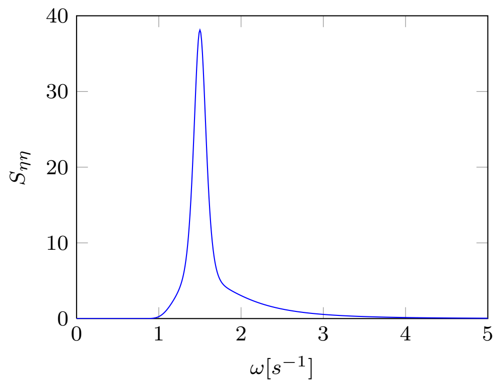


Figure 2.12: *Load spectra component $S_{\eta\eta}$, as defined by Equation 2.133, with the parameter values set as defined in Table 2.2.*

Section 2.6.3, to establish a load spectrum for a submerged shear frame. Figure 2.12 shows the Jonswap spectrum for spectral values chosen according to Table 2.2.

2.6.3 Example: submerged two-storey shear frame

In this section, a partially submerged two-storey shear frame, as illustrated in Figure 2.13, is considered. It is assumed that the vertical beams are massless. Further, the horizontal beams are considered infinitely stiff compared to the vertical ones, which gives a simple two-DOF system. This system is used to study the concept of covariance spectra. It is assumed that the wave hits directly along DOF u_1. The aerodynamic effects that comes from the submerging of the lower parts of the columns are neglected, and only the stochastic load directly along u_1 from the water considered.

The mass and stiffness of the shear frame are found in the example in Section 2.3. Equations 2.60 and 2.61 read out

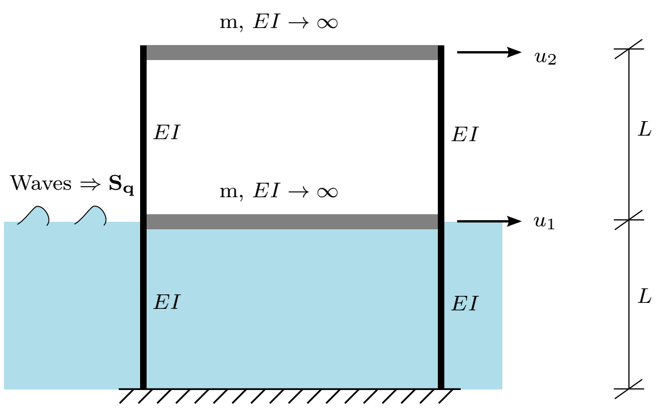


Figure 2.13: *Two-storey shear frame exposed to stochastic wave load. The load is assumed to act only along DOF u_1.*

$$\begin{aligned}\mathbf{K} &= \begin{bmatrix} 2k & -k \\ -k & k \end{bmatrix} \\ \mathbf{M} &= \begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix}\end{aligned}$$

Three cases are studied; one without damping, one with a diagonal damping matrix equal that shown in Equation 2.65, and one with a non-diagonal symmetric damping matrix, as the one in Equation 2.64. As discussed in Section 2.3, the last case will yield complex eigenvalues.

The values of the stiffness, damping and mass constants are set to

`k=6, c=2, m=2`

The Jonswap spectrum is used for the auto-covariance spectrum of the wave along DOF u_1, S_{η_1,η_1}. The transfer function is used as unity, such that $S_{p,11} = 1 \cdot S_{\eta_1,\eta_1}$ Also, we have that the auto-covariance spectrum of the load for u_2 is zero, i.e. $S_{p,22} = 0$ for all frequencies, and that the cross-spectrum terms, $S_{p,12}$ and $S_{p,21}$ are zero. From this, the load spectra matrix reads out

$$\mathbf{S_p} = \begin{bmatrix} S_{p,11} & S_{p,12} \\ S_{p,21} & S_{p,22} \end{bmatrix} = \begin{bmatrix} S_{p,11} & 0 \\ 0 & 0 \end{bmatrix} \tag{2.134}$$

The chosen values for the Jonswap spectrum are shown in Table 2.2. Also, the frequency range considered is ranging between 0 and 5 $\frac{rad}{s}$, and the gravitational constant set equal to $9.81 m/s^2$. This resulted in the load spectrum shown in Figure 2.12, which gave response

Table 2.3: *Eigenfrequencies for the three different cases studied.*

Case	$\omega_1[rad \cdot s^{-1}]$	$\omega_2[rad \cdot s^{-1}]$
No damping	1.0705	2.8025
Diagonal damping	1.0705	2.8025
Symmetric non-diagonal damping	1.0742	2.7928

spectra for both DOFs, including cross-terms, as shown in Figure 2.14. For no damping, the response spectrum, shown in Figure 2.14(a), reveal a clear resonance peak. The MATLAB script used is shown in Appendix B.10. The eigenfrequencies produced by these three cases are shown in Table 2.3.

2.6.4 Modal representation of response variance spectra

Have the following expression for the auto-covariance of the physical DOFs, $\{\mathbf{u}\}$

$$[\mathbf{C_u}(\tau)] = E[\{\mathbf{u}(t)\}\{\mathbf{u}(t+\tau)\}^T] \tag{2.135}$$

Similarly, for another set of generalized coordinates:

$$[\mathbf{C_y}(\tau)] = E[\{\mathbf{y}(t)\}\{\mathbf{y}(t+\tau)\}^T] \tag{2.136}$$

The displacement can be decomposed into modal DOFs like this

$$\{\mathbf{u}\} = [\mathbf{\Phi}]\{\mathbf{y}\} \tag{2.137}$$

as discussed in Section 2.2.2. Substitution of this into Equation 2.135 gives

$$[\mathbf{C_u}(\tau)] = E\left([\mathbf{\Phi}]\{\mathbf{y}(t)\}\{\mathbf{y}(t+\tau)\}^T[\mathbf{\Phi}]^T\right) = [\mathbf{\Phi}][\mathbf{C_y}(\tau)][\mathbf{\Phi}]^T \tag{2.138}$$

By substitution into Equation 2.85, this gives

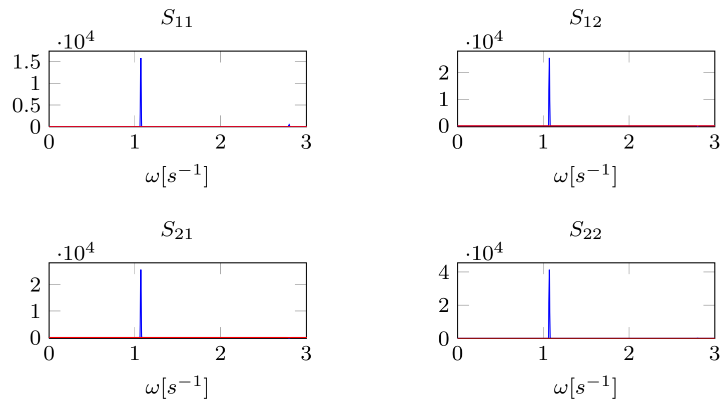
$$[\mathbf{S_u}] = \frac{1}{2\pi}\int_{-\infty}^{\infty}[\mathbf{\Phi}][\mathbf{C_y}(\tau)][\mathbf{\Phi}]^T e^{-i\omega\tau}d\tau \tag{2.139}$$

From the fact that $\mathbf{\Phi}$ is independent of the time, this can be written

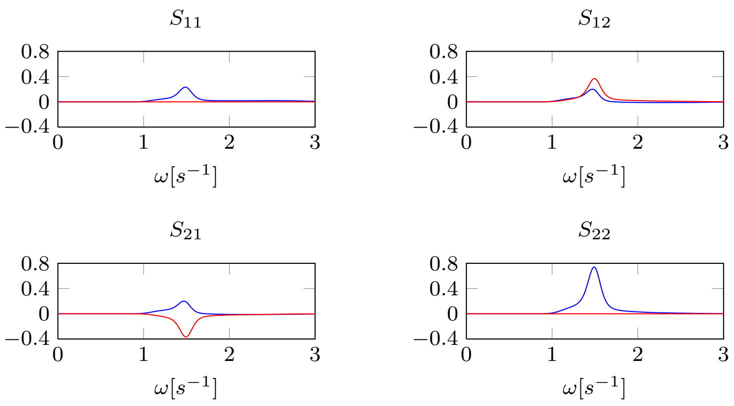
$$[\mathbf{S_u}] = [\mathbf{\Phi}]\frac{1}{2\pi}\int_{-\infty}^{\infty}[\mathbf{C_y}(\tau)]e^{-i\omega\tau}d\tau\,[\mathbf{\Phi}]^T \tag{2.140}$$

The definition of the response spectrum in modal coordinates equals that for the physical ones. Therefore

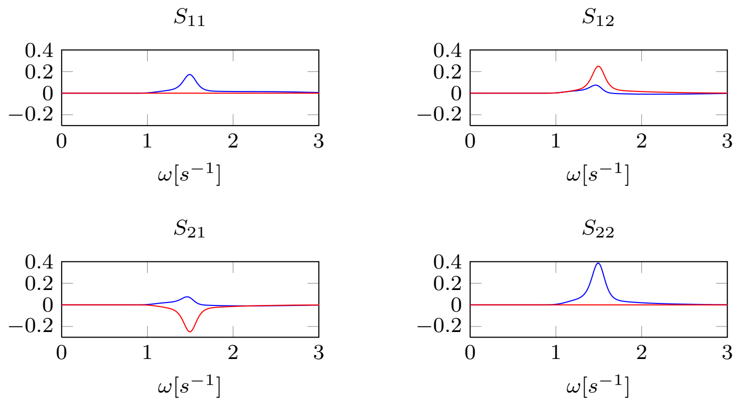
$$[\mathbf{S_u}] = [\mathbf{\Phi}][\mathbf{S_y}][\mathbf{\Phi}]^T \tag{2.141}$$



(a) No damping. This gives no imaginary parts for the response spectra.



(b) Diagonal damping matrix as shown in Equation 2.65 used.



(c) Non-diagonal symmetric damping matrix as shown in Equation 2.64 used.

Figure 2.14: *Variance spectra for response of the shear frame shown in Figure 2.13, exposed to wave load given by the wave spectrum shown in Figure 2.12. It is noted that significant response is found also for the components without loading. Imaginary parts are plotted in red, while the real parts are plotted in blue.*

2.6.5 Modal representation of load variance spectra

A load vector $\{\mathbf{p}\}$, acts along the assigned DOFs, $\{\mathbf{u}\}$. The load has the following covariance matrix:

$$[\mathbf{C}_\mathbf{p}(\tau)] = E[\{\mathbf{p}(t)\}\{\mathbf{p}(t+\tau)\}^T] \tag{2.142}$$

and a similar expression for the modally transformed load along generalized DOFs, $\{\mathbf{y}\}$:

$$\left[\tilde{\mathbf{C}}_\mathbf{p}(\tau)\right] = E[\{\tilde{\mathbf{p}}(t)\}\{\tilde{\mathbf{p}}(t+\tau)\}^T] \tag{2.143}$$

As described in Section 2.2.2, the load is transformed into modal load like this

$$\{\tilde{\mathbf{p}}\} = [\mathbf{\Phi}]^T \{\mathbf{p}\} \tag{2.144}$$

Substitution into Equation 2.143 gives

$$\left[\tilde{\mathbf{C}}_\mathbf{p}(\tau)\right] = E\left([\mathbf{\Phi}]^T \{\mathbf{p}(t)\}\{\mathbf{p}(t+\tau)\}^T [\mathbf{\Phi}]\right) \tag{2.145}$$

which can be rewritten

$$\left[\tilde{\mathbf{C}}_\mathbf{p}(\tau)\right] = [\mathbf{\Phi}]^T E\left(\{\mathbf{p}(t)\}\{\mathbf{p}(t+\tau)\}^T\right) [\mathbf{\Phi}] \tag{2.146}$$

The definition of variance spectra finally gives

$$\left[\tilde{\mathbf{S}}_\mathbf{p}\right] = [\mathbf{\Phi}]^T [\mathbf{S}_\mathbf{p}] [\mathbf{\Phi}] \tag{2.147}$$

2.6.6 Modal transformations for pontoon action

When only the load from the sea is considered, the bridge will only be exposed to loading in certain DOFs. The same goes for the addition to total system matrices; they also only apply in the global DOFs where the pontoons are attached to the structure. Assume that the DOFs are arranged in this convenient manner:

$$\{\mathbf{u}\} = \begin{Bmatrix} \{\mathbf{u}_p\} \\ \{\mathbf{u}_0\} \end{Bmatrix} \tag{2.148}$$

where $\{\mathbf{u}_p\}$ are the DOFs of all pontoons, and $\{\mathbf{u}_0\}$ are the remaining DOFs. This gives the following structure of the $[\mathbf{\Phi}]$-matrix for N modes:

$$[\mathbf{\Phi}] = \begin{bmatrix} \{\phi_p\}_1 & \{\phi_p\}_2 & \cdots & \{\phi_p\}_N \\ \{\phi_0\}_1 & \{\phi_0\}_2 & \cdots & \{\phi_0\}_N \end{bmatrix} \tag{2.149}$$

where $\begin{Bmatrix} \{\phi_p\}_j \\ \{\phi_0\}_j \end{Bmatrix}$ is the total $\{\phi\}$-vector for mode j. Further, the mass contribution to the system from the pontoons can be written

$$[\mathbf{M}_h] = \begin{bmatrix} [\mathbf{M}_{pp}] & [\mathbf{M}_{p0}] \\ [\mathbf{M}_{0p}] & [\mathbf{M}_{00}] \end{bmatrix} \tag{2.150}$$

Since no contribution exists for the DOFs other than those of the pontoons, this can be written

$$[\mathbf{M}_h] = \begin{bmatrix} [\mathbf{M}_{pp}] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{2.151}$$

Further, this gives the following modally transformed mass contribution from the pontoons:

$$[\tilde{\mathbf{M}}_h] = [\mathbf{\Phi}]^T[\mathbf{M}_h][\mathbf{\Phi}] \tag{2.152}$$

$$= \begin{bmatrix} \{\phi_p\}_1^T & \{\phi_0\}_1^T \\ \{\phi_p\}_2^T & \{\phi_0\}_2^T \\ \vdots & \vdots \\ \{\phi_p\}_N^T & \{\phi_0\}_N^T \end{bmatrix} \begin{bmatrix} [\mathbf{M}_{pp}] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \{\phi_p\}_1 & \{\phi_p\}_2 & \cdots & \{\phi_p\}_N \\ \{\phi_0\}_1 & \{\phi_0\}_2 & \cdots & \{\phi_0\}_N \end{bmatrix} \tag{2.153}$$

$$= \begin{bmatrix} \{\phi_p\}_1^T \\ \{\phi_p\}_2^T \\ \cdots \\ \{\phi_p\}_N^T \end{bmatrix} [\mathbf{M}_{pp}] \begin{bmatrix} \{\phi_p\}_1 & \{\phi_p\}_2 & \cdots & \{\phi_p\}_N \end{bmatrix} \tag{2.154}$$

which yields

$$[\tilde{\mathbf{M}}_h] = [\mathbf{\Phi}_p]^T[\mathbf{M}_h][\mathbf{\Phi}_p] \tag{2.155}$$

The same holds for the stiffness, damping and load spectra matrices. By imposing this simplification, the $[\mathbf{\Phi}]$-matrix can be reduced to only contain $6 \cdot n_p$ rows, where n_p is the number of pontoons.

2.7 Rigid rotation transformation matrix

Assume a three-dimensional rigid body with six DOFs:

$$\{\mathbf{u}\} = \begin{Bmatrix} u_x \\ u_y \\ u_z \\ \theta_x \\ \theta_y \\ \theta_z \end{Bmatrix} \tag{2.156}$$

Here u_x, u_y and u_z are translations along the x-, y- and z-axis, and θ_x, θ_y and θ_z rotations about the x-, y- and z-axis, respectively.

A set of DOFs $\{\bar{\mathbf{u}}\}$ is rotated with an angle α in the xy-plane compared to $\{\mathbf{u}\}$. Figure 2.15 shows this. The DOFs in the z-direction are unaffected. If the translational DOFs are considered first, Figure 2.15 gives this decomposition

$$\bar{u}_x = cos(\alpha)u_x + sin(\alpha)u_y \tag{2.157}$$
$$\bar{u}_y = -sin(\alpha)u_x + cos(\alpha)u_y \tag{2.158}$$
$$\bar{u}_z = u_z \tag{2.159}$$
$$\tag{2.160}$$

which gives the following linear system

$$\begin{Bmatrix} \bar{u}_x \\ \bar{u}_y \\ \bar{u}_z \end{Bmatrix} = \begin{bmatrix} cos(\alpha) & sin(\alpha) & 0 \\ -sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} u_x \\ u_y \\ u_z \end{Bmatrix} \tag{2.161}$$

The rotational DOFs transform in the exact same manner;

$$\begin{Bmatrix} \bar{\theta}_x \\ \bar{\theta}_y \\ \bar{\theta}_z \end{Bmatrix} = \begin{bmatrix} cos(\alpha) & sin(\alpha) & 0 \\ -sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{Bmatrix} \tag{2.162}$$

Combining the two gives

$$\{\bar{\mathbf{u}}\} = [\mathbf{T}]\{\mathbf{u}\} \tag{2.163}$$

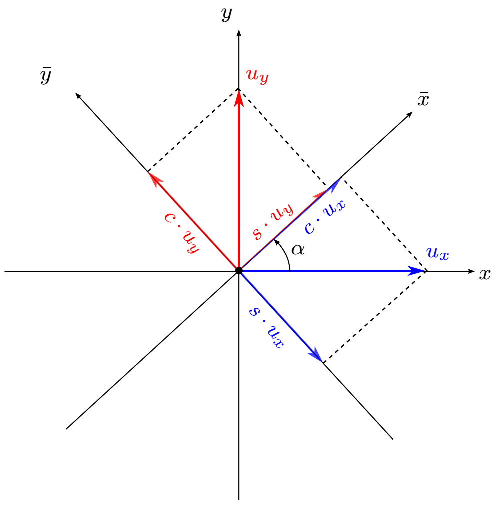


Figure 2.15: *Decomposition of vectors in different coordinate systems, where $s = sin(\alpha)$ and $c = cos(\alpha)$ are used.*

where

$$[\mathbf{T}] = \begin{bmatrix} cos(\alpha) & sin(\alpha) & 0 & 0 & 0 & 0 \\ -sin(\alpha) & cos(\alpha) & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos(\alpha) & sin(\alpha) & 0 \\ 0 & 0 & 0 & -sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.164}$$

The columns of the transformation matrix in Equation 2.161 can be interpreted as the base vectors of the original coordinate system (x,y,z) in the new coordinate system $(\tilde{x},\tilde{y},\tilde{z})$.

2.7.1 Transformation of load vector

The load vector $\{\bar{\mathbf{p}}\}$ works along the DOFs $\{\bar{\mathbf{u}}\}$, while $\{\mathbf{p}\}$ works along the DOFs $\{\mathbf{u}\}$. Requiring that the virtual work done in both coordinate systems equal, the following is obtained:

$$\{\delta\bar{\mathbf{u}}\}^T\{\bar{\mathbf{p}}\} = \{\delta\mathbf{u}\}^T\{\mathbf{p}\} \tag{2.165}$$

Equation 2.163 gives

$$\{\delta\bar{\mathbf{u}}\} = [\mathbf{T}]\{\delta\mathbf{u}\} \tag{2.166}$$

which inserted into Equation 2.165 results in

$$\{\delta\mathbf{u}\}^T[\mathbf{T}]^T\{\bar{\mathbf{p}}\} = \{\delta\mathbf{u}\}^T\{\mathbf{p}\} \tag{2.167}$$

This finally leads to

$$\{\mathbf{p}\} = [\mathbf{T}]^T\{\bar{\mathbf{p}}\} \tag{2.168}$$

2.7.2 Transformation of added system matrices

Because the matrices shown in Table 2.1 initially are related to the local coordinate system of a pontoon, a transformation of these matrices has to be performed, before adding these contributions to the overall modal properties of the system. Pontoon i is rotated an angle α_i from the global coordinate system used. Equation 2.163 gives

$$\{\bar{\mathbf{u}}_i\} = [\mathbf{T}_i]\{\mathbf{u}_i\} \tag{2.169}$$

where $\{\bar{\mathbf{u}}_i\}$ and $\{\mathbf{u}_i\}$ are the local and global oriented DOFs of pontoon i. $[\mathbf{T}_i]$ is the transformation matrix resulting from the rotation of pontoon i with an angle α_i. The principle of virtual work yields

$$\{\delta\mathbf{u}_i\}^T\left[\mathbf{K}_i\right]\{\mathbf{u}_i\} = \{\delta\bar{\mathbf{u}}_i\}^T\left[\bar{\mathbf{K}}_i\right]\{\bar{\mathbf{u}}_i\} \tag{2.170}$$

Introducing Equation 2.169, Equation 2.170 can be rewritten

$$\{\delta\mathbf{u}_i\}^T\left[\mathbf{K}_i\right]\{\mathbf{u}_i\} = \{\delta\mathbf{u}_i\}^T[\mathbf{T}_i]^T\left[\bar{\mathbf{K}}_i\right][\mathbf{T}_i]\{\mathbf{u}_i\} \tag{2.171}$$

which yields

$$[\mathbf{K}_i] = [\mathbf{T}_i]^T\left[\bar{\mathbf{K}}_i\right][\mathbf{T}_i] \tag{2.172}$$

The total global physical stiffness contribution from all pontoons becomes

$$[\mathbf{K}_h] = \begin{bmatrix} [\mathbf{T}_1]^T\left[\bar{\mathbf{K}}_1\right][\mathbf{T}_1] & \{\mathbf{0}\} & \cdots & \{\mathbf{0}\} \\ \{\mathbf{0}\} & [\mathbf{T}_2]^T\left[\bar{\mathbf{K}}_2\right][\mathbf{T}_2] & \cdots & \{\mathbf{0}\} \\ \vdots & \vdots & \ddots & \vdots \\ \{\mathbf{0}\} & \{\mathbf{0}\} & \cdots & [\mathbf{T}_n]^T\left[\bar{\mathbf{K}}_n\right][\mathbf{T}_n] \end{bmatrix} \tag{2.173}$$

where n is the total number of pontoons. The same holds for the contribution from the pontoons to the total system mass and damping.

3 Matlab program

To perform calculations on floating structures, the system studied is divided into two domains; the wet, *i.e.* from the interaction between the water and the pontoons, and the dry, *i.e.* the frame. A Matlab program was made, named `floatingBridge.m`, which tied these together, and performed needed calculations. This is shown in Appendix B.1. The main structure of the Matlab-program is shown in Figure 3.1, and the parts of the individual calculations explained in the following sections.

3.1 Solid structure

Abaqus/CAE is a commercial software package for advanced Finite Element Analysis (FEA). A data file resulting from a frequency job of the frame model in Abaqus/CAE is required. To get the displacements and the coordinates of all the nodes in the model printed in the resulting data file, the following lines should be added to the keywords of the Abaqus model:

```
*Output, history
*Node print
COORD, U
```

The resulting dat-file will further import into the Matlab program, using the subroutine `dat2eig.m`. This Matlab function is shown in Appendix B. It retrieves the coordinates of all nodes - and modal properties, such as modal mass and stiffness and displacement of all nodes for the different modes, yielding the full $[\mathbf{\Phi}]$-matrix. The resulting $[\mathbf{\Phi}]$-matrix is referred to the entire set of DOFs in the Abaqus model. To establish the damping of the structure, *Rayleigh damping* is used. Rayleigh damping is briefly explained in Section 2.2.3.

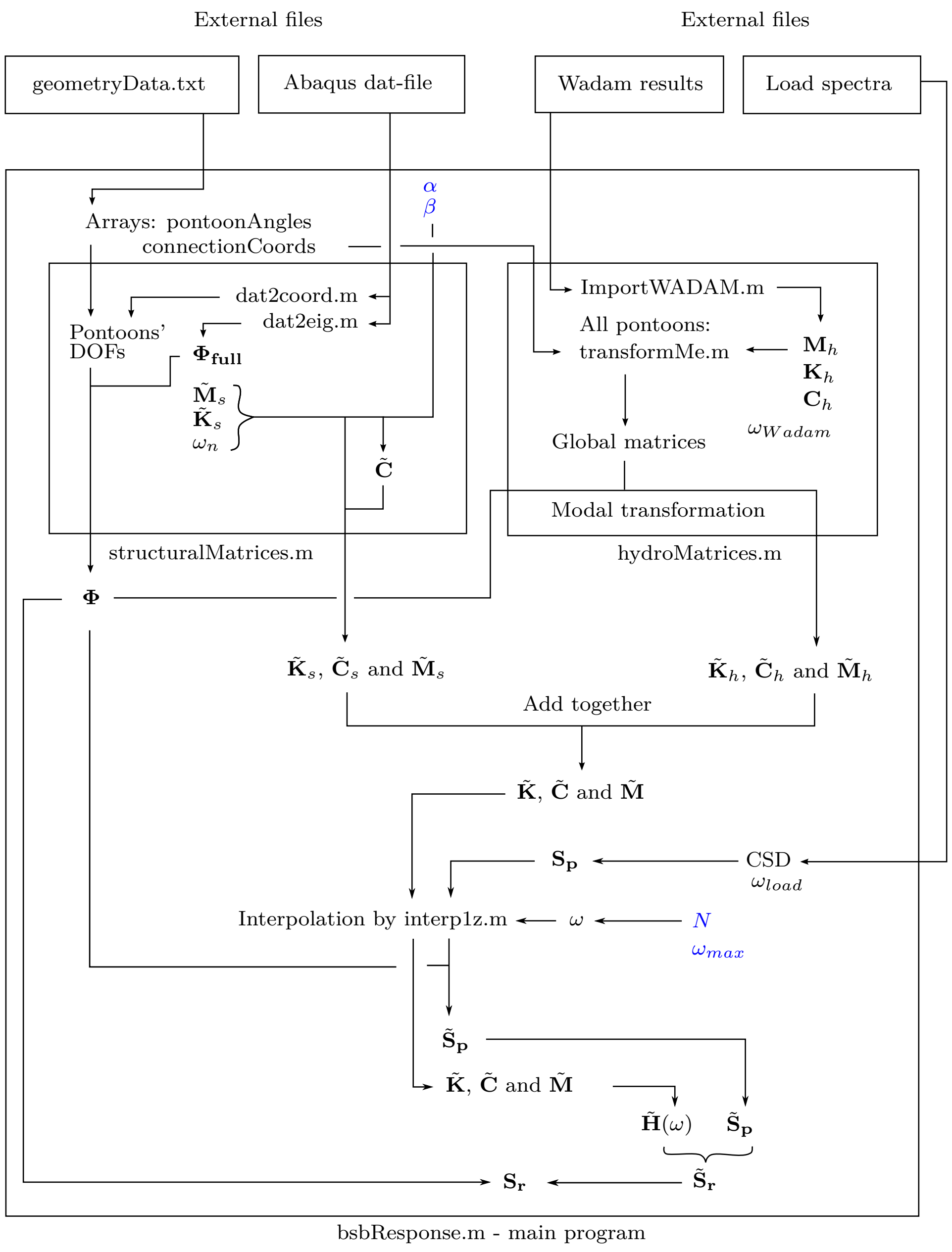


Figure 3.1: *Structure of the Matlab program. The parameters coloured blue, are defined on top of the program code in* `floatingBridge.m`.

3.2 Pontoons

The structure is assumed floating on discretely distributed pontoons, connected to the frame at specified nodes. To add the contributions from the pontoons to the modal mass, damping and stiffness from the structural part of the model, a modal transformation is performed. The modal transformation matrix is based on the [$\mathbf{\Phi}$]-matrix established from the data file from the ABAQUS job and the `structuralMatrices.m` subroutine, as discussed in Section 5.1. Because load and additional stiffness, mass and damping are assumed to apply only in the nodes of the pontoons, the transformation matrix can be drastically simplified. This is discussed in Section 2.6.6, for the mass contribution of the pontoons to the system. The global modal system matrices are found using the MATLAB function `hydroMatrices.m`, which is shown in Appendix B.3, and required DNV HYDROD WADAM analyses.

3.2.1 DNV HydroD Wadam

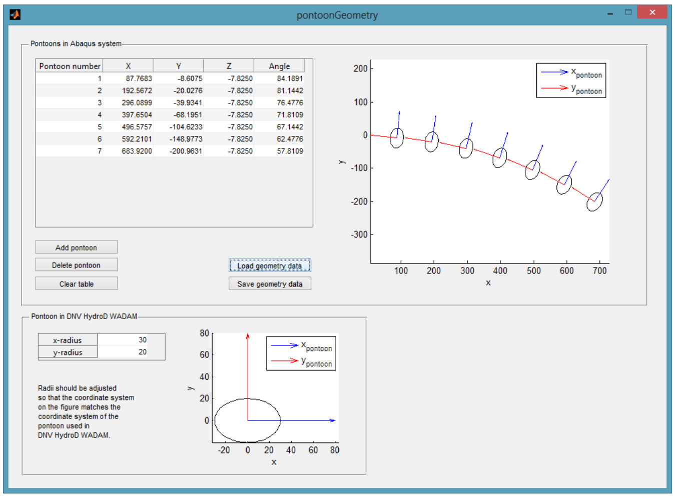
DNV HYDROD WADAM utilizes the panel method, three-dimensional radiation-diffraction theory and Airy wave theory [18], and is able to determine important properties for use with bodies submerged in fluids. The theory behind panel methods is briefly described in Appendix A.3.

To determine the contributions from each pontoon to the overall system properties, analyses from DNV HydroD Wadam are required. In these analyses, frequency dependent mass, stiffness and damping matrices will be determined for one submerged pontoon. By importing the resulting data files from the analyses of one pontoon, both frequency dependent and frequency independent contributions to mass, stiffness and damping from each pontoon are established.

Two different types of pontoons are supported at the same time in the main program.

3.2.2 Establishing the pontoon arrangement

To establish a working interface between the dry and wet domains, the coordinates where the pontoons are connected with the structure are needed. These are found by quering the nodes where the pontoons are attached in ABAQUS. Also, the angles between the local pontoon DOFs and the global coordinate system should be determined. To make sure this is done correctly, a supporting software, with an easy-to-use graphical user interface (GUI), was made. The MATLAB file is found in Appendix B.9, and the resulting GUI is shown in Figure 3.2. In this software, the global pontoon arrangement is shown, with local DOFs of each pontoon. The coordinates and angles of each pontoon are added in a table. This table can be both exported to and imported from a geometry data file. This geometry data file is thereafter used as input for the main calculations.



Pontoon number	X	Y	Z	Angle
1	87.7683	-8.6075	-7.8250	84.1891
2	192.5672	-20.0276	-7.8250	81.1442
3	296.0899	-39.9341	-7.8250	76.4776
4	397.6504	-68.1951	-7.8250	71.8109
5	496.5757	-104.6233	-7.8250	67.1442
6	592.2101	-148.9773	-7.8250	62.4776
7	683.9200	-200.9631	-7.8250	57.8109

x-radius	30
y-radius	20

Radii should be adjusted so that the coordinate system on the figure matches the coordinate system of the pontoon used in DNV HydroD WADAM.

Figure 3.2: *Supporting software used to establish the correct arrangement of the pontoons. The pontoon arrangement shown here is from the Bersøysund Bridge, discussed in Chapter 5.*

3.2.3 Transformation to modal coordinates

After the load spectra were oriented correctly and stacked the way shown in Equation 3.25, a transformation to modal coordinates is made. The procedure is derived in Section 2.6.5. Equation 2.147 reads out

$$\left[\tilde{\mathbf{S}}_\mathbf{p}\right] = [\mathbf{\Phi}]^T \left[\mathbf{S}_\mathbf{p}\right] [\mathbf{\Phi}]$$

Here, the modal transformation matrix refers to the simplified version of the full transformation matrix. This is discussed in Section 2.6.6.

3.2.4 Wet contributions included in Abaqus model

Ideally, all mass, damping and stiffness from the wet domain would have been added directly in Abaqus. When adding some of this after the modal properties are found, the mode shapes will no longer be exact. This is expected to converge towards the exact solution regardless of this, when enough modes are included. As far as the author knows at the present, Abaqus does not support frequency dependent mass, stiffness and damping to be added. The contributions independent of frequency, on the other hand, can fairly simply be added to the model. Which of the matrices from the pontoons to add after the Abaqus analysis, are defined before calculations are run in the Matlab program, to amend for these different possibilities.

3.3 Important operations

3.3.1 Transformation of load spectra

To support the case where the DOFs of the load spectra refer to the local DOFs of each pontoon, a transformation of this was implemented as an optional feature. Here, the local DOFs are assumed placed in the middle of each pontoon, with axes corresponding to the local axes of the pontoon. The local DOFs are assumed both rotated in the xy-plane and translated along the z-axis, compared to the global ones. To be able to use this load spectra matrix with the established system matrices, a transformation incorporating both these effects has to be done. The DOFs corresponding to a pontoon are denoted

$$\{\mathbf{u}_i\} = \begin{Bmatrix} u_{1,i} \\ u_{2,i} \\ u_{3,i} \\ u_{4,i} \\ u_{5,i} \\ u_{6,i} \end{Bmatrix} \tag{3.1}$$

The local DOFs are denoted $\{\bar{\mathbf{u}}_i\}$, which are both rotated and translated compared to $\{\mathbf{u}_i\}$. To ease the development of the transformation, a third temporary set of DOFs is introduced; $\{\hat{\mathbf{u}}_i\}$, which inhabits the same z-level as $\{\bar{\mathbf{u}}_i\}$ and the same rotation as $\{\mathbf{u}_i\}$. This is shown in Figure 3.3.

The cross-spectra between loading along the DOFs of pontoon i and j, $\{\mathbf{p}_i\}$ and $\{\mathbf{p}_j\}$ can, by imposing Equation 2.111, be written

$$[\mathbf{S}_{p,ij}] = \mathcal{F}\left(E\left[\{\mathbf{p}_i(t)\}\{\mathbf{p}_j(t+\tau)\}^T\right]\right) \tag{3.2}$$

The following relations between $\{\mathbf{u}_i\}$ and $\{\hat{\mathbf{u}}_i\}$ are found, using Figure 3.3:

$$\hat{u}_1 = u_1 - hu_5 \tag{3.3}$$
$$\hat{u}_2 = u_2 + hu_4 \tag{3.4}$$
$$\hat{u}_3 = u_3 \tag{3.5}$$
$$\hat{u}_4 = u_4 \tag{3.6}$$
$$\hat{u}_5 = u_5 \tag{3.7}$$
$$\hat{u}_6 = u_6 \tag{3.8}$$

where h is the translation in z-direction, or the length between the centre of gravity and the top of the pontoon. Alternatively, this can be written

$$\{\hat{\mathbf{u}}_i\} = [\mathbf{T}_{z,i}]\{\mathbf{u}_i\} \tag{3.9}$$

where

$$[\mathbf{T}_{z,i}] = \begin{bmatrix} 1 & 0 & 0 & 0 & -h & 0 \\ 0 & 1 & 0 & h & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

The angle between $\{\hat{\mathbf{u}}_i\}$ and $\{\bar{\mathbf{u}}_i\}$ is denoted α_i. This is also shown in Figure 3.3. The transformation between $\{\bar{\mathbf{u}}_i\}$ and $\{\hat{\mathbf{u}}_i\}$ is equivalent to that given in Equation 2.163. When using this, the following relations between $\{\bar{\mathbf{u}}_i\}$ and $\{\hat{\mathbf{u}}_i\}$ are found:

$$\begin{aligned} \bar{u}_1 &= cos(\alpha_i)\hat{u}_1 + sin(\alpha_i)\hat{u}_2 && (3.11)\\ \bar{u}_2 &= -sin(\alpha_i)\hat{u}_1 + cos(\alpha_i)\hat{u}_2 && (3.12)\\ \bar{u}_3 &= \hat{u}_3 && (3.13)\\ \bar{u}_4 &= cos(\alpha_i)\hat{u}_4 + sin(\alpha_i)\hat{u}_5 && (3.14)\\ \bar{u}_5 &= -sin(\alpha_i)\hat{u}_4 + cos(\alpha_i)\hat{u}_5 && (3.15)\\ \bar{u}_6 &= \hat{u}_6 && (3.16) \end{aligned}$$

On matrix form, this reads out

$$\{\bar{\mathbf{u}}_i\} = [\mathbf{T}_{\alpha,i}]\{\hat{\mathbf{u}}_i\} \tag{3.17}$$

where

$$[\mathbf{T}_{\alpha,i}] = \begin{bmatrix} cos(\alpha_i) & sin(\alpha_i) & 0 & 0 & 0 & 0 \\ -sin(\alpha_i) & cos(\alpha_i) & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos(\alpha_i) & sin(\alpha_i) & 0 \\ 0 & 0 & 0 & -sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

Combining Equations 3.3-3.8 with Equations 3.11-3.16, the following relation is obtained:

$$\{\mathbf{u}_i\} = [\mathbf{T}_i]\{\bar{\mathbf{u}}_i\} \tag{3.19}$$

where

$$[\mathbf{T}_i] = \begin{bmatrix} cos(\alpha_i) & sin(\alpha_i) & 0 & h\cdot sin(\alpha_i) & -h\cdot cos(\alpha_i) & 0 \\ -sin(\alpha_i) & cos(\alpha_i) & 0 & h\cdot cos(\alpha_i) & h\cdot sin(\alpha_i) & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cos(\alpha_i) & sin(\alpha_i) & 0 \\ 0 & 0 & 0 & -sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

The same result is achieved by multiplying the first transformation matrix in Equation 3.10 with the second transformation matrix in Equation 3.18; $[\mathbf{T}_i] = [\mathbf{T}_{z,i}][\mathbf{T}_{\alpha,i}]$. Equation 2.168 gives the following transformation of the load vectors referring to the sets of DOFs:

$$\{\mathbf{p}_i\} = [\mathbf{T}_i]^T \{\bar{\mathbf{p}}_i\} \tag{3.21}$$

Together with Equation 3.2 this gives

$$[\mathbf{S}_{p,ij}] = \mathcal{F}\left(E\left[[\mathbf{T}_i]^T \{\bar{\mathbf{p}}_i(t)\}\{\bar{\mathbf{p}}_j(t+\tau)\}^T [\mathbf{T}_j]\right]\right) \tag{3.22}$$

Because the transformation matrix is a constant matrix independent of time, this can be rewritten

$$[\mathbf{S}_{p,ij}] = [\mathbf{T}_i]^T \mathcal{F}\left(E\left[\{\bar{\mathbf{p}}_i(t)\}\{\bar{\mathbf{p}}_j(t+\tau)\}^T\right]\right)[\mathbf{T}_j] \tag{3.23}$$

Recognizing the Fourier transform as the covariance spectra between the loads along the local DOFs, this can again be rewritten

$$[\mathbf{S}_{p,ij}] = [\mathbf{T}_i]^T \left[\bar{\mathbf{S}}_{p,ij}\right][\mathbf{T}_j] \tag{3.24}$$

The total load spectra can thus be expressed as

$$[\mathbf{S}_\mathbf{p}] = \begin{bmatrix} [\mathbf{T}_1]^T\left[\bar{\mathbf{S}}_{p,11}\right][\mathbf{T}_1] & [\mathbf{T}_1]^T\left[\bar{\mathbf{S}}_{p,12}\right][\mathbf{T}_2] & \cdots & [\mathbf{T}_1]^T\left[\bar{\mathbf{S}}_{p,1n}\right][\mathbf{T}_n] \\ [\mathbf{T}_2]^T\left[\bar{\mathbf{S}}_{p,21}\right][\mathbf{T}_1] & [\mathbf{T}_2]^T\left[\bar{\mathbf{S}}_{p,22}\right][\mathbf{T}_2] & \cdots & [\mathbf{T}_2]^T\left[\bar{\mathbf{S}}_{p,2n}\right][\mathbf{T}_n] \\ \vdots & \vdots & \ddots & \vdots \\ [\mathbf{T}_n]^T\left[\bar{\mathbf{S}}_{p,n1}\right][\mathbf{T}_1] & [\mathbf{T}_n]^T\left[\bar{\mathbf{S}}_{p,n2}\right][\mathbf{T}_2] & \cdots & [\mathbf{T}_n]^T\left[\bar{\mathbf{S}}_{p,nn}\right][\mathbf{T}_n] \end{bmatrix} \tag{3.25}$$

where n is the total number of pontoons.

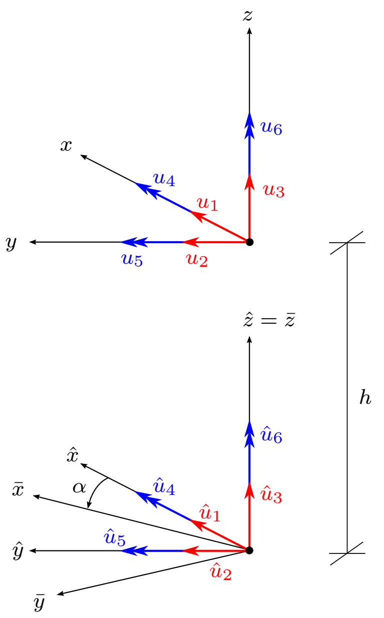


Figure 3.3: *Transformation of load spectra to global coordinates. Arrows coloured blue and with double arrowheads indicate rotation about the indicated axes.*

3.3.2 Modal response spectra

The modal frequency response function becomes

$$\left[\tilde{\mathbf{H}}(\omega)\right] = \left[[\tilde{\mathbf{K}}] + i\omega[\tilde{\mathbf{C}}] - \omega^2[\tilde{\mathbf{M}}]\right]^{-1} \tag{3.26}$$

where $[\tilde{\mathbf{K}}]$, $[\tilde{\mathbf{C}}]$ and $[\tilde{\mathbf{M}}]$ are the total modal stiffness, damping and mass matrices, respectively. The modal response spectra are then, as described in Section 2.6, found in the following manner:

$$[\mathbf{S_y}] = \left[\tilde{\mathbf{H}}(\omega)\right]\left[\tilde{\mathbf{S}_p}\right]\left[\tilde{\mathbf{H}}(\omega)\right]^* \tag{3.27}$$

where $\left[\tilde{\mathbf{H}}(\omega)\right]^*$ is the transposed complex conjugate of $\left[\tilde{\mathbf{H}}(\omega)\right]$.

3.3.3 Transformation to physical coordinates

The transformation of response spectra from modal DOFs to physical DOFs is explained in Section 2.6.4. Equation 2.141 gives the following physical response variance spectra:

$$[\mathbf{S_u}] = [\mathbf{\Phi}]\,[\mathbf{S_y}]\,[\mathbf{\Phi}]^T$$

```
Do for all n from 1 to 2N:
  1. Set $\omega = \omega_0$, as the $0^{th}$ iteration. $\omega_0 = 0$ as standard in iterateFreq.m.
  2. Solve the eigenvalue problem in Equation 3.28, with the chosen $\omega$, using
     eigSolve.m.
  3. Sort all the eigenvalues, and the eigenvectors correspondingly.
  4. Set $\omega$ equal the absolute value of the $n^{th}$ resulting eigenvalue.
  5. If the number of iterations has exceeded a specified value of maximum iterations
     itmax, or the previously found $\omega$ deviates with less than a specified tolerance
     tol from the new $\omega$: set $\lambda_n$ equal the $n^{th}$ resulting eigenvalue, and $q_n$ equal
     the $n^{th}$ resulting eigenvector - and break iteration loop. If neither is true, do
     steps 2-5 over.
```

Figure 3.4: *Iteration procedure for frequency dependent eigenvalue problem.*

3.4 Eigenfrequencies and modal damping coefficients

The eigenfrequencies and eigenvectors exported from the structural model in Abaqus are used for the analysis. The correct values of these will depend on the total mass, damping and stiffness, and not only the structural ones. Therefore, the natural frequencies and the damping ratios for the modes used will deviate from that of the actual modes. Using the theory presented in Section 2.3, these values are recalculated from the modal matrices. Since the contributions from the pontoons results in non-diagonal modal matrices, the eigenvalues calculated will be complex.

The modal equation of motion for free vibration of the entire system can be written

$$[\tilde{\mathbf{M}}]\{\ddot{\mathbf{y}}\} + [\tilde{\mathbf{C}}]\{\dot{\mathbf{y}}\} + [\tilde{\mathbf{K}}]\{\mathbf{y}\} = \{\mathbf{0}\} \tag{3.28}$$

Because the mass and damping are frequency dependent, this represents a non-linear eigenvalue problem. Iteration is therefore needed, which is implemented in the appended Matlab function `iterateFreq.m` (see Appendix B.6). Here, [**M**], [**C**] and [**K**] are input as 3D arrays, with dimensions $NxNxL$, where N is the total number of modes and L is the length of the frequency vector. The procedure for the iteration is given in Figure 3.4.

Equation 2.59 is used to solve the eigenvalue problem. Here, this reads out

$$\{\mathbf{y}\} = [\mathbf{\Psi}]\{\mathbf{g}\} \tag{3.29}$$

where $\{\mathbf{y}\}$ is the vector containing the modal coordinates corresponding to the modes found from the frame only, and $\{\mathbf{g}\}$ represents the coordinates after a second modal transformation from $\{\mathbf{y}\}$.

The original modal decomposition is written

$$\{\mathbf{u}\} = [\mathbf{\Phi}]\{\mathbf{y}\} \tag{3.30}$$

Together, this yield

$$\{\mathbf{u}\} = [\mathbf{\Phi}][\mathbf{\Psi}]\{\mathbf{g}\} \tag{3.31}$$

The *new* mode shapes reveal how much of each *original* mode shape is activated in each *new* mode. The total modal transformation matrix is therefore $[\mathbf{\Phi}][\mathbf{\Psi}]$. From Equations 2.46 and 2.47, the values of ω and ξ for each mode are calculated, and output from the main program. The corresponding mode shapes are also output, in the form of the modal transformation matrices.

3.5 Versatility

The Matlab program was developed with the Bersøysund Bridge especially in mind, but will work on any floating structure with discretely distributed pontoons. The program needs an Abaqus data file from a frequency analysis, based on a model of an arbitrary structure with floating elements connected in given nodes, and corresponding DNV HydroD Wadam analyses of the pontoons used.

Also, to be able to either exclude the dry or the wet domains totally, two boolean parameters are defined in the program:

```
includeWet = 1/0
includeDry = 1/0
```

It is also possible to use white noise loading, instead of inputting load spectra. This implies that the diagonal terms of the cross-spectra of the load are set to an equal and constant value, defined in the program. The concept of white noise is briefly introduced in Section 2.5.3.

4 Case: Simply supported beam

Using the MATLAB program described in Chapter 3, two simplified cases were studied. First a simply supported beam, with no hydrodynamic action was studied. Thereafter, a single pontoon was assumed placed on the midspan of the same beam. The results are compared and discussed successively in the following chapter.

4.1 Simply supported beam

A simply supported beam was modelled in ABAQUS/CAE, with geometry as shown in Figure 4.1. The total length of the beam was $L = 400m$, and the midpoint of the beam at $x = 200$m. 50 modes were included in the ABAQUS calculations. Stiffness, damping and mass data for the model are found in Table 4.1. No pontoons were included in this case. The simulation was run with Rayleigh damping as defined in Section 2.2.3, with $\alpha = 5 \cdot 10^{-2} s^{-1}$ and $\beta = 3 \cdot 10^{-2} s^{-2}$. A white noise loading, corresponding to a unity constant value of the load spectrum, was used for all diagonal (auto) elements in the load spectra matrix. In other words, the beam was loaded with white noise along all 6 DOFs in the midpoint.

4.1.1 Results and discussion

The response spectra for the translational DOFs, resulting from white noise on the simply supported beam, are shown in Figure 4.2. It is noted that one sharp peak is found in both the two transversal (z- and y-direction) response spectra. As expected, the response spectra for the two transversal DOFs are equal. This comes from the fact that the beam

Table 4.1: *Stiffness and mass data for the simply supported beam studied.*

	Moments of inertia about COG				
Mass $[kg]$	$I_{xy}[m^4]$	$I_{xz}[m^4]$	$I_{zz}[m^4]$	Cross-section area $[m^2]$	E $[GPa]$
$3.91 \cdot 10^5$	$1.56 \cdot 10^6$	$5.21 \cdot 10^9$	$5.21 \cdot 10^9$	37.7	210

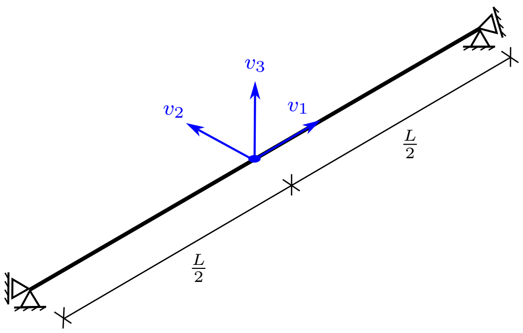


Figure 4.1: *Simply supported beam. The DOFs illustrated are the translational DOFs. Rotational DOFs 4, 5 and 6 are defined positive from the right hand rule, about the illustrated DOFs 1, 2 and 3 respectively.*

is axially symmetric, and nothing distinguish the properties along these DOFs. No action appears to happen along the axial DOF, which makes sense from the fact that the axial stiffness is much higher than the bending stiffness.

Modal convergence

The covariance spectra of the simply supported beam were calculated with different number of total modes included, to study the effects this had on the solution. Selected results are shown in Figure 4.3.

From Figures 4.3(a) and 4.3(c), the number of modes included is seen to impact the solution drastically for the cross-spectrum response. Figures 4.3(b) and 4.3(d) indicate a much smaller dependence of number of modes used for the spectral densities of the auto-covariance. For the auto-spectra, 10 modes appear to be sufficient, while for cross-spectra, results have not yet converged with 50 modes included. This also outlines the case for the elements in the response spectra matrix not shown here. On the other hand; if the orders of the response spectra are taken into account, the significant part of the response converge fast. By looking at Figure 4.2, it is noted that the response in the transversal directions is the significant one. The convergences of the response spectra corresponding to these DOFs are very fast.

It is noted that the structural parameters for mode no. 47 were badly scaled compared to the rest of the modes for this model. This is shown in Table 4.2. Attempts to exclude this particular mode were made, without any observed change in the results. Still, caution should be made with this particular issue, to avoid badly scaled matrices before matrix inversions are performed. Even though the mode itself does not contribute appreciable to the overall solution, it can cause numerical instability for the entire system.

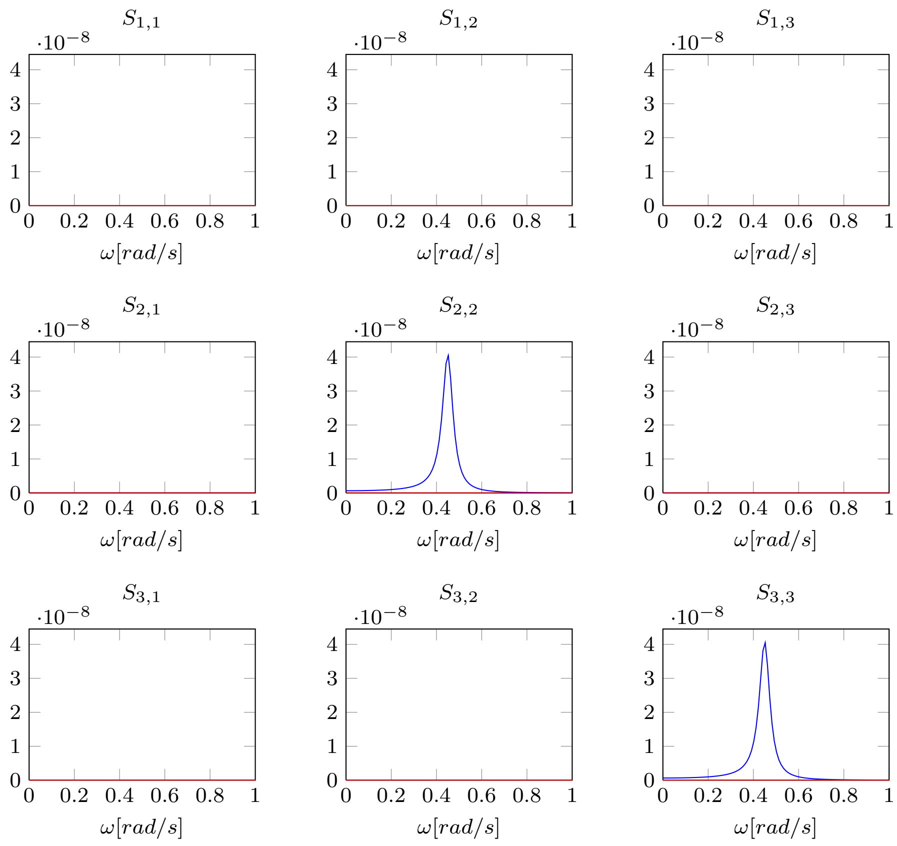


Figure 4.2: *Response spectra for translational DOFs of the simply supported beam without any pontoons, for the case of white noise load spectra. Blue plots shows real parts while red shows imaginary parts of the solution.*

Table 4.2: *Structural parameters for selected modes of the simply supported beam.*

Mode no. (n)	$\tilde{K}_n$	$\tilde{C}_n$	$\tilde{M}_n$
46	$1.57 \cdot 10^{10}$	$4.70 \cdot 10^{8}$	$1.92 \cdot 10^{5}$
47	$3.78 \cdot 10^{32}$	$1.13 \cdot 10^{31}$	$3.51 \cdot 10^{22}$
48	$2.50 \cdot 10^{10}$	$7.51 \cdot 10^{8}$	$1.82 \cdot 10^{5}$

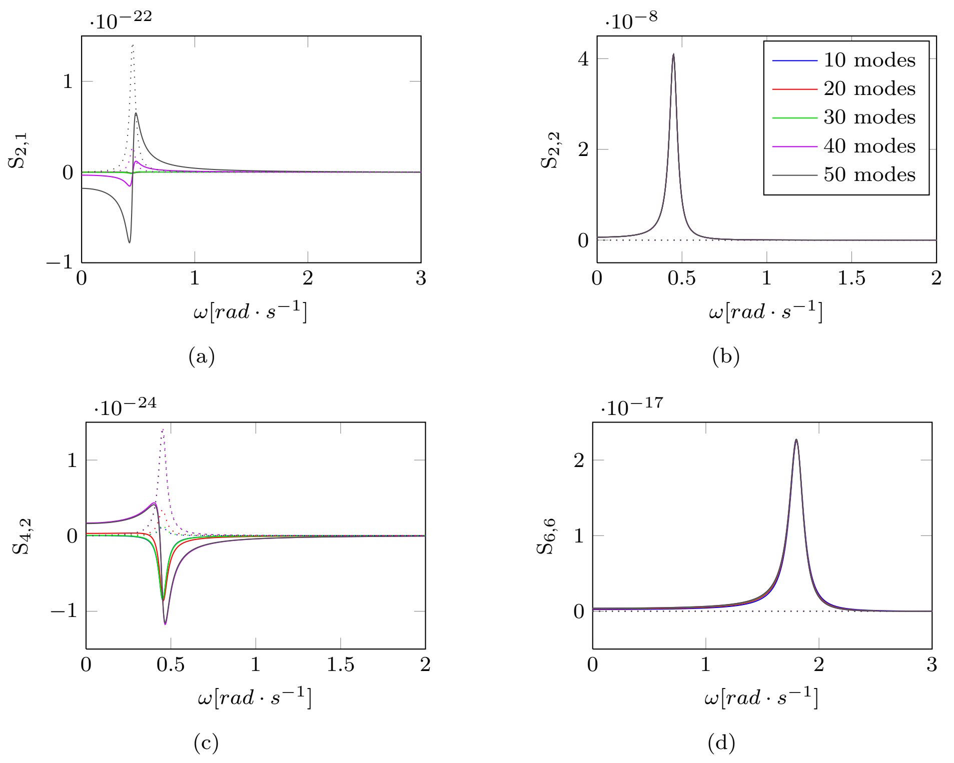


Figure 4.3: *Selected cross- and auto-covariance spectra for the simply supported beam, with different number of modes included in the calculations. Dotted lines represent imaginary parts of the solutions.*

4.2 Simply supported beam with pontoon on midspan

One pontoon (the type used on pontoon number 1 and 7 on the Bergsøysund bridge) was then assumed on the midspan of the simply supported beam. This set-up is illustrated in Figure 4.4. The same structural model as described in Section 4.1 was used, together with this single pontoon. 50 modes were included in the Abaqus calculations. The geometry data for the pontoon arrangement, which is used as input in the Matlab program, is shown in Table 4.3. Load and damping were chosen equal to that used with the simply supported beam without any pontoons, discussed in the previous section.

4.2.1 Results and discussion

Response from white noise

The response spectra for the translational DOFs from white noise on the simply supported beam are shown in Figure 4.5. Also in this case, sharp peaks are found in the solution, which indicate resonant frequencies. Significant translational response is found only along the transversal DOF denoted v_2 in Figure 4.4, *i.e.* the transversal DOF tangent to the water surface.

Modal convergence

Figure 4.6 shows part of the modal convergence study performed on the system. The auto-covariance spectra experience faster convergence than the cross-covariance spectra in this case as well. Also for this system, the solution converges faster when only the significant spectra are considered. The effect is especially prominent in Figure 4.6(c). This hypothesis is supported in Figure 4.7, which shows the variances of the displacements of

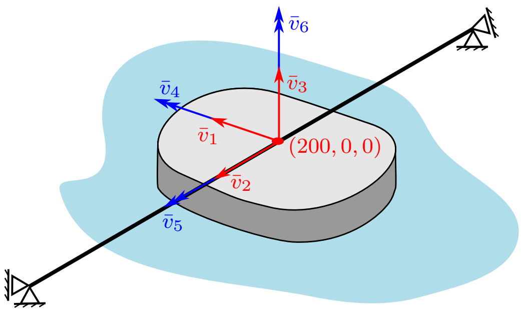


Figure 4.4: *Simply supported beam with pontoon on midspan. The DOFs shown are related to the local coordinate system of the pontoon. By comparison with Figure 4.1, it is noted that the pontoon is rotated* 90°.

Table 4.3: *Geometry data file for pontoon on midspan of simply supported beam.*

	Coordinates			
Pontoon no.	x [m]	y [m]	z [m]	Angle [°]
1	200	0	0	90

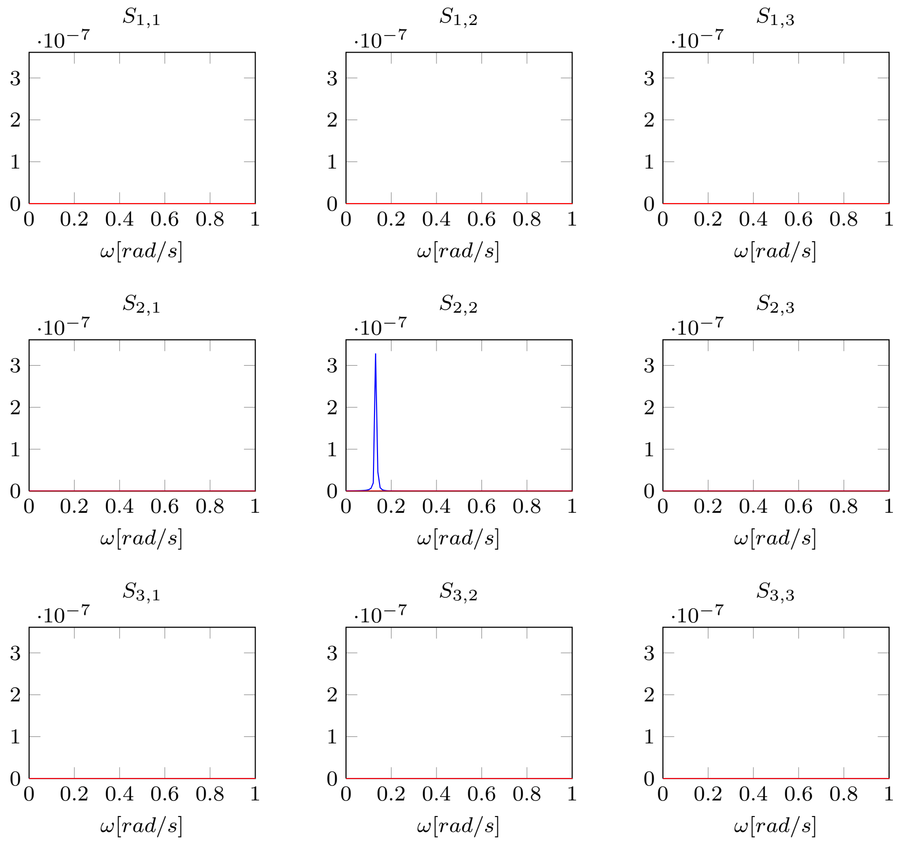


Figure 4.5: *Response spectra for translational DOFs of the simply supported beam with a pontoon on the midspan, for the case of white noise load spectra. Blue plots shows real parts while red shows imaginary parts of the solution.*

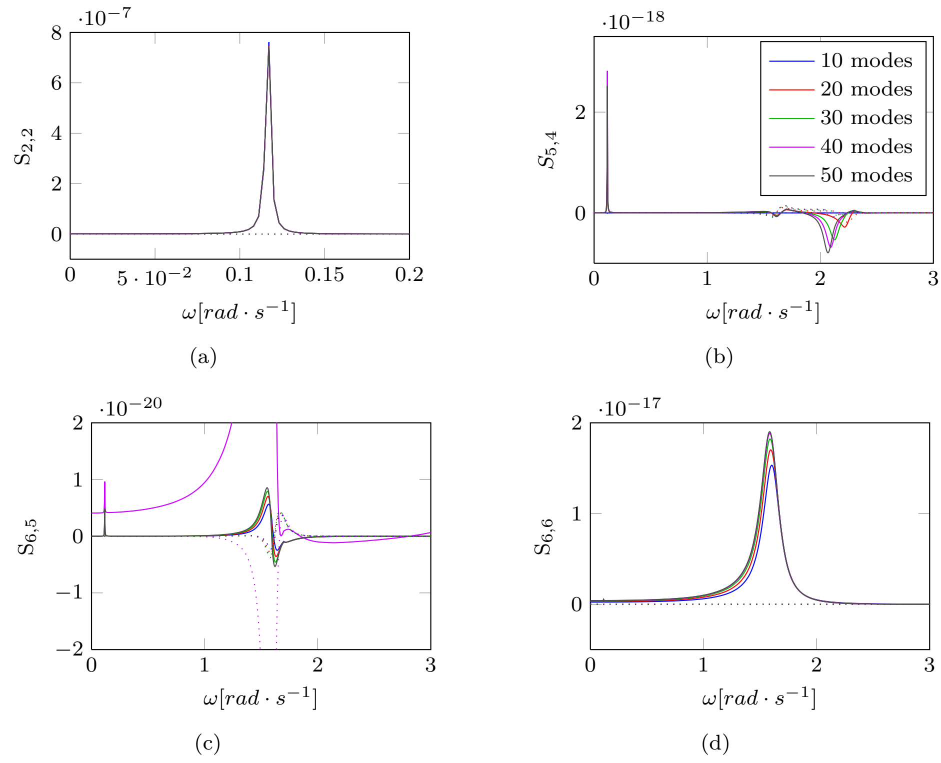


Figure 4.6: *Selected cross- and auto-covariance spectra for the simply supported beam with an pontoon placed at midspan, with different number of modes included in the calculations. Dotted lines represent imaginary parts of the solutions.*

the different DOFs. The variances of the auto-spectra terms were found by integrating the spectral densities, as given in Equation 2.101.

By disregarding the low-valued elements of the response spectra matrix, it is concluded that ~ 10 modes are sufficient for this kind of problem.

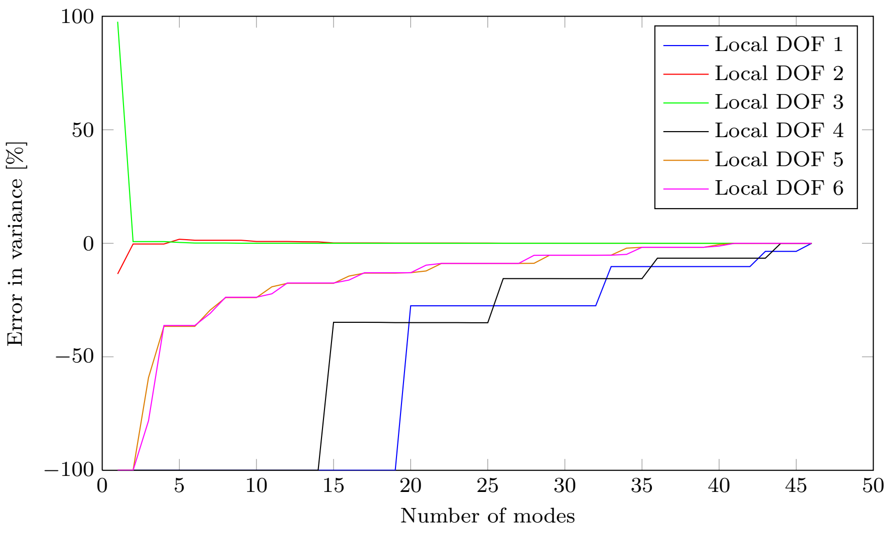


Figure 4.7: *Response variances of the simply supported beam with one pontoon, as deviance from the variance with all 50 included modes.*

5 Case: The Bersøysund Bridge

The Bersøysund Bridge in Møre og Romsdal county crosses the strait Bergsøysund between Bergsøy and Aspøy, and is part of the European route E39. The bridge is 931 meters long, and consists of 7 floating pontoons, with a maximum distance of 106 meters between them [4]. Its floating design makes it interesting for benchmarking of numerical procedures for calculation of floating structures. In particular, this is relevant for use with the possible floating bridges, as part of the Norwegian government's vision of eliminating all ferries along the new Coastal Highway Route E39.

The MATLAB program was used to perform dynamic response calculations on the Bersøysund Bridge, in the frequency domain. Both white noise and an estimated wave load were used as input for the load spectra. First, introductory information about the model setup is given. At the end, the results, and discussions regarding these, are presented.

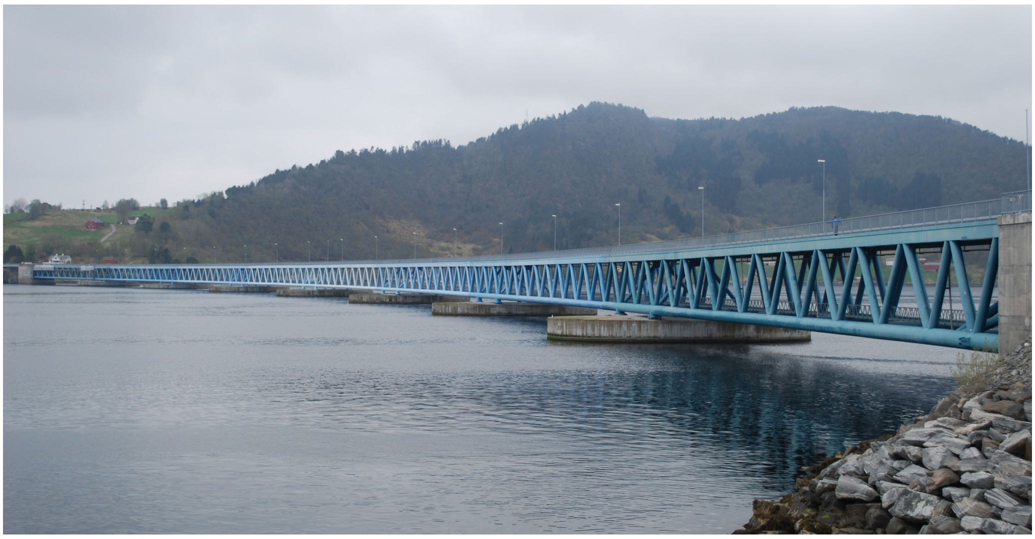
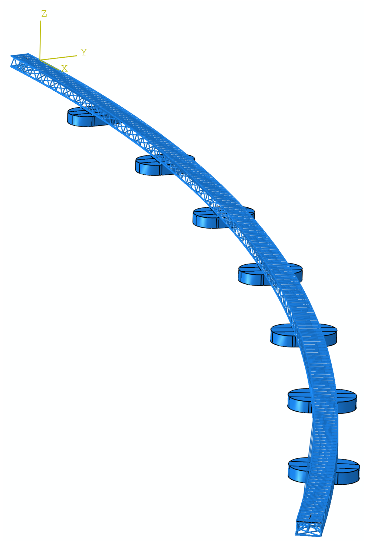
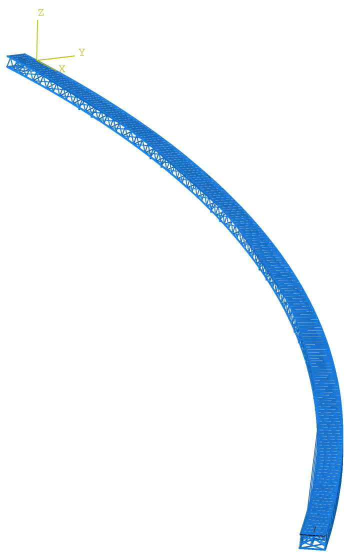


Figure 5.1: *The Bersøysund Bridge. Photo: Cato Edvardsen.*



(a) Entire model, with pontoons.



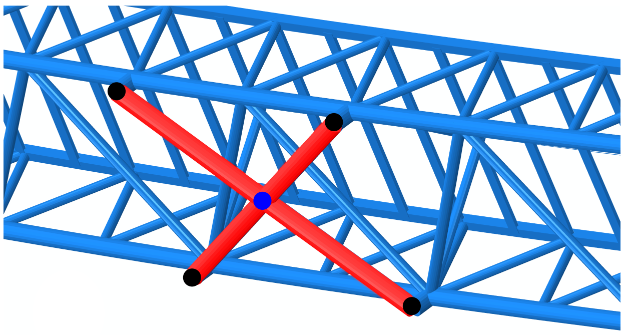
(b) Only bridge truss. This was used to find the structural properties from Abaqus.

Figure 5.2: *Model used for calculations of structural properties in* ABAQUS/CAE*.*

5.1 Frame

An ABAQUS/CAE model truss model of the Bersøysund Bridge by Sindre M. Hermstad was used. This model is shown in Figure 5.2(b). The origin of the global coordinate system is indicated on both Figures 5.2 and 5.4. As part of this model, a nearly rigid cross was placed where the pontoons and the structure were connected. This cross is shown in Figure 5.3. This was done because all the additional stiffness, mass and damping from the analyses in DNV HYDROD WADAM were defined with DOFs in the middle of the top surface of the pontoon. If this kind of adjustment of the FEA model had been unwanted, a transformation between the four points and the single point in the middle, similar to that shown in Appendix A.2, could have been performed. In the calculations of the response of the Bersøysund Bridge, the damping was defined using Rayleigh damping, with $\alpha = 5 \cdot 10^{-2} s^{-1}$ and $\beta = 3 \cdot 10^{-2} s^{-2}$.

The constant additions of mass and stiffness from the pontoons, were added directly into ABAQUS/CAE, and acted therefore as part of the frame. The different possibilities regarding this are discussed in Section 3.2.4.



● Real connection

● Model connection

Figure 5.3: *Detail below bridge, at pontoon connection. The simplified model used a stiff connection between the real four connection points, and gave one single connection point in the middle.*

5.2 Pontoons

The Bersøysund Bridge consists of 7 pontoons. These are placed and numbered according to Figure 5.4. The pontoons contribute to the system with the matrices shown in Table 2.1. These contributions are related to the 6 local DOFs of each pontoon, shown in Figure 5.5. As mentioned in the last section, $\left[\bar{\mathbf{K}}_{h,0}\right]$ and $\left[\bar{\mathbf{M}}_{h,0}\right]$ were included in the Abaqus/CAE model, and only the frequency dependent terms added afterwards.

DNV HydroD Wadam analyses performed by Hermstad were used for the calculations of the bridge. His analyses were based on a similar analysis performed by A. Suyuhthi, which is documented in [19]. Different ballasting are used on the 7 pontoons. This results in different values of the matrices shown in Table 2.1. Only 2 different pontoon types were assumed in the calculations: no. 1 and 7 denoted *type 1* and 2, 3, 4, 5 and 6 denoted *type 2*. This is also indicated in Figure 5.4. Therefore, two different analyses, for the different pontoons were used.

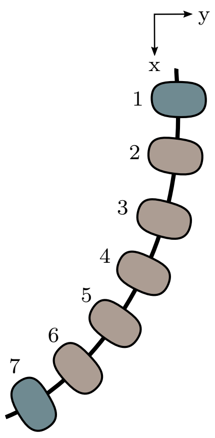


Figure 5.4: *Principal drawing of bridge from above. Pontoons 1 and 7 differ from pontoons 2 to 6.*

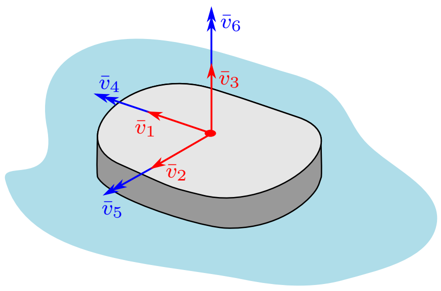


Figure 5.5: *DOFs of pontoon.*

Table 5.1: *Geometry data file for all pontoons of the Bersøysund Bridge.*

	Coordinates			
Pontoon	x [m]	y [m]	z [m]	Angle [°]
1	87.76833	-8.607546	-7.825	84.1891
2	192.567204	-20.027607	-7.825	81.1442
3	296.089925	-39.93409	-7.825	76.4776
4	397.650443	-68.195072	-7.825	71.8109
5	496.57571	-104.623267	-7.825	67.1442
6	592.210143	-148.977263	-7.825	62.4776
7	683.919968	-200.963124	-7.825	57.8109

5.3 Load cases

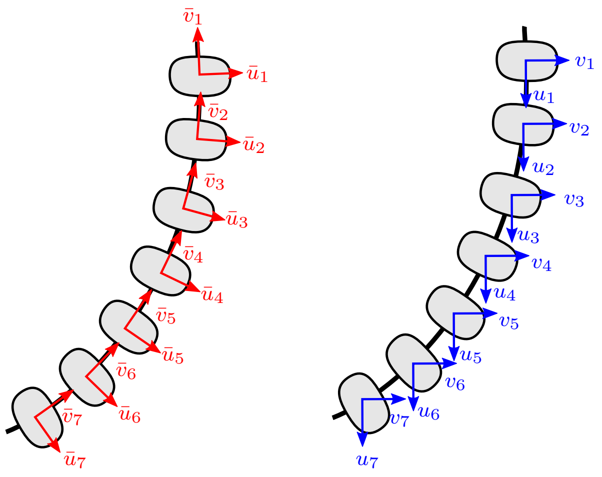
The response of the bridge was studied when exposed to two different load cases; (a) *white noise*, *i.e.* constant and equal values on the diagonal of the load cross-spectra matrix, and (b) *estimated load spectra matrix.*

5.3.1 White noise

White noise was used as loading by choosing a constant and equal value for all terms along the diagonal of the load spectra matrix. This kind of load is capable of revealing important characteristics of the system, since the structure is loaded equally in all frequencies. For normal loads, the spectra are zero outside a given range, which leads to a sort of *masking* of the system's full characteristic behaviour.

5.3.2 Estimated load spectra

A wave load spectra matrix estimated by Ragnar Sigbjörnsson was also used as input. This is based on the theory presented in Section 2.6.2. The load spectra used were referred to the local axes of the pontoons, and the transformation implemented in the program, explained in Section 3.3.1, had to be enforced. The orientations of the DOFs before and after this transformation are shown in Figure 5.6. Table 5.1 define the geometry data for pontoon arrangement of the Bersøysund Bridge, which was used as input for these transformations.



(a) DOFs of load spectra. (b) Global DOFs.

Figure 5.6: *Difference between coordinate systems for the load spectra and the Abaqus model. Blue arrows indicate global DOFs, while red indicate local DOFs of pontoons.*

5.4 Results and discussion

5.4.1 Eigenfrequencies, mode shapes and damping ratios

The eigenvalue problem from the total modal matrices, *i.e.* mass, damping and stiffness, was solved. This was done using the method described in Section 3.4. The calculated eigenfrequencies and damping ratios deviated from the eigenfrequencies given by the modal FEA of the structural model, as expected.

The differences between the eigenfrequencies corresponding to the same mode number, are shown in Figure 5.7. It is emphasized that equal mode numbers do not necessarily imply the same natural modes. From these figures, most eigenfrequencies of the entire system appear to be lower than those of the frame alone. Here, the frequency independent additions to stiffness and mass from the pontoons are regarded included in the frame. Thus, the frequency dependent contribution from the pontoons is responsible for this effect. Change in damping ratios for each mode number is shown in Figure 5.8. Modal damping ratios and natural frequencies for the first 10 modes, when conjugates are not counted, are given in Table 5.2.

The mode shapes corresponding to the damping ratios and eigenfrequencies are shown in Figures 5.9 and 5.10, seen from two different perspectives. These mode shapes are found using the method described in Section 3.4. It is noted that the plots shown only illustrate the translational normalized displacements from the modes. Also, only the real part of the effective modal transformation vectors are shown in this figure, *i.e.* the mode shape plots are found at $t = 0$. Assuming a mode shape is found from the normalized contribution to

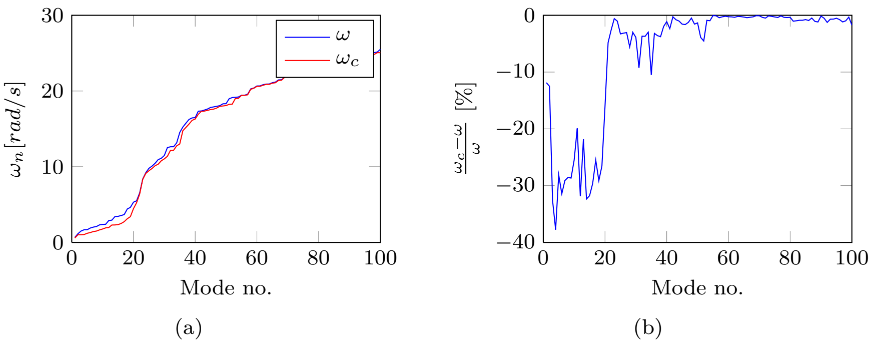


Figure 5.7: *Comparison between the eigenfrequencies obtained directly from Abaqus, and the ones calculated when including the contributions from the pontoons, for 100 included modes.* ω_c *denotes the eigenfrequency from the complex eigenvalue calculated using the procedure in Section 3.4 and* ω *the eigenfrequency resulting from* ABAQUS*.*

Table 5.2: *Natural frequencies and damping ratios of the Bersøysund Bridge for modes 1 to 10.*

Mode number, n	Natural frequency, ω_n [rad/s]	Damping ratio, ξ_n
1	0.5795	0.0399
2	1.0057	0.0413
3	1.0098	0.0415
4	1.0407	0.0427
5	1.2052	0.0441
6	1.3019	0.0445
7	1.4341	0.0472
8	1.4995	0.0507
9	1.6521	0.0521
10	1.7727	0.0528

displacement along each physical DOF, this will in fact change shape as time goes. This is a direct implication of the spreading effect described in Section 2.3.

In Figure 5.11, the mode shapes retrieved directly from ABAQUS, before any frequency dependent terms were added, are shown. Because these shapes are assumed and used as transformation basis, before the frequency dependent contributions are added, the mode shapes shown in Figures 5.9 and 5.10 are based on superposition of these. It is seen that the *original* mode shapes are cleaner in their appearance. This is reasonable, given the fact that the *new* mode shapes are based on superposition of the *original* mode shapes.

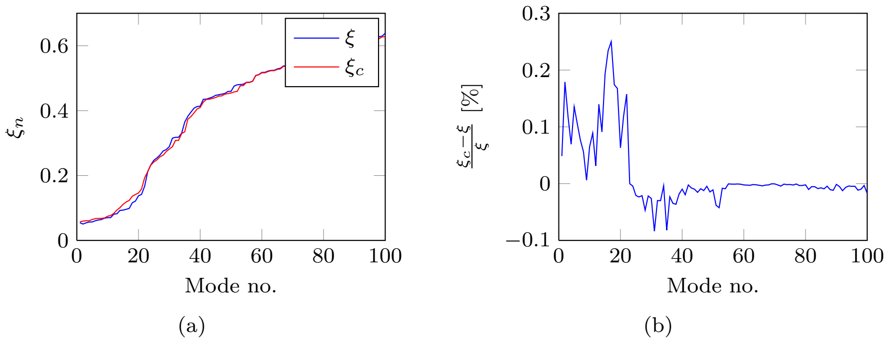


Figure 5.8: *Comparison between the modal damping ratios obtained using Rayleigh damping from the Abaqus structure, and the ones calculated when including the contributions from the pontoons, for 100 included modes. ξ_c denotes the damping ratio from the complex eigenvalue calculated using the procedure in Section 3.4 and ξ the modal damping ratio resulting from Rayleigh damping and modal mass and eigenvalues from* Abaqus.

5.4.2 Response from white noise

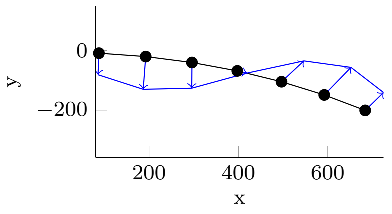
First, a unity white noise loading was enforced. Figure 5.12 shows the response spectra for the translational DOFs of pontoon no. 3 of the Bersøysund Bridge. Equation 2.102 was used to calculate coherence, which is seen in the elements above the diagonal in Figure 5.12. The response spectra for heave (DOF 3) of all pontoons are shown in Figure C.1.

As expected, the response spectra of the Bersøysund Bridge have much more complex appearance than the response spectra for the simply supported beam cases, shown in Figures 4.2 and 4.5. The number of peaks is also much larger, meaning that more modes are activated. By studying the response plots, it is seen that mode 1 is highly activated, with a natural frequency around $0.6rad/s$. Mode 2, 3 or 4, or combinations of these, are also seen to be very important, with frequencies around $1.0rad/s$. Most of these represent displacements in the XY-plane, as is seen in Figures 5.9 and 5.10. Mode 8, with natural frequency around $1.5rad/s$, is also seen to contribute significantly to the total response. This is mainly active out of the XY-plane, normal to the bridge deck.

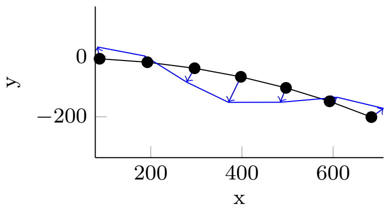
Modal convergence

Figure 5.13 shows selected response spectra from the white noise calculations, with different number of modes included. The results are considered near-converged when 20 modes are included. A slight difference from 20 and 100 included modes is seen. When considering the order of the solutions on the other hand, and emphasizing the auto-spectral densities, *e.g.* that shown in Figure 5.13(b), 20 modes will yield sufficiently accurate results.

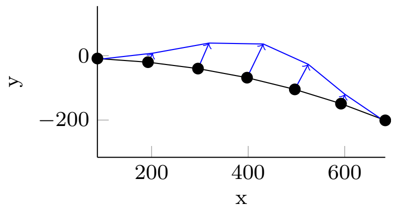
The variance of *heave* and *yaw* response, *i.e.* DOFs 3 and 6, are plotted in Figure 5.14 with respect to the number of modes included in the calculations. The convergence rate



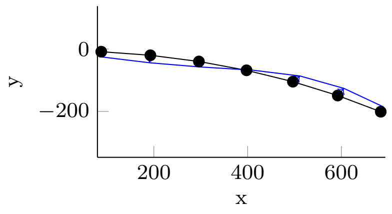
(a) Mode 1. $\omega_n = 0.5795 rad/s$.



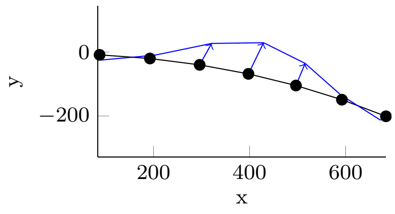
(b) Mode 2. $\omega_n = 1.0057 rad/s$.



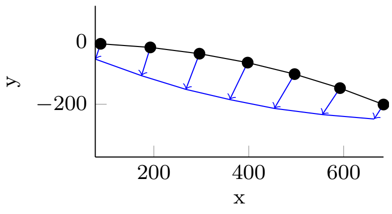
(c) Mode 3. $\omega_n = 1.0098 rad/s$.



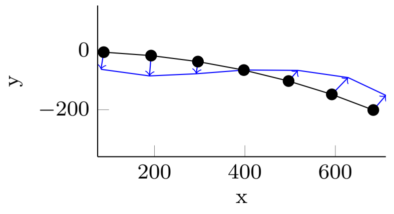
(d) Mode 4. $\omega_n = 1.0407 rad/s$.



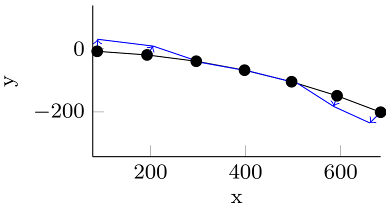
(e) Mode 5. $\omega_n = 1.2052 rad/s$.



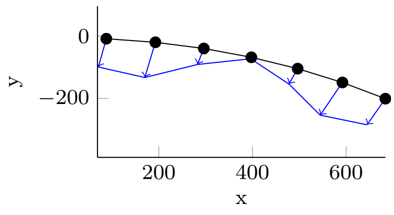
(f) Mode 6. $\omega_n = 1.3019 rad/s$.



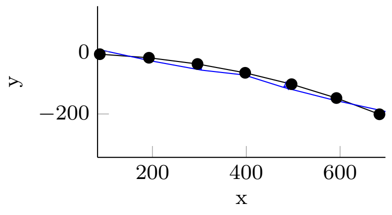
(g) Mode 7. $\omega_n = 1.4341 rad/s$.



(h) Mode 8. $\omega_n = 1.4995 rad/s$.

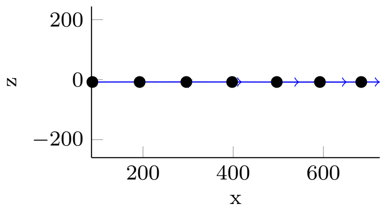


(i) Mode 9. $\omega_n = 1.6521 rad/s$.

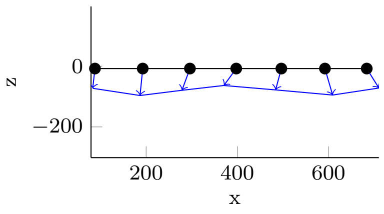


(j) Mode 10. $\omega_n = 1.7727 rad/s$.

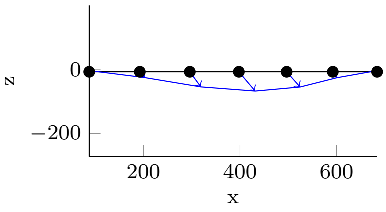
Figure 5.9: *Mode shapes from complex eigenvalue problem. The bridge is seen from above. Units are in meter, and the mode shapes are automatically scaled, based on all three dimensions of the mode vectors.*



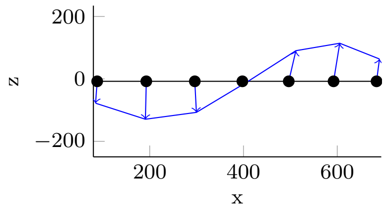
(a) Mode 1. $\omega_n = 0.5795 rad/s$.



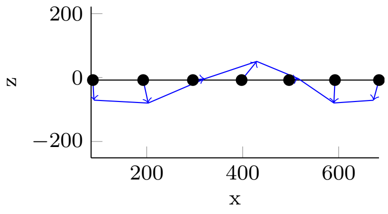
(b) Mode 2. $\omega_n = 1.0057 rad/s$.



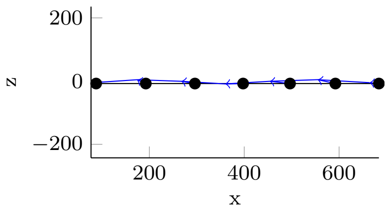
(c) Mode 3. $\omega_n = 1.0098 rad/s$.



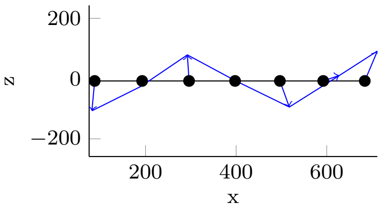
(d) Mode 4. $\omega_n = 1.0407 rad/s$.



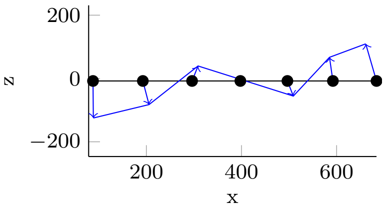
(e) Mode 5. $\omega_n = 1.2052 rad/s$.



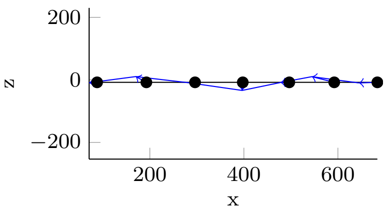
(f) Mode 6. $\omega_n = 1.3019 rad/s$.



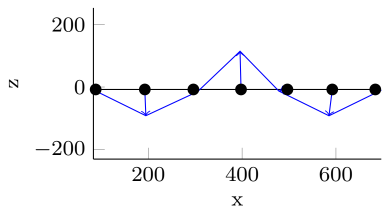
(g) Mode 7. $\omega_n = 1.4341 rad/s$.



(h) Mode 8. $\omega_n = 1.4995 rad/s$.

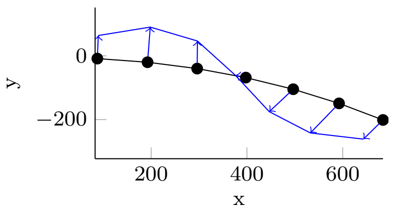


(i) Mode 9. $\omega_n = 1.6521 rad/s$.

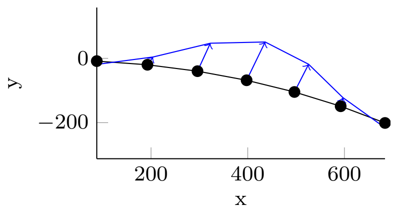


(j) Mode 10. $\omega_n = 1.7727 rad/s$.

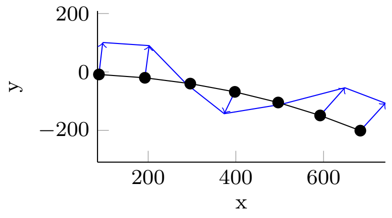
Figure 5.10: *Mode shapes from complex eigenvalue problem. The bridge is seen from the side. Units are in meter, and the mode shapes are automatically scaled, based on all three dimensions of the mode vectors.*



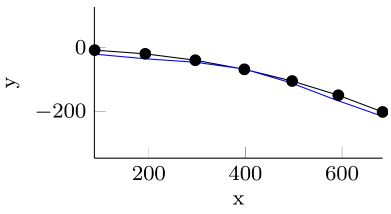
(a) Mode 1. $\omega_n = 0.6576 rad/s$.



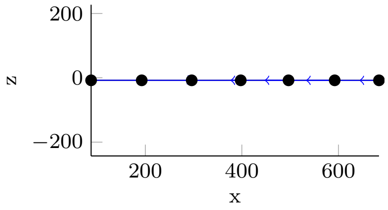
(b) Mode 2. $\omega_n = 1.1492 rad/s$.



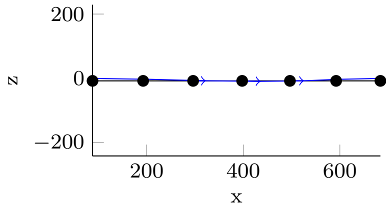
(c) Mode 3. $\omega_n = 1.4977 rad/s$.



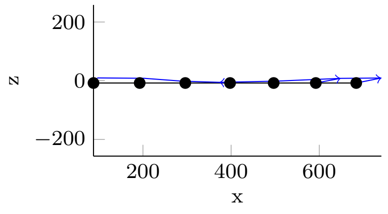
(d) Mode 4. $\omega_n = 1.6715 rad/s$.



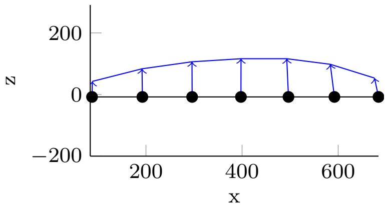
(e) Mode 1. $\omega_n = 0.6576 rad/s$.



(f) Mode 2. $\omega_n = 1.1492 rad/s$.



(g) Mode 3. $\omega_n = 1.4977 rad/s$.



(h) Mode 4. $\omega_n = 1.6715 rad/s$.

Figure 5.11: *Mode shapes from real eigenvalue problem, where no frequency dependent terms are added, seen from above (a)-(d) and the side (e)-(h). These mode shapes are plots of the original* ABAQUS *results. Units are in meter, and the mode shapes are automatically scaled, based on all three dimensions of the mode vectors.*

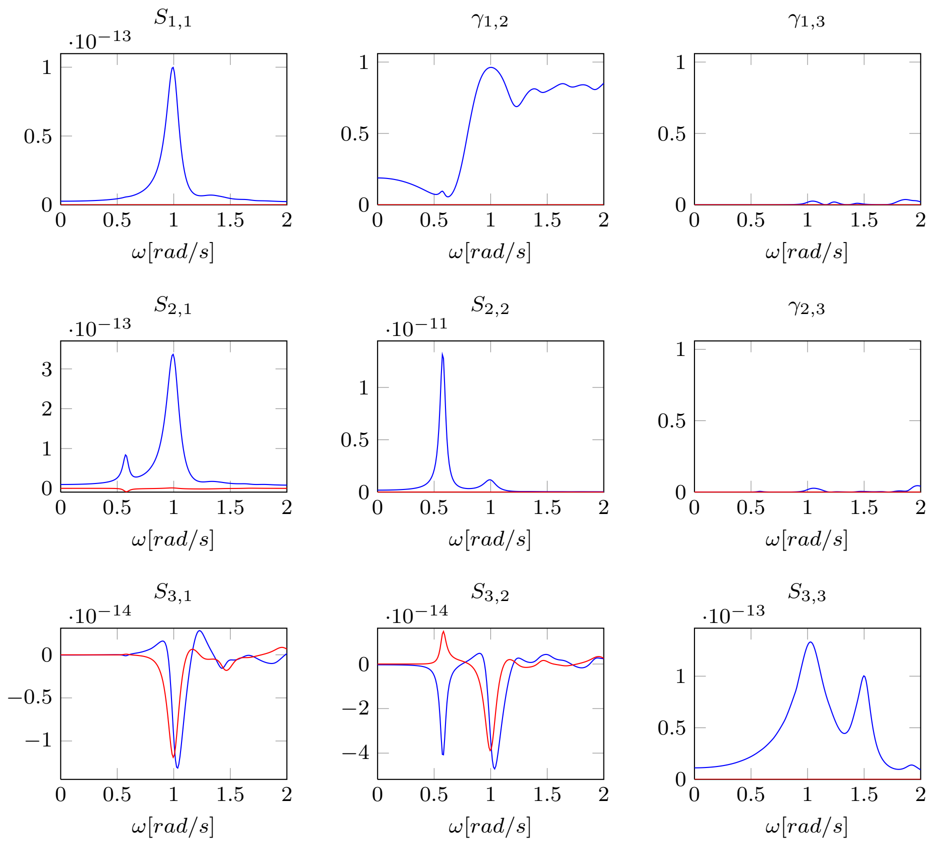
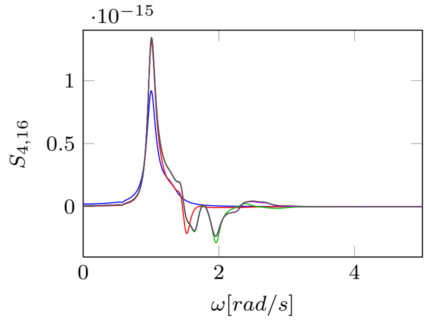
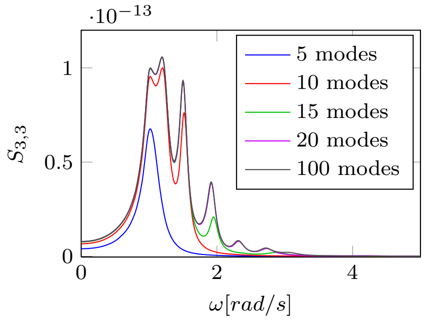


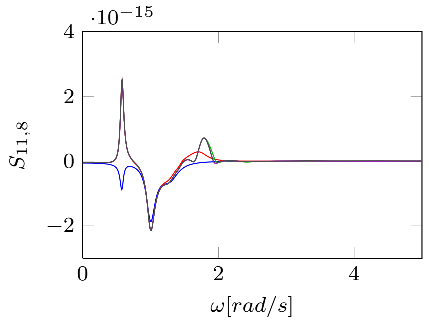
Figure 5.12: *Response spectra for translational DOFs of pontoon number 3 of the Bersøysund Bridge, for the case of white noise load spectra. The plots above the diagonal show the coherence. Blue plots shows real parts while red shows imaginary parts of the solution.*



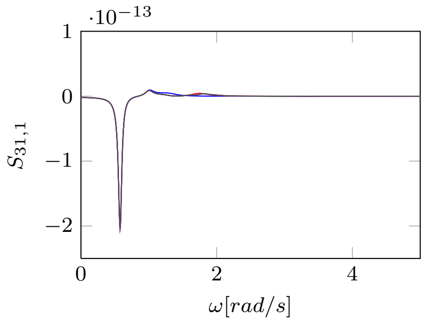
(a) Cross-spectrum between DOF 4 on pontoon 1 and DOF 4 on pontoon 3.



(b) Auto-spectrum of DOF 3 on pontoon 1.

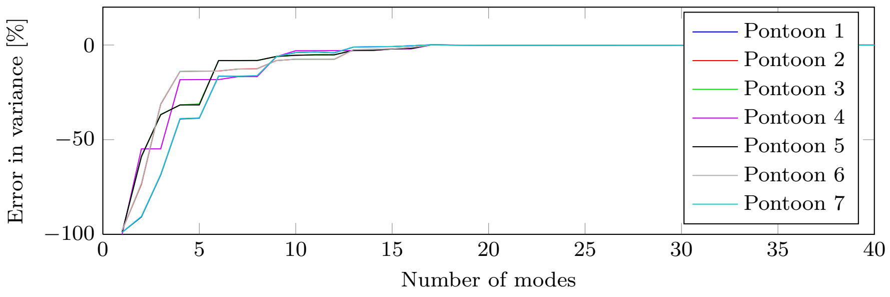


(c) Cross-spectrum between DOF 5 on pontoon 2 and DOF 2 on pontoon 2.

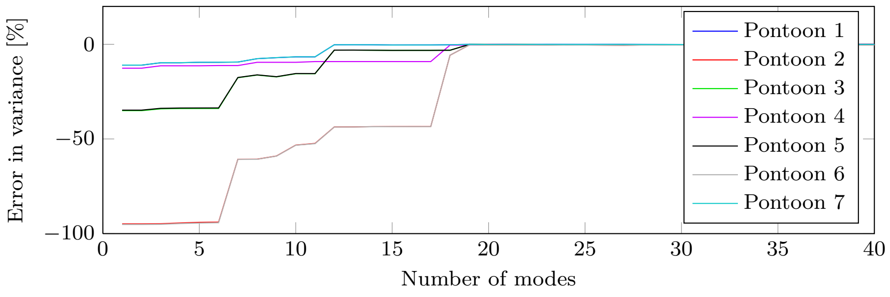


(d) Response spectrum between DOF 1 on pontoon 6 and DOF 1 on pontoon 1.

Figure 5.13: *Selected cross- and auto-covariance spectra for the Bergsøysund Bridge, with different number of modes included in the calculations. Only real parts of the spectra are plotted. Because the white noise loading is assumed unity, the response spectra are also unitless.*



(a) Values of variance for heave (DOF no. 3) on each pontoon.



(b) Values of variance for yaw (DOF no. 6) on each pontoon.

Figure 5.14: *Response variance for DOFs 3 (heave) and 6 (yaw) on the Bersøysund Bridge, as deviance from the variance with all 100 included modes. In this case white noise loading is used.*

for the pontoons appear to be symmetric, as a nearly perfect overlap of the plots is seen. In other words, pontoon pairs 1 & 7, 2 & 6 and 3 & 5 seem to converge with the same rate with respect to the number of modes included. This is not the case for the other DOFs, which makes sense given the non-symmetric orientation of the global coordinate system. These figures support the previous findings regarding convergence with respect to number of modes used; 20 included modes result in near-converged results.

Time series comparison

The amplitude of the spectral densities on the diagonal of the load spectra matrix was set to the constant value $10^{12}N^2$. The Rayleigh damping coefficients were set to $\alpha = 3.9 \cdot 10^{-3}s^{-2}$ and $\beta = 5.1 \cdot 10^{-3}s^{-1}$. A Monte Carlo simulation of this load spectra was used as load on a time domain analysis run by Sindre M. Hermstad, with a total duration of $3000s$. The spectral densities of the resulting time response were thereafter estimated by Hermstad, using Welch's method with 5 subdivisions. Using 100 modes, the response spectra were calculated using the MATLAB program. Figure 5.15 shows the comparison between the time domain and the frequency domain auto-spectral densities, which are seen to agree very well. Even better results are expected if the time domain results are based on more and longer time series. A comparison with different number of modes used is shown in

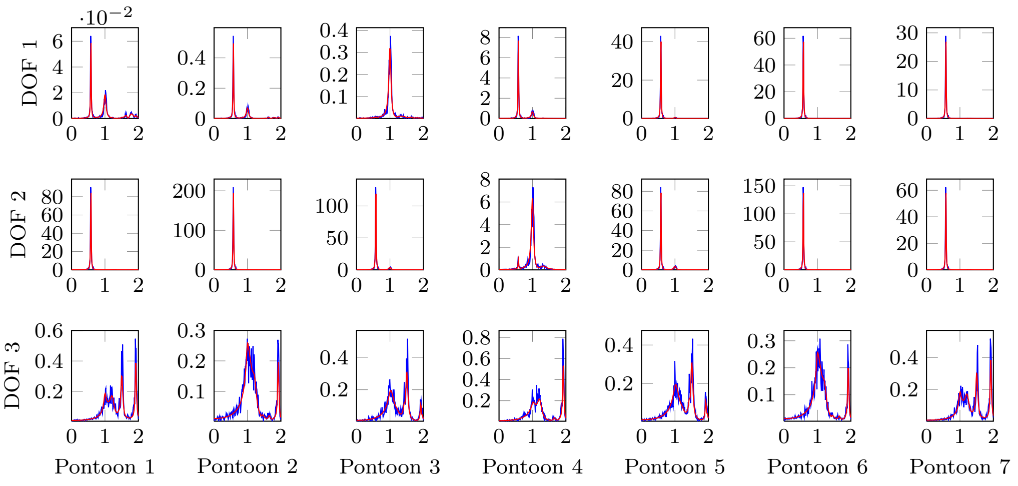


Figure 5.15: *Comparison between two-sided spectra based on time domain solution by Sindre M. Hermstad, and the frequency domain solutions from the* MATLAB *program with 100 modes. Red plots indicate frequency domain solution, while blue plots indicate time domain solution. The x-axis shows the frequency, in rad/s, and the y-axis shows the auto-spectral density of the displacements, in m^2.*

Table 5.3: *Deviance of variances retrieved in the time domain analysis, compared to variances found in the frequency domain analysis.*

	Pontoon no.						
	1	2	3	4	5	6	7
DOF 1	5.7724%	7.7228%	1.6383%	3.7064%	6.3136%	7.3430%	7.9884%
DOF 2	7.1218%	7.7787%	8.2121%	1.5864%	4.7108%	6.9007%	7.6820%
DOF 3	12.2180%	6.2572%	10.4866%	14.7577%	9.0743%	1.9417%	9.6169%

Figure 5.16, for the heave response of pontoon 3. From this it is seen that 10 modes are not sufficient to capture the peaks with higher frequency. It is hard to differentiate the response spectra for 20 and 100 included modes, as stated in the discussion regarding modal convergence, and both curves match the time domain solution good.

The auto-spectral densities found from the frequency domain calculations were integrated over the frequency, yielding the variances of the displacements, as given in Equation 2.101. The resulting variances were further compared with the variances found from the displacements in the time series, by simply enforcing the MATLAB function `var`. This comparsion is shown in Table 5.3, for 100 modes in the frequency domain analysis. It is noted that all the variances from the time domain analysis are larger than the corresponding variances from the frequency domain analysis.

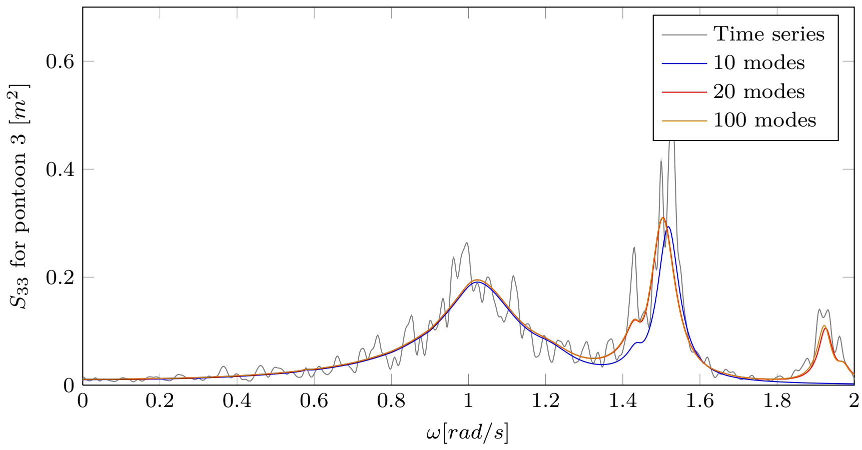


Figure 5.16: *Heave response of pontoon 3, from time domain analysis and frequency domain analysis.*

5.4.3 Response from load spectra

Then, an estimated load spectra matrix calculated by Ragnar Sigbjörnsson was used. Figure 5.17 shows the response cross-spectra for the heave components of all the pontoons. Above the diagonal, the coherences, as introduced in Equation 2.102, are plotted. The coherences plotted are seen to vary dramatically. This is blamed on numerical effects caused by the fact that the values in the denominator and the numerator are both very small.

Modal convergence

A study of the modal convergence was also performed for the case of estimated sea load. The same selected response spectra as for the modal study with the white noise loading are shown in Figure 5.18. Based on this, it is seen that the convergence of the response spectra, with respect to number of modes included, is slower when estimated sea load is used instead of white noise loading. 20 modes still produce response spectra well within engineering norms, but a few more modes are required to observe convergence. By introducing 10 extra modes, the results from Figures 5.18(c) and 5.18(b) improve, as shown in Figures 5.19(b) and 5.19(a). Based on these figures, 30 modes appear to yield near-converged results for the Bergsøysund Bridge model, exposed to the estimated sea load.

Time series comparison

The load spectra routine given did not produce similar loading for the time and frequency domain cases. This deviance is illustrated in Figure C.2, shown in Appendix C. A direct

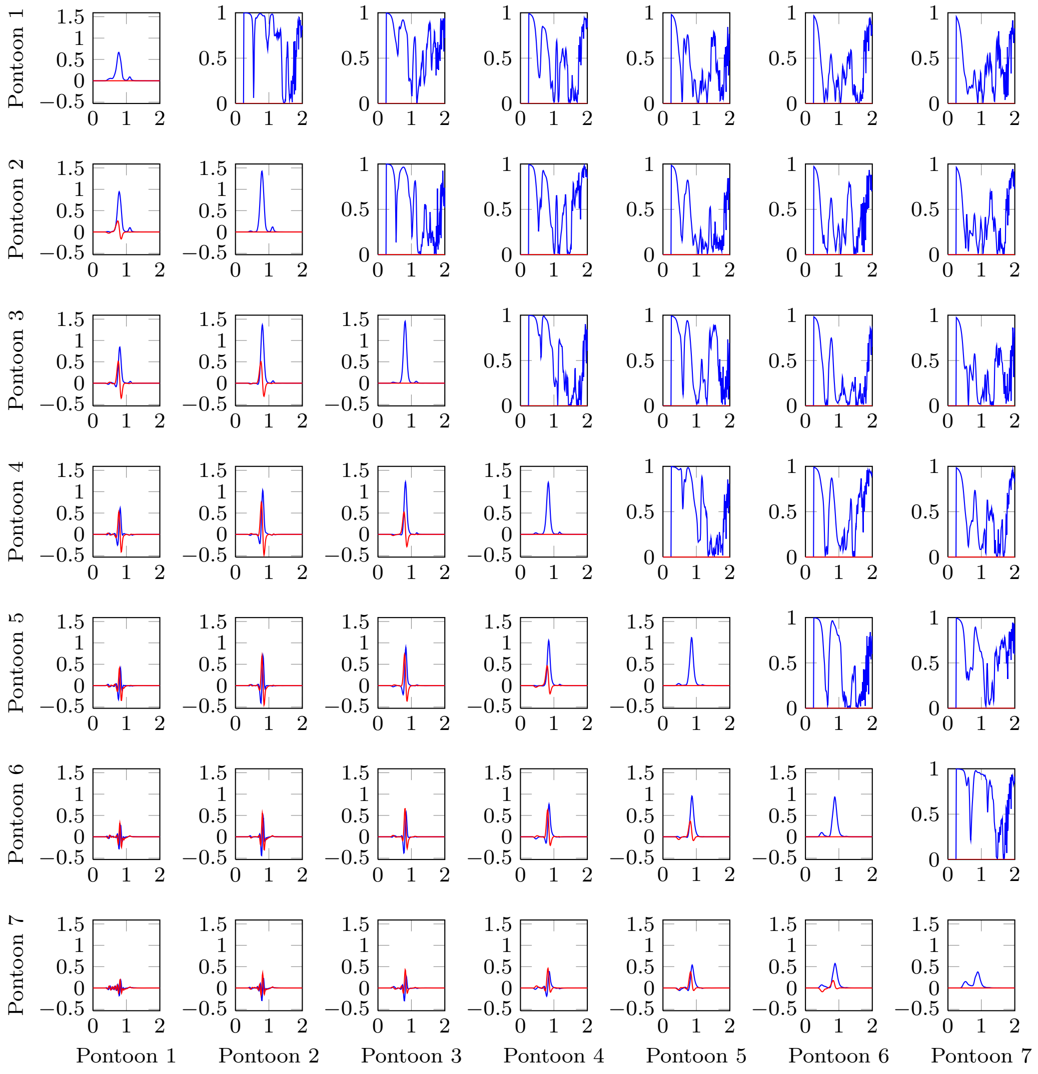
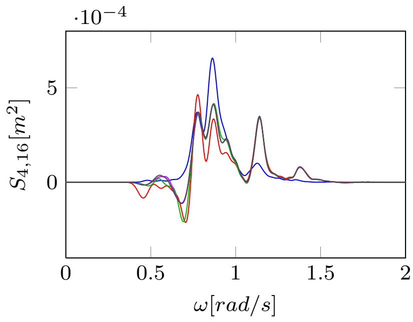
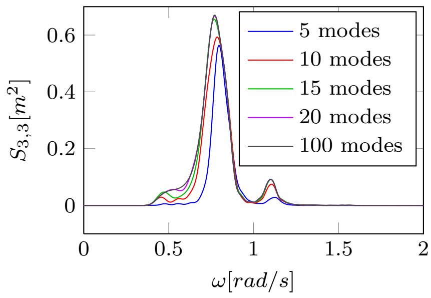


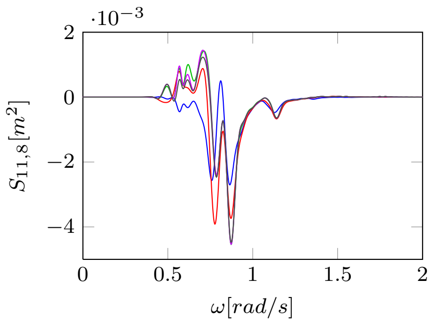
Figure 5.17: *Response spectra for heave (DOF 3) of all pontoons of the Bersøysund Bridge, for the case of estimated load spectra. The x-axis shows the frequency, in rad/s, and the y-axis shows the cross-spectral density between the displacements, in m^2, except for the plots above the diagonal. The plots above the diagonal show the coherence. Blue plots show real parts, while red plots show imaginary parts of the solutions.*



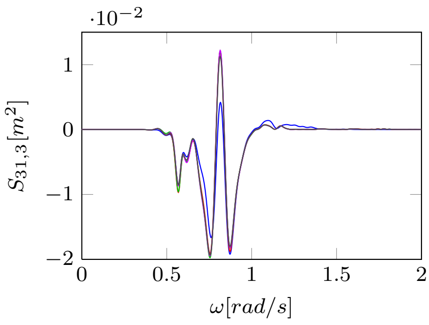
(a) Cross-spectrum between DOF 4 on pontoon 1 and DOF 4 on pontoon 3.



(b) Auto-spectrum of DOF 3 on pontoon 1.

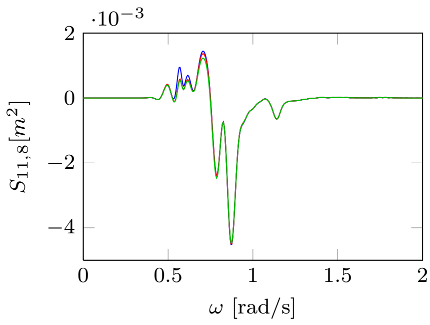


(c) Cross-spectrum between DOF 5 on pontoon 2 and DOF 2 on pontoon 2.

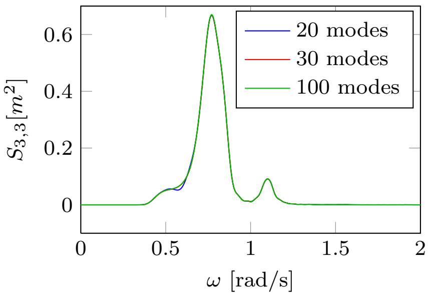


(d) Response spectrum between DOF 1 on pontoon 6 and DOF 1 on pontoon 1.

Figure 5.18: *Selected cross- and auto-covariance spectra for the Bergsøysund Bridge, with different number of modes included in the calculations. Only real parts of the spectra are plotted. Here, estimated sea load is used.*



(a) Cross-spectrum between DOFs 5 and 2 on pontoon 2.



(b) Auto-spectrum for DOF 3 on pontoon 1.

Figure 5.19: *Selected cross- and auto-covariance spectra for the Bergsøysund Bridge, with different number of modes included in the calculations. Only real parts of the spectra are plotted. Here, estimated sea load is used.*

comparison between time and frequency domain response from this load spectra was therefore uninteresting. This is an important case, as it represents an estimation of a physical feasible loading state, rather than the artificially constructed white noise loading. It is therefore recommended that this is done at a later time.

Sequence of addition of wet contributions

As mentioned previously, ideally all the mass, damping and stiffness contributions from the pontoons would have been added to the system before Abaqus calculated the modal sizes. In the project at hand, only the constant contributions were added in Abaqus.

The Matlab program made it simple to include the constant terms *after*, rather than *before*, the calculations in Abaqus were performed. An indication of the impact this had on the solution was wanted, so two similar cases were run with the frequency *in*dependent mass and stiffness contributions added *before* and *after* the Abaqus calculations. The response spectra for the translational DOFs of pontoon 3 are plotted in Figure 5.20, for both these two cases, denoted *before* and *after*, respectively. In both cases, 100 modes were included in the calculations. The tendency shown in the figure is representative for the rest of the response spectra as well. As expected, no significant difference between the two different approaches are noticed. This bodes well for the fact that the mode shapes are based on the frequency *in*dependent terms only, due to the before-mentioned limitations in Abaqus.

5.4.4 Duration of calculations

The program was run on the Bersøysund Bridge multiple times, with different number of modes included, and the time consumption was measured. As seen in Table 5.4, the number of modes included in the calculations affects the used time considerably. All the run times for the response spectra calculations are still well within fair boundaries. The duration of the complex and non-linear eigenvalue problem solving is seen to be more drastically impacted by the number of modes included. The required convergence tolerance, *i.e.* the maximum difference between the previous and the new iterated value of ω_n, was set to $0.001 rad/s$. Also, *10* iterations per eigenvalue was defined as an upper boundary. It is noted that the solving time will fall if these requirements are loosened.

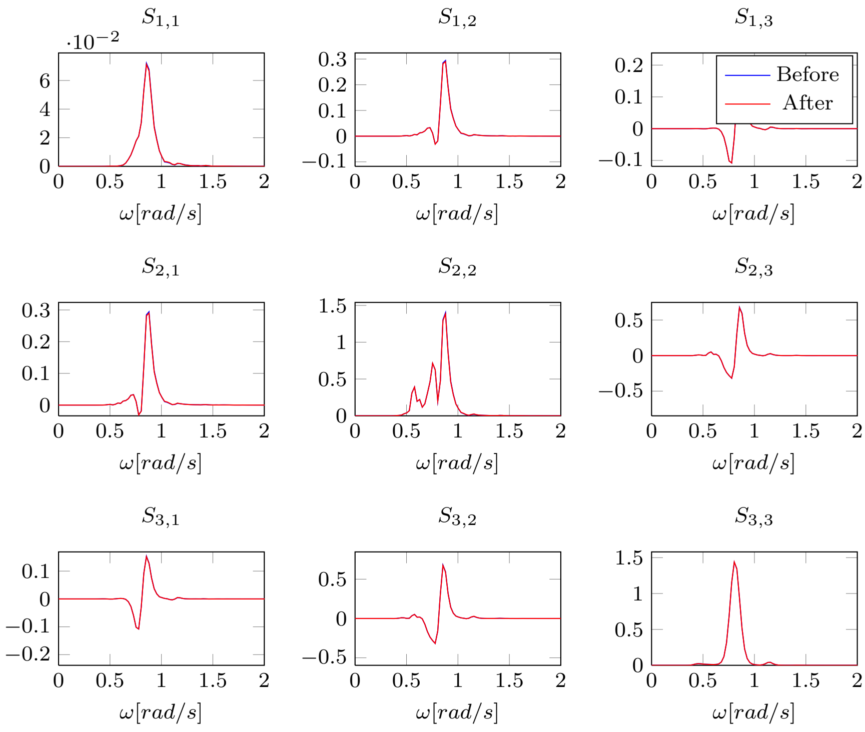


Figure 5.20: *Real part of response spectra for the translational DOFs on pontoon no. 3 on the Bersøysund Bridge. 100 modes were included in both cases.*

Table 5.4: *Approximate durations of calculations of the Bersøysund Bridge. The numbers indicate time consumption on a mid-range laptop. In this case, the frequency axis had 2000 points.*

Operation	Number of modes	Time consumption
Response spectra calculation	100	15 s
	50	8 s
	10	5 s
	1	5 s
Eigenvalue problem solving	100	19 s
	50	2 s
	10	0.06 s
	1	0.04 s

6 Concluding remarks

In this thesis, a MATLAB program was developed based on well-established frequency domain methods, for calculation of floating structures exposed to sea loads. This was in particular used to study the dynamic response of the Bersøysund Bridge exposed to an estimated sea load.

For a simply supported beam connected with one single floating element on the midspan, approximately *10 modes* are seen to give near-converged results. The response spectra of the Bergsøysund Bridge model approaches convergence when approximately *20 modes* are included, when exposed to white noise load. When estimated sea load spectra are enforced, approximately *30 modes* give near-converged results. On the other hand, the 100 modes used on the Bersøysund Bridge model at most, did not introduce considerable computational time. For medium-sized problems, like the one at hand, the computational time spent with 100 included modes, is considered unproblematic.

The resulting response spectra compare well with those retrieved from equivalent time domain calculations, when white noise loading is enforced. This comparison is based on the exact same structural model and the same pontoon models, which represent a major weakness. On the other hand, this gives the best indication of how the methods themselves compare. No comparison of response for estimated sea load was performed, due to inconsistency between the load representations in time and frequency domains.

For the best possible accuracy with a given number of included modes, modal transformations should be done after all additions to system stiffness, damping and mass are included, including the frequency dependent contributions. This is as far as the author knows, not currently possible in ABAQUS. For 100 included modes, adding the frequency *in*dependent contributions from the pontoons *before* or *after* the modal transformation, do not lead to any significant differences. As long as adequately many modes are included, this will also be the case for the frequency dependent terms of the pontoons.

The natural frequencies and damping ratios were recalculated after all modal sizes were added, by using state space form and a simple iterative scheme. The resulting frequencies matched well with the measured peaks in the response. Natural frequencies found from all system properties, including frequency dependent ones, deviated considerably from the eigenfrequencies found in ABAQUS using only the frequency *in*dependent contributions. In general, the natural frequencies were seen to be lowered when the frequency dependent terms were added.

6.1 Future work

Further investigation of some important aspects, not within the scope of this thesis, is wanted. The most important ones are summarized below.

- Use equal load spectra input for frequency and time domain analyses, and compare the results.
- Perform measurements of the displacements of the Bergsøysund Bridge, and compare the measurements with calculated spectra based on a load spectra representing the real wave loading state in Bergsøysund.
- In the present work, a fully linear behaviour has been assumed. The possibility to include non-linear effects should be investigated.
- Other environmental loading than sea load should be included in a comprehensive calculation set-up.

References

[1] Norwegian Public Roads Administration Olav Elleveset. Project Overview Coastal Highway Route E39, 2012.

[2] Ivar Langen and Ragnar Sigbjörnsson. *Dynamisk analyse av konstruksjoner.* Tapir, 1979.

[3] Masanobu Shinozuka. Monte carlo solution of structural dynamics. *Computers and Structures*, 2(5–6):855 – 874, 1972.

[4] Ole Øiseth, Abdillah Suyuthi, Bernt Leira, Kjell Magne Mathisen, Anders Rönnquist, Ragnar Sigbjörnsson, and Svein Remseth. Predicition of wave induced dynamic response in time domain using the finite element method, 2013.

[5] Erwin Kreyszig. *Advanced Engineering Mathematics.* Wiley, 2006.

[6] Anil K. Chopra. *Dynamics of Structures - Theory and Applications to Earthquake Engineering.* Pearson Prentice Hall, University of California at Berkeley, 2007.

[7] Gerard Lallement and Daniel J. Inman. A tutorial on complex eigenvalues, 1995.

[8] Vetco Aibel AS Christopher Hoen. An engineering interpretation of the complex eigensolution of linear dynamic systems, 2005.

[9] Peter Fischer. Eigensolution of nonclassically damped structures by complex subspace iteration, 1999.

[10] Fernando Cortés and María Jesús Elejabarrieta. Computational methods for complex eigenproblems in finite element analysis of structural systems with viscoelastic damping treatments, 2006.

[11] Fredric J. Harris. On the use of windows for harmonic analysis with the discrete Fourier transform, 1978.

[12] Peter D. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *Audio and Electroacoustics, IEEE Transactions on*, 15(2):70–73, 1967.

[13] Julius S. Bendat and Allan G. Piersol. *Random Data - Analysis and Measurement Procedures.* John Wiley and Sons, Inc., 1986.

[14] Walpole, Myers, Myers, and Ye. *Probability and Statistics for Engineers and Scientists.* Pearson Education, 2012.

[15] Arvid Naess. *An introduction to random vibrations.* Centre for Ships and Ocean Structures Norwegian University of Science and Technology NO-7491 Trondheim, Norway, 2012.

[16] Einar Strømmen. *Theory of Bridge Aerodynamics.* Springer-Verlag Berlin Heidelberg, Department of Structural Engineering Norwegian University of Science and Technology 7491 Trondheim, Norway, 2006.

[17] Ivar Langen. Frequency domain analysis of a floating bridge exposed to irregular short-crested waves.

[18] DNV. DNV HydroD Wadam web information. `http://www.dnv.com/services/software/products/sesam/sesamhydrod/wadam.asp`.

[19] Abdillah Suyuthi. Hydrodynamic analysis of floating bridge - modeling of a single pontoon (no 1 of 7), 2012.

[20] D. E. Cartwright and M. S. Longuet-Higgins. The statistical distribution of the maxima of a random function. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 237(1209):pp. 212–232, 1956.

[21] John J. Bertin and Russel M. Cummings. *Aerodynamics for engineers.* Prentice Hall, 2009.

[22] D.J. Auld and K. Srinivas. Aerodynamics for students. `https://sites.google.com/site/aerodynamics4students/table-of-contents`.

A Supplementary theory

A.1 Engineering results from spectral values

When response spectra for the entire structure are calculated, the stochastic measures should be interpreted. Important considerations are among others maximum displacements, maximum internal forces, and for fatique analyses also the count of threshold crossings for *i.e.* displacements and stresses.

A.1.1 Threshold crossings

The following is adapted from [2]. The *Heaviside step function* reads out [5]

$$u(x-a) = \begin{cases} 1 & x > a \\ 0 & x < a \end{cases} \tag{A.1}$$

Introduce an additional condition for the case of zero in the function argument:

$$u(x-a) = \begin{cases} 1 & x > a \\ \frac{1}{2} & x = a \\ 0 & x < a \end{cases} \tag{A.2}$$

By using the Heaviside function, the following is introduced:

$$y(t) = H(u(t) - \xi) \tag{A.3}$$

Here, $u(t)$ is a scalar displacement in the time domain and ξ is the predefined threshold value set for the displacement. Assuming $u(t)$ is differentiable, Equation A.3 yields the following threshold crossings count over the time interval T:

$$\dot{y}(t) = \dot{u}(t)\delta(r(t) - \xi) \tag{A.4}$$

where $\delta(\cdot)$ is the *Dirac delta function.* For a stationary displacement, Equation A.4 yields

$$n(\xi, T) = \int_t^{t+T} |\dot{u}(t)|\delta(u(t) - \xi)dt \tag{A.5}$$

For a stochastic displacement, the expected number of threshold crossings reads out

$$E(n(\xi, T)) = T \cdot E\left[|\dot{u}(t)|\delta(u(t) - \xi)\right] \tag{A.6}$$

$$= T \cdot \int_{-\infty}^{\infty} |\dot{u}| f(u, \dot{u})(\xi, \dot{u}) d\dot{u} \tag{A.7}$$

where $f(u, \dot{u})$ is the joint probability distribution between u and $\dot{u}$. For the case where $u(t)$ is an ergodic Gaussian process with $\mu = 0$, [2] gets

$$f(u, \dot{u}) = \frac{1}{2\pi\sigma_u\sigma_{\dot{u}}} \cdot exp(-\frac{u^2}{2\sigma_u^2} - -\frac{\dot{u}^2}{2\sigma_{\dot{u}}{}^2}) \tag{A.8}$$

and which by imposing Equation A.6, further gets the following number of crossing per time T:

$$\frac{1}{T} \cdot E[n(\xi, T)] = \frac{1}{2\pi}\frac{\sigma_{\dot{u}}}{\sigma_u} \cdot exp(-\frac{\xi^2}{2\sigma_u^2}) \tag{A.9}$$

A.1.2 Extremal values

The standard deviation of $u(t)$ can be found from the spectral density;

$$\sigma_u^2 = \int_{-\infty}^{\infty} S_u(\omega)d\omega \tag{A.10}$$

Consider a time interval, T. According to [17], the largest maximum has this expectancy value:

$$E[u_{max}] = \sigma_u \cdot \left[\sqrt{2 \cdot \ln\frac{T}{T_z}} + \frac{0.5772}{\sqrt{2 \cdot \ln\frac{T}{T_z}}}\right] \tag{A.11}$$

where T_z is introduced as the *zero crossing period.* This can be found by choosing threshold value $\xi = 0$ in Equation A.9:

$$T_z = 2\pi\sqrt{\frac{\sigma_u}{\sigma_{\dot{u}}}} \tag{A.12}$$

When $\dot{u} = i\omega u$ and the spectral density of the velocity is given by Equations 2.81 and 2.85 the zero crossings period can be rewritten

$$T_z = 2\pi\sqrt{\frac{\sigma_u}{\omega^2\sigma_u}} = \frac{2\pi}{\omega} \tag{A.13}$$

which in fact is the same as the natural period of the displacement. Further, [17] introduces the standard deviation of the largest maximum as

$$\sigma_{u_{max}} = \sigma_u \frac{\pi}{\sqrt{12 \cdot \ln\frac{T}{T_z}}} \tag{A.14}$$

For more information about statistical distribution of maxima, it is referred to Cartwright and Longuet-Higgins [20].

A.2 Establishing transformation matrix for pontoon

The added stiffness and mass matrices for the pontoon found using DNV HydroD Wadam correspond to DOFs placed in the centroid of the top plane $\{\mathbf{v}\}$. In the first Abaqus model on the other hand, the pontoon was modelled with four points connected to the bridge frame, with DOFs denoted $\{\bar{\mathbf{v}}\}$ here. To be able to add the hydrodynamic mass and stiffness matrices, a transformation matrix between the two sets of degrees of freedom was sought.

If the pontoon is assumed rigid (a very fair assumption), the following constraints are true:

$$\bar{v}_{1A} = \bar{v}_{1D} \tag{A.15}$$
$$\bar{v}_{1B} = \bar{v}_{1C} \tag{A.16}$$
$$\bar{v}_{2A} = \bar{v}_{2B} \tag{A.17}$$
$$\bar{v}_{2C} = \bar{v}_{2D} \tag{A.18}$$
$$\bar{v}_{4B} = \bar{v}_{4C} \tag{A.19}$$
$$\bar{v}_{4A} = \bar{v}_{4D} \tag{A.20}$$
$$\bar{v}_{5A} = \bar{v}_{5B} \tag{A.21}$$
$$\bar{v}_{5C} = \bar{v}_{5D} \tag{A.22}$$
$$\bar{v}_{6A} = \bar{v}_{6B} = \bar{v}_{6C} = \bar{v}_{6D} \tag{A.23}$$

The contraints in Equations A.15 - A.23 and Figure A.1 give

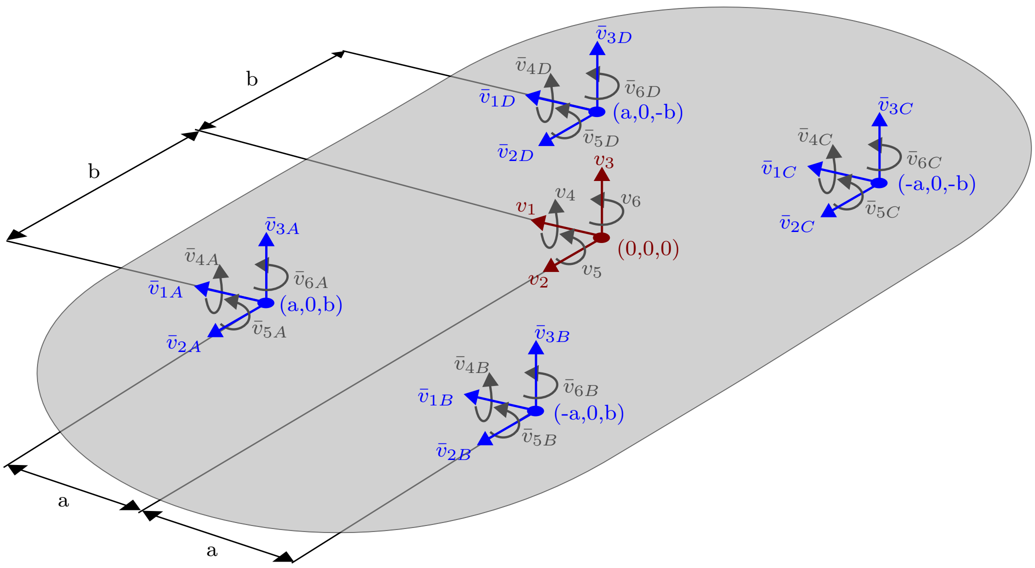


Figure A.1: *Two sets of DOFs;* $\{\mathbf{v}\}$ *and* $\{\bar{\mathbf{v}}\}$.

$$v_1 = \bar{v}_1 = \frac{1}{4}[\bar{v}_{1A} + \bar{v}_{1B} + \bar{v}_{1C} + \bar{v}_{1D}] \tag{A.24}$$

$$v_2 = \bar{v}_2 = \frac{1}{4}[\bar{v}_{2A} + \bar{v}_{2B} + \bar{v}_{2C} + \bar{v}_{2D}] \tag{A.25}$$

$$v_3 = \bar{v}_3 = \frac{1}{4}[\bar{v}_{3A} + \bar{v}_{3B} + \bar{v}_{3C} + \bar{v}_{3D}] \tag{A.26}$$

$$v_4 = \bar{v}_4 + \frac{1}{4b}[-\bar{v}_{3A} - \bar{v}_{3B} + \bar{v}_{3C} + \bar{v}_{3D}] \tag{A.27}$$

$$v_5 = \bar{v}_5 + \frac{1}{4a}[-\bar{v}_{3A} + \bar{v}_{3B} + \bar{v}_{3C} - \bar{v}_{3D}] \tag{A.28}$$

$$v_6 = \bar{v}_6 + \frac{1}{4b}[-\bar{v}_{1A} - \bar{v}_{1B} + \bar{v}_{1C} + \bar{v}_{1D}] + \frac{1}{4a}[\bar{v}_{2A} - \bar{v}_{2B} - \bar{v}_{2C} + \bar{v}_{2D}] \tag{A.29}$$

where it is used that

$$\bar{v}_i = \frac{1}{4}[\bar{v}_{iA} + \bar{v}_{iB} + \bar{v}_{iC} + \bar{v}_{iD}] \tag{A.30}$$

Assume that

$$\{\mathbf{v}\} = [\mathbf{T}]\{\bar{\mathbf{v}}\} \tag{A.31}$$

and that the degrees of freedom are stacked in the following manner:

$$\{\mathbf{v}\} = \begin{Bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{Bmatrix} \tag{A.32}$$

$$\{\bar{\mathbf{v}}\} = \begin{Bmatrix} \bar{v}_{1A} \\ \bar{v}_{2A} \\ \bar{v}_{3A} \\ \bar{v}_{4A} \\ \bar{v}_{5A} \\ \bar{v}_{6A} \\ \bar{v}_{1B} \\ \bar{v}_{2B} \\ \bar{v}_{3B} \\ \bar{v}_{4B} \\ \bar{v}_{5B} \\ \bar{v}_{6B} \\ \bar{v}_{1C} \\ \bar{v}_{2C} \\ \bar{v}_{3C} \\ \bar{v}_{4C} \\ \bar{v}_{5C} \\ \bar{v}_{6C} \\ \bar{v}_{1D} \\ \bar{v}_{2D} \\ \bar{v}_{3D} \\ \bar{v}_{4D} \\ \bar{v}_{5D} \\ \bar{v}_{6D} \end{Bmatrix} \tag{A.33}$$

This gives the following transformation matrix:

$$[\mathbf{T}] = \frac{1}{4}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{b} & 1 & 0 & 0 & 0 & 0 & -\frac{1}{b} & 1 & 0 & 0 & 0 & 0 & \frac{1}{b} & 1 & 0 & 0 & 0 & 0 & \frac{1}{b} & 1 & 0 & 0 \\ 0 & 0 & -\frac{1}{a} & 0 & 1 & 0 & 0 & 0 & \frac{1}{a} & 0 & 1 & 0 & 0 & 0 & \frac{1}{a} & 0 & 1 & 0 & 0 & 0 & -\frac{1}{a} & 0 & 1 & 0 \\ -\frac{1}{b} & \frac{1}{a} & 0 & 0 & 0 & 1 & -\frac{1}{b} & -\frac{1}{a} & 0 & 0 & 0 & 1 & \frac{1}{b} & -\frac{1}{a} & 0 & 0 & 0 & 1 & \frac{1}{b} & \frac{1}{a} & 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.34}$$

A.3 Aerodynamics

To calculate the contributions to system mass, damping and stiffness from the pontoons, DNV HydroD Wadam was used. This utilizes a *panel method*, and the resulting matrices are dependent of the frequency of the load applied. To better understand the way DNV HydroD Wadam works, the methodology behind panel methods is presented. Before the panel method can be developed, important presumptions are needed.

A.3.1 Basic fluid mechanics

Stoke's theorem [21] reads out

Theorem A.3.1. *The circulation of a fluid is defined as*

$$\Gamma = \oint_C \vec{v} \cdot \vec{ds} = \int\int_A (\nabla \times \vec{v}) \cdot \hat{n} dA \tag{A.35}$$

Here $\hat{n}dA$ represents the normal vectors of the surface A, and C a curve enclosing the surface A. $\vec{ds}$ is the differential tangent vector of the curve C.

This is further elaborated in [5]. Further, an *irrotational* fluid is mathematically defined:

Definition A.3.1. *If a flow satisfies*

$$\nabla \times \vec{v}(x, y, z) \equiv 0$$

and no singularities are present, the flow is irrotational [21]. Here $\vec{v} = [u, v, w]^T$ is the velocity at a point (x,y,z) in the fluid and ∇ is the del-operator, defined

$$\nabla = \begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{Bmatrix}$$

A.3.2 Linear potential functions

The following is adapted from [21]. Certain presumptions are needed for a linear potential function to exist for a flow. The flow needs to be potential;

Definition A.3.2. *If there exists a velocity potential $\phi(x, y, z)$ for a flow, such that*

$$\vec{v} = \nabla\phi$$

the flow is called a potential flow [21].

According to Stoke's theorem we have the following for an irrotational fluid:

$$\Gamma = \oint_C \vec{v}\vec{ds} = 0 \tag{A.36}$$

The line integral in Equation A.36 is independent of path, which is satisfied for $curl\vec{v} \equiv 0$. When a line integral is independent of the integration path, this indicates that the integral value only depends on the limits of integration. A line integral independent of its path requires that the integrand is an exact differential;

$$\vec{v} \cdot \vec{ds} = d\phi \tag{A.37}$$

Here, $d\phi$ is an exact differential. Equation A.37 can be written

$$udx + vdy + wdz = \frac{\partial\phi}{\partial x}dx + \frac{\partial\phi}{\partial y}dy + \frac{\partial\phi}{\partial z}dz \tag{A.38}$$

This can be written more compact as

$$\vec{v} = \nabla\phi \tag{A.39}$$

which is nothing more than what is stated in Definition A.3.2 for a potential flow. Thus, an irrotational flow is also a potential flow.

Incompressible fluids can not change volume, or in other words, have constant density. For constant density the continuity equation [21] becomes

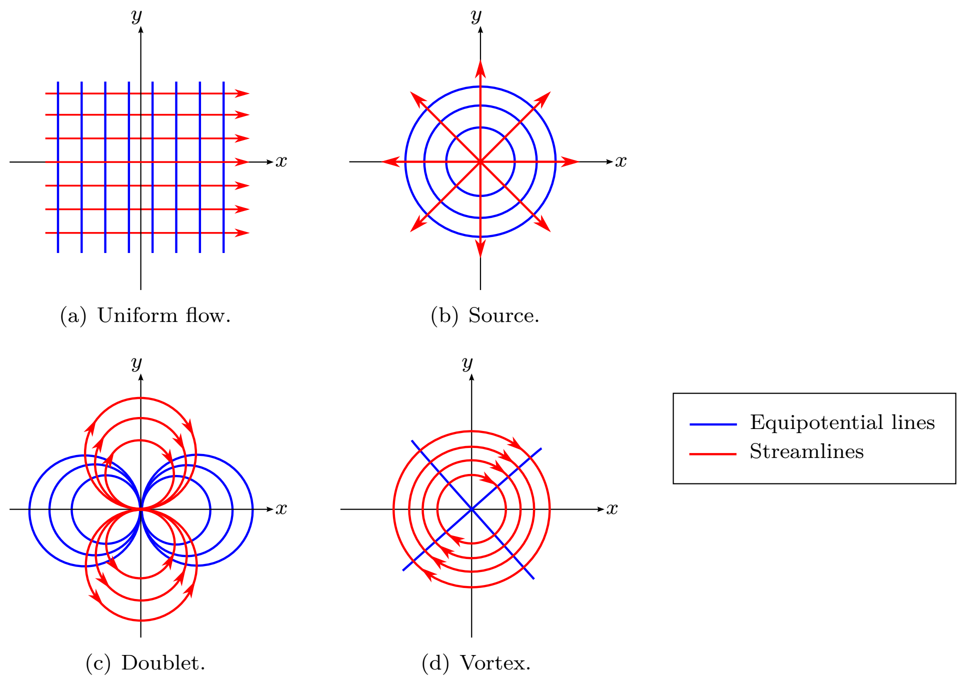


Figure A.2: *The most important flows with potential functions. Streamlines are lines parallel to the movement of the particles in the fluid, and equipotential lines are lines with $\phi = const$.*

$$\nabla \cdot \vec{v} = 0 \tag{A.40}$$

If it is assumed that the fluid considered is incompressible and irrotational, Definition A.3.1 and Equation A.40 yield

$$\nabla^2 \phi = 0 \tag{A.41}$$

where the *Laplacian*, ∇^2 is defined $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$. This equation is known as *Laplace's equation*, and needs to be satisfied for a potential function used to describe the flow in an incompressible and irrotational fluid. The equation is a second order, linear partial differential equation.

Using these assumptions, velocity potential functions for elementary flow patterns have been constructed. The four most important flow patterns are shown in Figure A.2, and their velocity potential functions given in Table A.1. For development of these functions, it is referred to [21].

Because Equation A.41 is linear, potential functions that satisfy this can be super-positioned to obtain potential functions of more complex flows. This is the basis for panel methods

Table A.1: *Potential functions for elementary flows in 2D. Here cylindrical coordinates (r, θ) are used.*

Flow type	Potential, ϕ	Comment
Uniform flow	$\phi = U_\infty rcos(\theta)$	U_∞ is the magnitude of the velocity. Shown in Figure A.2(a).
Source/sink	$\phi = \frac{K}{2\pi} ln(r)$	For negative source strength K, the function represents a sink. Shown in Figure A.2(b).
Doublet	$\phi = \frac{B}{r} cos(\theta) cos(\alpha)$	Shown in Figure A.2(c).
Vortex	$\phi = \frac{\Gamma\theta}{2\pi}$	Shown in Figure A.2(d).

in computational aerodynamics. When the velocity potential is known, the velocity field is easily determined using the equation given in Definition A.3.2 and further the pressure field using *Bernoulli's equation.* Bernoulli's equation is only valid for flows with the following properties [21];

- Inviscid
- Incompressible
- Steady-state (stationary)
- Irrotational (or only along a streamline)
- With conservative body forces

and relates the pressure, elevation and velocity of particles in the fluid. In practise, an irrotational flow is also inviscid. That is, no friction or drag forces are present. In real-life problems, this is often a too drastic assumption, and it is often needed to solve a set of *boundary-layer equations* to establish skin friction afterwards. This is beyond the reach of this project, and reference is therefore made to [21] for information about this subject.

A.3.3 Panel method

Panel methods constitute an important tool in numerical aerodynamics. They are based on superposition of linear potential functions for simple features. By approaching the appearance of the real flow with simple linear potential functions, it is possible to estimate important aerodynamic sizes, of almost arbitrary physical surfaces. The simplistic and generic nature of the methods makes them well suited for implementation in numerical software.

For the calculation of the forces induced on a surface due to some flow around it, two main steps are required:

1. Inviscid panel method

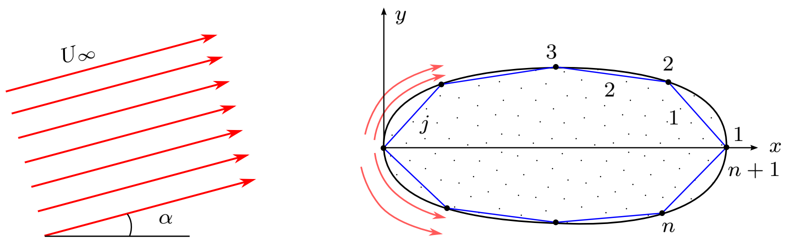
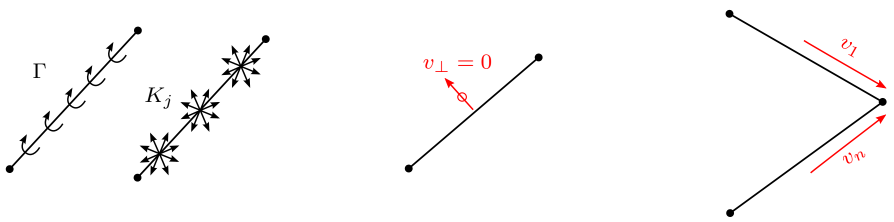


Figure A.3: *Panel method discretization. Two-dimensional surface is discretized into flat panels, which are given conditions as shown in Figure A.4. Panel j, numbered inside body, connects nodes j and $j+1$, numbered outside body.*



(a) Source and vortex strengths for panel j.

(b) Impenetrability implies no normal velocity.

(c) Kutta condition requires equal velocities on trailing edge. This implies $v_1 = v_n = 0$.

Figure A.4: *Construction and physical requirements for panels.*

2. Viscous boundary layer calculations

Only the first of the two steps is presented here, and two-dimensional flow conditions assumed (flow only dependent on two coordinates).

The following is adapted from [22]. Nodes are defined along the two-dimensional surface shown in Figure A.3. A uniform flow with attack angle α is assumed. Panels connect the nodes together. With nodes numbered 1 to $n+1$, each panel j connects nodes j and $j+1$, yielding a total of n panels. Each panel is assigned a source strength K_j and a vorticity Γ constant for all panels. These parameters correspond to the parameters in the potential functions for source and vortex, shown in Figure A.2 and explained in Table A.1.

From the concept of impenetrability it is required that the velocity normal to the surface in the middle of all panels. Also, for the trailing edge (where the flow leaves the surface, at right hand side), it is required that the velocity on the upper and lower surface equal both in magnitude and velocity, to avoid discontinuities. Thus, both velocities have to be zero. This is the *Kutta condition*. These two conditions make it possible to solve the unknown source strengths and vorticities for all panels.

The normal velocity is written for panel i as

$$v_{\perp,i} = \sum_{j=1}^{n} A_{ij} K_j + \sum_{j=1}^{n} B_{ij} \Gamma + C_i U_\infty = 0 \tag{A.42}$$

The contribution from sources and vortices placed on panel j, on the total normal velocity on panel i, are given by A_{ij} and B_{ij}, respectively. The contribution of the uniform flow is given by C_{ij}. These coefficients are functions of the spatial coordinates, since the velocity contribution will be dependent of distance. This will give n equations. There are $n+1$ unknowns.

The following is true regarding the trailing edge tangential velocities:

$$v_{||,1} + v_{||,n} = \sum_{j=1}^{n} D_j K_j + \sum_{j=1}^{n} E_j \Gamma + C_{n+1} U_\infty = 0 \tag{A.43}$$

Here, the contribution from sources and vortices on panels j, on the total tangential velocity on panels 1 and n, are given by D_j and E_j, respectively. The contribution of the uniform flow on the panels is given by C_{n+1}. Equations A.42 and A.43 give the following equation system:

$$\begin{bmatrix} A_{11} & A_{12} & ... & A_{1n} & \sum_{j=1}^{n} B_{1j} \\ A_{21} & A_{22} & ... & A_{2n} & \sum_{j=1}^{n} B_{2j} \\ ... & ... & ... & ... & ... \\ A_{n1} & A_{n2} & ... & A_{nn} & \sum_{j=1}^{n} B_{nj} \\ D_1 & D_2 & ... & D_n & \sum_{j=1}^{n} E_j \end{bmatrix} \begin{Bmatrix} K_1 \\ K_2 \\ ... \\ K_n \\ \Gamma \end{Bmatrix} = -U_\infty \begin{Bmatrix} C_1 \\ C_2 \\ ... \\ C_n \\ C_{n+1} \end{Bmatrix} \tag{A.44}$$

After determining the unknown vorticity and source strengths by solving the equation system, the tangential velocities on all panels can be calculated, and the pressure field determined using Bernoulli's equation.

B Matlab code

B.1 floatingBridge.m

```
%% FLOATINGBRIDGE.m Frequency domain script for calculations of response spectra.
%
% Required functions (sub—functions in parantheses):
%     * structuralMatrices.m (dat2eig.m)
%     * hydroMatrices.m
%     * WADAM_Import.m
%     * iterateFreq.m (eigSolve.m)
%
% Required input files:
%     * Wadam results file(s) (minimum 1, maximum 2)
%     * Abaqus data file from frequency job, with displacements and coordinates of ...
     all nodes for all modes
%     * Geometry ASCII file, containing geometry of pontoons, with this structure
%
%      LINE 1:     | Pontoon     X       Y       Z       Angle |
%      LINE 2:     |    1       X_1     Y_1     Z_1     a_1    |
%      ...         |    :        :       :       :       :     |
%      LINE N:     |    1       X_N     Y_N     Z_N     a_N    |
%
%
%
% Knut Andreas Kvaale (c) 2013
%

%% CLEAN UP
clear all;

%% PARAMETERS, FILENAMES AND DATA PATH
% RAYLEIGH DAMPING
alpha=5e—2;%3.9e—3; %mass proportional damping
beta=3e—2;%5.1e—3;  %stiffness proportional damping
rayleighPontoons=0; %rayleigh damping on constant additions from pontoons?

% GENERAL OPTIONS
includeWet=true;                    %true to include pontoons, false to exclude pontoons
includeDry=true;                    %true to include frame, false to exclude frame
whiteNoise=false;                   %use white noise instead of load spectra? ...
     Numerical value indicate amplitude

whiteNoisePontoons=[1:7];           %which pontoons j to assign Sq_j=eye(6)
calculateComplexeigs=false;         %true to calculate xi and wn again with all ...
     contributions
exportData=false;                   %export the data to BRIDGECALCULATIONS.mat

```

```
                  %const    f(w)        1 if included, 0 if not:
includedInAbaqus= [1        0;          % |  K0   K(w) |
                   0        0;          % |  C0   C(w) |
                   1        0];         % |  M0   M(w) |

includedModes=[1:100];  %modes to be included in calcs (from Abaqus dat-file)

% PONTOONS
wadamfile_type1='Pontoon1and7.txt'; %file name, pontoon type 1
wadamfile_type2='Pontoon2to6.txt';  %file name, pontoon type 2
pontoons_type1=[1,7];               %assign pontoon type 1 to these pontoons
pontoons_type2=[2:6];               %assign pontoon type 2 to these pontoons
h=0;%6.07*0.5;  %dz between DOFs of load spectra and global connections

% INPUT FILES
path='DATA\';   %path of all data files (dat, CSD, geometry, wadam...)
datfile='bsb_freq_job_1000.dat';
CSDfile='CROSS0206_1113.mat';
geometryFile='geometryData.txt';

% LOAD SPECTRA
load([path CSDfile]);
omegaLoad=w;    %make sure to assign omegaLoad = frequency output from spectra
CSD=CSD;        %and CSD = load spectra (ordered p1,p2,dof1,dof2,freq in 5D array)
globalDOFs=false;    %when ==false, the load spectra is given in local DOFs, and ... thus transformed to global coordinates using angles from geometry file

% OMEGA-AXIS
N=1000; %number of measurement points (data set is interpolated)
omegaMax=5;%max(omegaLoad); %maximum value of omega axis
omega=linspace(0,omegaMax,N);    %new interpolated omega axis

%% GEOMETRY FROM GEOMETRYDATA-FILE
fid=fopen([path geometryFile]);
[¬]=fgetl(fid); %skip first line

while true
   tline=fgetl(fid);

   if ¬ischar(tline)
       break
   end
   tline=str2num(tline);
   pontoon=tline(1);
   pontoonAngles(pontoon) = tline(5);   %check this again and again before delivery :)!
   XYZpontoons(pontoon,:) = tline(2:4)';
end

pontoonMax=length(pontoonAngles);

%% IF ONLY PONTOONS
if includeDry==false && includeWet==true
   Phi=eye(6*pontoonMax);
   Nmodes=size(Phi,2);
   Ksg=zeros(Nmodes); Msg=zeros(Nmodes); Csg=zeros(Nmodes);
end

%% RETRIEVE GENERALIZED STRUCTURAL PROPERTIES
if includeDry==true
    [Ksg,Msg,Phi,wn]=structuralMatrices([path datfile], XYZpontoons);
    Nload=length(omegaLoad);
    Csg=alpha.*Msg+beta.*Ksg;  %Rayleigh damping

    xi=Csg(includedModes)./(2.*sqrt(Ksg(includedModes).*Msg(includedModes)));
    wn=wn(includedModes);

    Ksg=diag(Ksg(includedModes));
    Csg=diag(Csg(includedModes));
    Msg=diag(Msg(includedModes));

    Phi=Phi(:, includedModes);

    Nmodes=size(Phi,2);


elseif includeDry==false
    Ksg=zeros(Nmodes);
    Csg=zeros(Nmodes);
    Msg=zeros(Nmodes);
    wn=[];
    xi=[];
else
    error('includeDry should be logical, i.e. equal 0 or 1')
end

%% RETRIEVE GENERALIZED HYDRODYNAMIC PROPERTIES
if includeWet==true
    include=includedInAbaqus==0;    %boolean "invert" matrix

if rayleighPontoons==true
    rayleigh=[alpha; beta];
else
    rayleigh=[0;0];
end

[Khg1, Chg1, Mhg1,omegaWadam1,Kh1,Ch1,Mh1] = hydroMatrices([path wadamfile_type1], ...
     Phi,pontoonAngles,pontoons_type1,include,rayleigh); %Find modal stiffness, ...
     damping and mass
[Khg2, Chg2, Mhg2,omegaWadam2,Kh2,Ch2,Mh2] = hydroMatrices([path wadamfile_type2], ...
     Phi,pontoonAngles,pontoons_type2,include,rayleigh);

Khg=interp1z(omegaWadam1,Khg1,omega)+interp1z(omegaWadam2,Khg2,omega);
Chg=interp1z(omegaWadam1,Chg1,omega)+interp1z(omegaWadam2,Chg2,omega);
Mhg=interp1z(omegaWadam1,Mhg1,omega)+interp1z(omegaWadam2,Mhg2,omega);

elseif includeWet==false
    Khg=repmat(zeros(Nmodes),[1 1 length(omega)]);  %zero contribution from pontoons
    Chg=repmat(zeros(Nmodes),[1 1 length(omega)]);
    Mhg=repmat(zeros(Nmodes),[1 1 length(omega)]);
    omegaWadam=[0;1];
else
    error('includeWet should be logical, i.e. equal 0 or 1')
end

%% USE WHITE NOISE INSTEAD OF Sq?
if whiteNoise>0
    clear CSD;
    CSD(1:pontoonMax,1:pontoonMax,1:6,1:6,1:length(omegaLoad))=0; %allocating space
    for freqIndex=1:length(omegaLoad)
        for pontoon=whiteNoisePontoons
            CSD(pontoon,pontoon,:,:,freqIndex)=eye(6).*whiteNoise;
        end
    end
end

%% ESTABLISH Sq
Sq=zeros(6*pontoonMax,6*pontoonMax,length(omegaLoad));      %Allocating space
for k = 1:length(omegaLoad)
    for pontoon1 = 1:pontoonMax
        firstdof1=(pontoon1-1)*6+1;
        pontoon1_Sq=pontoon1;% pontoonMax-pontoon1+1;
        for pontoon2 = 1:pontoonMax
            pontoon2_Sq=pontoon2;%pontoonMax-pontoon2+1;
            firstdof2=(pontoon2-1)*6+1;

            if globalDOFs==true
```

```
                Sq(firstdof1:firstdof1+5,firstdof2:firstdof2+5,k)= ...
                    reshape(CSD(pontoon1,pontoon2,:,:,k),[6 6]);  %when load ...
                    spectra is referred to global coordinates, avoid transformation
            else
                Sq(firstdof1:firstdof1+5,firstdof2:firstdof2+5,k)= ...
                    Tmat(pontoonAngles(pontoon1),h)'*reshape(CSD(pontoon1, ...
                    pontoon2,:,:,k),[6 6])*Tmat(pontoonAngles(pontoon2),h);
            end

        end
    end
end
Sq=interp1z(omegaLoad,Sq,omega);    %Interpolate load spectra

%% FIND MODAL Sq
Sqg=zeros(Nmodes,Nmodes,length(omega));

for k=1:length(omega)
    Sqg(:,:,k)=Phi.'*Sq(:,:,k)*Phi;  %establish modal Sq
end

%% CALCULATE Sr
%Calculate Sr in generalized coordinates
Sr=zeros(pontoonMax*6,pontoonMax*6,length(omega));
Mgtot=zeros(Nmodes,Nmodes,1); Kgtot=Mgtot; Cgtot=Mgtot;

for k=1:length(omega)
    Mgtot=Mhg(:,:,k)+Msg;
    Cgtot=Chg(:,:,k)+Csg;
    Kgtot=Khg(:,:,k)+Ksg;

    Hinv=-omega(1,k)^2*Mgtot+1i*omega(1,k)*Cgtot+Kgtot;   %Inverse of transfer ...
         function, modal

    Srg=Hinv\Sqg(:,:,k)/Hinv';    %Modal response spectra
    Sr(:,:,k) = Phi*Srg*Phi.';  %Physical response spectra
end

%% CALCULATE EIGENFREUENCIES AND DAMPING RATIOS FOR COMPLEX EIGENVALUE PROBLEM

if calculateComplexeigs == 1

Mgtot=zeros(Nmodes,Nmodes,length(omega)); Kgtot=Mgtot; Cgtot=Mgtot;

    for k=1:length(omega)
       Mgtot(:,:,k)=Mhg(:,:,k)+Msg;
       Cgtot(:,:,k)=Chg(:,:,k)+Csg;
       Kgtot(:,:,k)=Ksg;
    end


    itmax=10; %maximum iterations per eigenvalue
    tol=0.001; %convergence tolerance (frequency)
    [lambda,q,wn_complex,xi_complex] = iterateFreq(Kgtot,Cgtot,Mgtot,omega, itmax,tol);

    %Check if unstable system - and create warning if so
    if ¬isempty(find(real(lambda)>1,1))
        warning('SYSTEM IS UNSTABLE - NEGATIVE DAMPING RATIO DETECTED')
    end
elseif calculateComplexeigs == 0
    wn_complex=[];
    xi_complex=[];
    q=[];
else
    error('includeComplexeigs should be logical, i.e. equal 0 or 1')

end

```

```
%% OUTPUT DATA
if exportData==1
    save('BRIDGECALCULATIONS.mat','Sr','Sq','omega','pontoonMax','wn','xi', ...
         'wn_complex','xi_complex','q','Phi')
end
```

B.2 structuralMatrices.m

```
function [ Ksg,Msg, Phi_small, wn ] = structuralMatrices(datfile,XYZpontoons)
%%STRUCTURALMATRICES Create matrices for structural stiffness, damping and
%%mass. Requires Abaqus datfile and connection coordinates matrix.
%
% INPUT:    datfile:            filename of Abaqus data file
%           connectionCoords:   matrix containing

% OUTPUT:   Khg:            generalized global stiffness matrix, after ...
     transformation and
%                           direct addition of all pontoons' contributions
%           Mhg:            generalized global mass matrix, after transformation and ...
     direct
%                           addition of all pontoons' contributions
%           Chg:            generalized global damping matrix from hydrodynamics
%           omegaWadam:     frequency values corresponding to the 3rd dimension
%                           axis of Khg, Chg and Mhg.
%
%
% Knut Andreas Kvaale (c) 2013
%

pontoonMax=size(XYZpontoons,1);
[Phi, lambda, Msg, XYZ] = dat2eig(datfile);       %Retrieve eigenvalues, ...
     eigenvectors, and nodal coordinates from dat-file

%%
Ksg=Msg.*lambda;
wn=sqrt(lambda);

[¬,Nmodes] = size(Phi);

Nsmall=pontoonMax*6;
%% FIND NODE NUMBER AND MAKE SMALL PHI-VECTOR
tolerance=0.051;
Phi_small=zeros(Nsmall,Nmodes);
for pontoon=1:pontoonMax
    for node=1:length(XYZ)
        deviance = max(abs(XYZpontoons(pontoon,:)-XYZ(node,:)));
        if deviance < tolerance
            doffirst=(node-1)*6+1;
            doffirst_new=(pontoon-1)*6+1;
            Phi_small(doffirst_new:doffirst_new+5,:)=Phi(doffirst:doffirst+5,:);
            break %the innermost for loop
        end
    end
end
```

B.3 hydroMatrices.m

```
function [Khg,Chg,Mhg,omegaWadam,Kh,Ch,Mh] = hydroMatrices(wadamfile, Phi, ...
     pontoonAngles,pontoons, include, rayleigh)
%%HYDROMATRICES Create matrices for hydrodynamic stiffness, damping and
%%mass. Requires WADAM results file.
%
% INPUT:    wadamfile:          filename of WADAM results file
%           Phi:                modal transformation matrix Phi
%           pontoonAngles:      array consisting of all angles (must correspond to ...
     Phi (same coordinate system)
%           pontoons:           which pontoons are used with this Wadamfile
%           include:            matrix with info about which parts of the matrices ...
     should be used
%                               has the following structure:
%
%                               |  K0  K(w) | (elements are logical, 1 or 0)
%                               |  C0  C(w) |
%                               |  M0  M(w) |
%
%           rayleigh:           rayleigh damping coefficients for constant terms of ...
     pontoons, [massprop. ; stiffnessprop.]
%
% OUTPUT:   Khg:                generalized global stiffness matrix, after ...
     transformation and
%                               direct addition of all pontoons' contributions
%           Mhg:                generalized global mass matrix, after transformation ...
     and direct
%                               addition of all pontoons' contributions
%           Chg:                generalized global damping matrix from hydrodynamics
%           omegaWadam:         frequency values corresponding to the 3rd dimension
%                               axis of Khg, Chg and Mhg.
%
%
% Knut Andreas Kvaale (c) 2013
%

pontoonMax=length(pontoonAngles);
include=include>0;    %make sure all elements are 0 or 1

%% IMPORT DATA FROM WADAM
[Madd,Cadd,Kh_local,T,M0] = WADAM_Import(wadamfile);
[¬,¬,freqTot] = size(Madd);
[N,Nmodes,¬] = size(Phi);

omegaWadam=2*pi./T;

%Total local mass for each pontoon (sum of const. M0 and freq. dependent Madd)
for i=1:freqTot
    Mh_local(:,:,i)=include(3,1).*M0+include(3,2).*Madd(:,:,i);   %All matrices are ...
        related to the local coordinates (middle of top plane of pontoon)
    Ch_local(:,:,i)=include(2,1).*(M0*rayleigh(1) + Kh_local*rayleigh(2))+Cadd(:,:,i);
end

%% TRANSFORM THE LOCAL MATRICES TO RESTACKED GLOBAL MATRICES
%Establish stiffness, damping and mass
    Mh=zeros(N,N,freqTot);
    Ch=zeros(N,N,freqTot);
    Kh=zeros(N,N,freqTot);
    Mhg=zeros(Nmodes,Nmodes,freqTot);

for freqNo=1:freqTot
    for pontoon=pontoons
        firstdof=(pontoon−1)*6+1;

        %TRANSFORMATION
        Kh(firstdof:firstdof+5, firstdof:firstdof+5,freqNo) = ...
             Tmat(pontoonAngles(pontoon),0)' * Kh_local * ...
             Tmat(pontoonAngles(pontoon),0);
        Ch(firstdof:firstdof+5, firstdof:firstdof+5,freqNo) = ...
             Tmat(pontoonAngles(pontoon),0)' * Ch_local(:,:,freqNo) * ...
             Tmat(pontoonAngles(pontoon),0);
        Mh(firstdof:firstdof+5, firstdof:firstdof+5,freqNo) = ...
             Tmat(pontoonAngles(pontoon),0)' * Mh_local(:,:,freqNo) * ...
             Tmat(pontoonAngles(pontoon),0);
    end

    Khg(:,:,freqNo) = Phi'*Kh(:,:,freqNo)*Phi;
    Chg(:,:,freqNo) = Phi'*Ch(:,:,freqNo)*Phi;
    Mhg(:,:,freqNo) = Phi'*Mh(:,:,freqNo)*Phi;

end

%% FLIP FREQUENCY AXIS
if omegaWadam(end)<omegaWadam(1)
    Chg=Chg(:,:,end:-1:1);
    Mhg=Mhg(:,:,end:-1:1);
    Khg=Khg(:,:,end:-1:1);
    omegaWadam=omegaWadam(end:-1:1);
end

Chg=include(2,2).*Chg;  %include C(w) if include(2,2) = 1
Khg=include(1,1).*Khg;  %include K0 if include(1,1) = 1
```

B.4 dat2eig.m

```
function [Phi, lambda, Mg, XYZ] = dat2eig(datfile)
%% DAT2EIG Extracts modal parameters from Abaqus dat-file.
%
% INPUT:    datfile:            Abaqus job filename, eg. 'Job-1.dat'
%
% OUTPUT:   Phi:                modal transformation matrix - size [MxN] where N
%                               is number of modes and M is number of DOFs in the model
%           lamda:              eigenvalues (square of eigenfrequencies)
%           Mg:                 generalized mass (modal mass)
%           XYZ:                matrix with coordinates for all nodes
%
%
% Knut Andreas Kvaale (c) 2013
%
%% RETRIEVE PHI AND COORDINATES MATRIX
fid=fopen(datfile);
S=textscan(fid,'%s','Delimiter','\n');
S=S{1};
phiTrigger='E I G E N V A L U E    N U M B E R';
phiUntrigger='';
eigTrigger='E I G E N V A L U E    O U T P U T';
eigNo=0;
phiStart=Inf;

for line=1:length(S)-1
    currentLine=S{line};

    if strncmp(currentLine, phiTrigger,34)
        eigNo=eigNo+1;
        phiStart=line;
    elseif line >= phiStart+15
        node=line-(phiStart+15)+1;
        dofFirst=(node-1)*6+1;
        numData=sscanf(currentLine(8:end),'%f');
        Phi(dofFirst:dofFirst+5,eigNo) = numData(4:end);

        if eigNo==1;     %retrieve coordinates (only do this for first mode)
            XYZ(node,:)=numData(1:3)';
        end

        if strcmp(S{line+1},phiUntrigger)       %node data is finished
            phiStart=Inf;
        end
    end

    if strncmp(currentLine,eigTrigger,34)
        eigStart=line;
    end
end

%% RETRIEVE EIGENVALUE DATA
firstLine=eigStart+6;
lambda=zeros(eigNo,1); Mg=lambda; %allocate space
for n = 1:eigNo
    line=firstLine+n-1;
    currentLine=S{line};
    numData=sscanf(currentLine(8:end),'%f');
    lambda(n)=numData(1);
    Mg(n)=numData(4);
end
```

B.5 Tmat.m

```
function Tmatrix = Tmat(alpha,h)
%%Tmat Creates transformation matrix for 6 DOF system with height transform
%%included
%
% INPUT:    alpha:              rigid body rotation angle in XY-plane (about Z-axis)
%           h:                  height used for transformation of height between ...
     sets of DOFs
%
% OUTPUT:   Tmatrix:            resulting transformation matrix
%
%
% Knut Andreas Kvaale (c) 2013
%

alpha=alpha*pi/180;     %deg --> rad

s=sin(alpha);    %to make matrix more compact below
c=cos(alpha);

Tmatrix=[c s 0 h*s -h*c 0;
    -s c 0 h*c h*s 0;
    0 0 1 0 0 0;
    0 0 0 c s 0;
    0 0 0 -s c 0;
    0 0 0 0 0 1];

end
```

B.6   iterateFreq.m

```
function [lambda,q,w,xi] = iterateFreq(Ktot,Ctot,Mtot,omega,itmax,tol)
%% ITERATEFREQ Iterates to find eigensolution for frequency dependent K, C and M.
%
% INPUT:    Ktot:               frequency dependent stiffness, 3D array ...
     (dof1,dof2,omega)
%           Ctot:               frequency dependent mass, 3D array (dof1,dof2,omega)
%           Mtot:               frequency dependent damping, 3D array (dof1,dof2,omega)
%           omega:              frequency axis (rad/s)
%           itmax:              maximum number of iterations per eigenvalue
%           tol:                tolerance for which the iteration is ended
%
% OUTPUT:   w:                  eigenfrequencies
%           xi:                 damping ratios
%           lambda:             eigenvalues
%
% Knut Andreas Kvaale (c) 2013
%

% Avoid warning of nearly singular matrix. Caution!
warning('off','MATLAB:nearlySingularMatrix');

%Initial conditions
[nmax,¬,¬] = size(Ktot);  %number of DOFs

%Main calculations
lambda=zeros(2*nmax,1);
for n = 1:2:2*nmax  %skip the conjugates (but save them, see below)
    w=0;    %initial condition, w0 (iteration 0)
    wlast=Inf;
    wlastlast=—Inf;

    for i = 1:itmax
        [¬,index] = min(abs(omega — w));    %find the index in omega for our value w

        [thisLambda,thisq] = ...
             eigSolve(Ktot(:,:,index),Ctot(:,:,index),Mtot(:,:,index),0,1);    ...
             %solve eigenvalue problem for this frequency input
        thisLambda=sort(thisLambda);
        [thisLambda,ix]=sort(thisLambda);
%         thisq=thisq(:,ix);  %sort vectors according to sort of eigenvalues

        w=abs(thisLambda(n));

        if abs(w — wlastlast) < tol && abs(w — wlast) > tol      %diverged
            warning('Divergence found!')
        elseif abs(w — wlastlast) < tol && abs(w — wlast) < tol    %converged (3 ...
             last entries equal)
            break
        end

        wlastlast=wlast;
        wlast=w;
    end
        lambda(n) = thisLambda(n);
        lambda(n+1) = conj(thisLambda(n));
        q(:,n) = thisq(:,n);
        q(:,n+1) = conj(thisq(:,n));
end

w=unique(abs(lambda));
xi=unique(—real(lambda)./(abs(lambda)));

%Turn warning back on.
```

```
warning('on','MATLAB:nearlySingularMatrix');

end
```

B.7 eigSolve.m

```
function [lambda,q] = eigSolve(K,C,M,dispWarnings,sortOutput)
%% EIGSOLVE Solve complex eigenvalue problem in space state form.
%
% INPUT:    K:                  stiffness matrix (NxN), N is number of DOFs
%           C:                  damping matrix (NxN), N is number of DOFs
%           M:                  mass matrix (NxN), N is number of DOFs
%           dispWarnings:       boolean input, true if function warnings are
%                               wanted
%           sortOutput:         boolean input, true if sorting based on
%                               absolute value of eigenvalues are wanted
%
% OUTPUT:   lambda:             resulting eigenvalues
%           q:                  resulting eigenvectors
%
%
% Knut Andreas Kvaale (c) 2013
%


%% ALLOCATE SPACE AND FIND INITIAL SIZES
omega=0; xi=0;
N=length(K);

%% DEFINE THE SYSTEM MATRIX FOR STATE SPACE EQUATION
A(1:N,1:N) = zeros(N);
A(1:N,N+1:2*N) = eye(N);
A(N+1:2*N,1:N) = -M\K;
A(N+1:2*N,N+1:2*N) = -M\C;

% Q=[zeros(N,1); M\P]; % load vector for state space system

%% SOLVE EIGENVALUE PROBLEM
[q, lambda] = eig(A);
lambda=diag(lambda);

%q_i=[phi_i; phi_i * lambda]
q=q(1:N,:);

%% FIND POSITIVE REAL VALUED EIGENVALUES
tol=1E-10;   %max positive value ≠ negative
ncl=find(real(lambda) > tol);

%% WARNINGS
if dispWarnings==true
    clf
    close all
    if sum(abs(real(lambda)))>tol
        errorMsg = sprintf(['Complex eigenvalues and eigenvectors!']);
        uiwait(warndlg(errorMsg));
    end

    if size(ncl) ≥ 1
        errorMsg = sprintf(['Warning: ' num2str(length(ncl)) ' positive real valued ...
            eigenvalues detected! Indicates unstable system.']);
        uiwait(warndlg(errorMsg));
    end

    if size(omega) > N
        errorMsg = sprintf('Warning: Number of distinct natural frequencies exceeds ...
             the dimensions of the system.');
        uiwait(warndlg(errorMsg));
    end
end

%% SORT OUTPUT
if sortOutput == true
    [lambda,ix]=sort(lambda);
    q=q(:,ix);  %sort vectors according to sort of eigenvalues
end
```

B.8 plotModes.m

```
function [figureHandle] = plotModes(Phi,XYZ0,modes,scaling,viewAxis)
%%PLOTMODES Create plots of mode shapes for given coordinates and Phi—matrix.
%
% INPUT:    Phi:          modal transformation matrix
%           XYZ0:         coordinates of original nodes [X1 Y1 Z1; X2 Y2..]
%           modes:        which mode to plot
%           scaling:      scaling factor of modal vectors, if 0 is chosen
%                         automatic scaling is enforced
%
% OUTPUT:   figureHandle:   handle for figure created
%
% Knut Andreas Kvaale (c) 2013
%

%% EXTRACT TRANSLATIONAL DISPLACEMENTS
%Assume Phi is for 6 DOFs, extract only translational DOFs
Phi=real(Phi);  %make sure only real part is considered
dX=Phi(1:6:end,modes);
dY=Phi(2:6:end,modes);
dZ=Phi(3:6:end,modes);

%% SCALING OF MODES
if scaling==0   %AUTO SCALING
    k=0.2;
    scaling=k.*norm([max(XYZ0(:,1)) — min(XYZ0(:,1)), max(XYZ0(:,2))—min(XYZ0(:,2)), ...
         max(XYZ0(:,3))—min(XYZ0(:,3))])/norm([max(abs(dX)), max(abs(dY)), ...
         max(abs(dZ))]);
end

%% PLOTTING
    figureHandle = figure;
    X0=XYZ0(:,1); Y0=XYZ0(:,2); Z0=XYZ0(:,3);
    C=[0 0 1; 1 0 0; 0 0.8 0; 0.6 0 1; 0.5 0.5 0];

if viewAxis == 0
    %UNDEFORMED PLOT
    set(0,'CurrentFigure',figureHandle)
    line(X0,Y0,Z0,'color','black'), hold on
    scatter3(X0,Y0,Z0,36,[0 0 0],'fill')

    for k = 1:length(modes)
        set(0,'CurrentFigure',figureHandle)
        X=X0+dX(:,k).*scaling;
        Y=Y0+dY(:,k).*scaling;
```

```
            Z=Z0+dZ(:,k).*scaling;
            quiver3(X0,Y0,Z0, X-X0,Y-Y0,Z-Z0,0,'Color',C(k,:))
            plot3(X,Y,Z,'Color',C(k,:))

        end
        set(0,'CurrentFigure',figureHandle)
        axis equal
        xlabel('x')
        ylabel('y')
        zlabel('z')

    elseif viewAxis ≠0
        %UNDEFORMED PLOT
        set(0,'CurrentFigure',figureHandle)
        if viewAxis==1
               line(Y0,Z0,'color','black'), hold on
               scatter(Y0,Z0,36,[0 0 0],'fill')
        elseif viewAxis==2
               line(X0,Z0,'color','black'), hold on
               scatter(X0,Z0,36,[0 0 0],'fill')
        elseif viewAxis==3
               line(X0,Y0,'color','black'), hold on
               scatter(X0,Y0,36,[0 0 0],'fill')
        end

        for k = 1:length(modes)
            set(0,'CurrentFigure',figureHandle)
            X=X0+dX(:,k).*scaling;
            Y=Y0+dY(:,k).*scaling;
            Z=Z0+dZ(:,k).*scaling;

            if viewAxis==1
                quiver(Y0,Z0,Y-Y0,Z-Z0,0,'Color',C(k,:))
                plot(Y,Z,'Color',C(k,:))
                xlabel('y')
                ylabel('z')
            elseif viewAxis==2
                quiver(X0,Z0,X-X0,Z-Z0,0,'Color',C(k,:))
                plot(X,Z,'Color',C(k,:))
                xlabel('x')
                ylabel('z')
            elseif viewAxis==3
                quiver(X0,Y0,X-X0,Y-Y0,0,'Color',C(k,:))
                plot(X,Y,'Color',C(k,:))
                xlabel('x')
                ylabel('y')
            end

        end
        set(0,'CurrentFigure',figureHandle)
        axis equal


    end
```

B.9 pontoonGeometry.m

```
function varargout = pontoonGeometry(varargin)
% PONTOONGEOMETRY MATLAB code for pontoonGeometry.fig
%      PONTOONGEOMETRY, by itself, creates a new PONTOONGEOMETRY or raises the existing
%      singleton*.
```

```
%
%      H = PONTOONGEOMETRY returns the handle to a new PONTOONGEOMETRY or the handle to
%      the existing singleton*.
%
%      PONTOONGEOMETRY('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PONTOONGEOMETRY.M with the given input arguments.
%
%      PONTOONGEOMETRY('Property','Value',...) creates a new PONTOONGEOMETRY or ...
    raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pontoonGeometry_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pontoonGeometry_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help pontoonGeometry

% Last Modified by GUIDE v2.5 05-May-2013 13:49:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pontoonGeometry_OpeningFcn, ...
                   'gui_OutputFcn',  @pontoonGeometry_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before pontoonGeometry is made visible.
function pontoonGeometry_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pontoonGeometry (see VARARGIN)

% Choose default command line output for pontoonGeometry
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
gatherAndUpdate(handles)

% UIWAIT makes pontoonGeometry wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = pontoonGeometry_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% ——— Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%SAVE
[filename path] = uiputfile('*.txt','Save geometry data');

if filename== 0
    return
end

dataMatrix=get(handles.uitable2,'Data');
dataExport=dataset({dataMatrix,'PONTOON' 'X' 'Y' 'Z' 'ANGLE'});

export(dataExport,'file',[path filename],'delim','\t')
dataExport

% ——— Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%LOAD

[filename path] = uigetfile('*.txt','Load geometry data');
if filename== 0
    return
end

fid=fopen([path filename]);
[¬]=fgetl(fid); %skip first line

while true
   tline=fgetl(fid);

   if ¬ischar(tline)
       break
   end
   tline=str2num(tline);
   pontoon=tline(1);
   pontoonAngles(pontoon,:) = tline(5);
   connectionCoords(pontoon,:) = tline(2:4);
   pontoons(pontoon)=pontoon;
end

dataMatrix=[pontoons' connectionCoords pontoonAngles];
set(handles.uitable2,'Data',dataMatrix);
gatherAndUpdate(handles)

% ——— Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
dataMatrix=get(handles.uitable2,'Data');
n=max([max(dataMatrix(:,1)) size(dataMatrix(:,1),1)]);

dataMatrix=[dataMatrix; n+1 0 0 0 0];
set(handles.uitable2,'Data',dataMatrix);
gatherAndUpdate(handles)

function gatherAndUpdate(handles)
dataMatrix=get(handles.uitable2,'Data');
ellipseDef=get(handles.uitable5,'Data');
```

```

plotOnePontoon(ellipseDef,handles)
plotPontoons(dataMatrix,ellipseDef,handles)


function plotSq(angleInput,handles)
angle=angleInput*pi/180;

cla(handles.axes6);
hold(handles.axes6,'on');

T = [cos(angle) -sin(angle);  sin(angle)  cos(angle)];
xvec=T(:,1);
yvec=T(:,2);

h3=quiver(handles.axes6, 0, 0, xvec(1),xvec(2),1,'color','blue');
h4=quiver(handles.axes6, 0, 0, yvec(1),yvec(2),1,'color','red');

legend([h3 h4], 'x_{spectra}','y_{spectra}')

axis(handles.axes6,'equal');
xlabel(handles.axes6,'x');
ylabel(handles.axes6,'y');


function plotOnePontoon(ellipseDef,handles)
cla(handles.axes7);
hold(handles.axes7,'on')
a=ellipseDef(1);
b=ellipseDef(2);

T=drawEllipse(0,0,a,b,0,100,handles.axes7);
h1=quiver(handles.axes7,0, 0, 1,0,80,'color','blue');
h2=quiver(handles.axes7,0, 0, 0,1,80,'color','red');
axis(handles.axes7,'equal');
legend([h1 h2],'x_{pontoon}', 'y_{pontoon}')
xlabel(handles.axes7,'x');
ylabel(handles.axes7,'y');

function plotPontoons(dataMatrix,ellipseDef,handles)
a=ellipseDef(1);
b=ellipseDef(2);

[n,¬] = size(dataMatrix);
connectionCoords=dataMatrix(:,2:4);
pontoonAngles=dataMatrix(:,5);
cla(handles.axes2);
hold(handles.axes2,'on');

for i = 1:n
    T=drawEllipse(connectionCoords(i,1),connectionCoords(i,2),a,b, ...
         pontoonAngles(i),100,handles.axes2);
    xvec=T(1:2,1);
    yvec=T(1:2,2);
    h1=quiver(handles.axes2,connectionCoords(i,1), connectionCoords(i,2), ...
         xvec(1),xvec(2),80,'color','blue');
    h2=quiver(handles.axes2,connectionCoords(i,1), connectionCoords(i,2), ...
         yvec(1),yvec(2),80,'color','red');
end

legend([h1 h2],'x_{pontoon}', 'y_{pontoon}')
axis(handles.axes2,'equal')
xlabel(handles.axes2,'x')
ylabel(handles.axes2,'y')


function [T] = drawEllipse(x0,y0,a,b,angle,res,inputhandle)
angle = angle.*pi/180;
```

```matlab
theta = linspace(0,2*pi,res);
s(1,:) = a.*cos(theta);
s(2,:) = b.*sin(theta);

T = [cos(angle) -sin(angle);  sin(angle)  cos(angle)];
s = T*s;

s(1,:) = s(1,:) + x0;
s(2,:) = s(2,:) + y0;

outputHandle=plot(inputhandle,s(1,:),s(2,:),'LineWidth',1,'color','black');

% --- Executes when entered data in editable cell(s) in uitable2.
function uitable2_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to uitable2 (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty if Data ...
    was not changed
%   Error: error string when failed to convert EditData to appropriate value for Data
% handles    structure with handles and user data (see GUIDATA)
gatherAndUpdate(handles)


% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.uitable2,'Data',[1 0 0 0 0]);
gatherAndUpdate(handles)


% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%delete row

if ¬isempty(handles.rowNo)
    row=handles.rowNo;
    dataMatrix=get(handles.uitable2,'Data');
    if row≤size(dataMatrix,1)
        dataMatrix(row,:)=[];
        set(handles.uitable2,'Data',dataMatrix);
    end
end

gatherAndUpdate(handles)

% --- Executes when selected cell(s) is changed in uitable2.
function uitable2_CellSelectionCallback(hObject, eventdata, handles)
% hObject    handle to uitable2 (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) currently selecteds
% handles    structure with handles and user data (see GUIDATA)
if ¬isempty(eventdata.Indices)
    rowInfo = guidata(hObject);
    rowInfo.rowNo=eventdata.Indices(1);
    guidata(hObject,rowInfo);
end
gatherAndUpdate(handles)

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved — to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
gatherAndUpdate(handles)


% ——— Executes when entered data in editable cell(s) in uitable5.
function uitable5_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to uitable5 (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty if Data ...
     was not changed
%   Error: error string when failed to convert EditData to appropriate value for Data
% handles    structure with handles and user data (see GUIDATA)
gatherAndUpdate(handles)


% ——— Executes when selected cell(s) is changed in uitable5.
function uitable5_CellSelectionCallback(hObject, eventdata, handles)
% hObject    handle to uitable5 (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) currently selecteds
% handles    structure with handles and user data (see GUIDATA)
gatherAndUpdate(handles)
```

B.10 shearframe.m

```
%Initializing commands
clc
format long
clear all
clf

%Initial vectors and values
k=6; m=2; c=2;
omega=0:0.01:5;
kmax=length(omega);

%System matrices
K=[2*k -k;-k k]; %Consistent stiffness
M=[m 0; 0 m ]; %Lumped mass
C=[c 0; 0 c];

%JONSWAP wave spectrum (load assumed only along/in DOF no #1)
g=9.81;
sigma=0.07;
alpha=1.5;      %Randomly chosen (other values from page 445 in "Dynamisk analyse av ...
     konstruksjoner" by Langen and Sigbjornsson
beta=1.25;
gamma=7.0;      %Spreading parameter (more spread out if gamma is low), 7 from ...
     Langen&Sigbjornsson + an introduction to random vibrations
wp=1.5;           %Frequency (rad/s) for peak value

Sq0=JONSWAP(alpha,beta,gamma,sigma,wp,omega);
Sq=zeros(2,2,length(omega));

```

```
Sq(1,1,:)=reshape(Sq0,[ 1 1 length(omega)]);


%Frequency dependent transfer function
N=length(K);
Sr=zeros(N,N,kmax); H=Sq; %Allocating memory
for k=1:kmax
    w=omega(k);
    H(:,:,k)=inv(K − w.^2.*M + 1i.*w.*C);               %Transfer function
    Sr(:,:,k) = H(:,:,k)*Sq(:,:,k)*H(:,:,k)';     %Response variance spectrum
end

%Plotting
% figHandle=figure(1);
% incr=50;
% clf



for i = 1:N
    for j=1:N
        datano=j+N*(i−1);
         subplot(N,N,datano)
        x=omega';
%          figure(plotno)

        y1=(squeeze(Sr(i,j,:)));
        y2=(squeeze(Sq(i,j,:)));

         plot(x,real(y1),'blue'); hold on;
         plot(x,imag(y1),'red')
%
%          title(['S_{' num2str(i) ',' num2str(j) '}'])
%          xlabel('\omega [s^{−1}]')
%          set(axre(2),'YColor','red');
%          set(hreal2,'color','red')

    X(:,datano)=x;
    Y(:,datano)=imag(y1);
    end
end

titles={'$S_{11}$' '$S_{12}$' '$S_{21}$' '$S_{22}$'};
xlabels={'$\omega [s^{−1}]$'};
ylabels={''};
% makegrouptikz(X,Y,2,2,titles,xlabels,ylabels,'Sr_shearframe_free_Im.tikz')

y=squeeze(Sq(1,1,:));
x=omega';

disp('Eigenfrequencies:')
disp(num2str(unique(abs(eigSolve(K,C,M,0)))))
```

B.11 interp1z.m

```
function [ Mi ] = interp1z(z,M,zi)
%%interp1z Interpolates input data in similar fashion to interp1, but along z−axis ...
     (3. dimension)
%
% INPUT:    z:                   original z−axis
%           M:                   original matrix
%            zi:                  interpolated z—axis
%
% OUTPUT:    Mi:                  interpolated matrix
%
%
% Knut Andreas Kvaale (c) 2013
%

[Lx,Ly,Lz] = size(M);
Lzi=length(zi);

Mmod=reshape(M,[],Lz).';
Mmodi = interp1(z,Mmod,zi,'linear','extrap');
Mi=reshape(Mmodi.',Lx,Ly,Lzi);

end
```

C | Comprehensive plots

Only small parts of the total results are presented in the report. Some comprehensive plots are therefore given below.

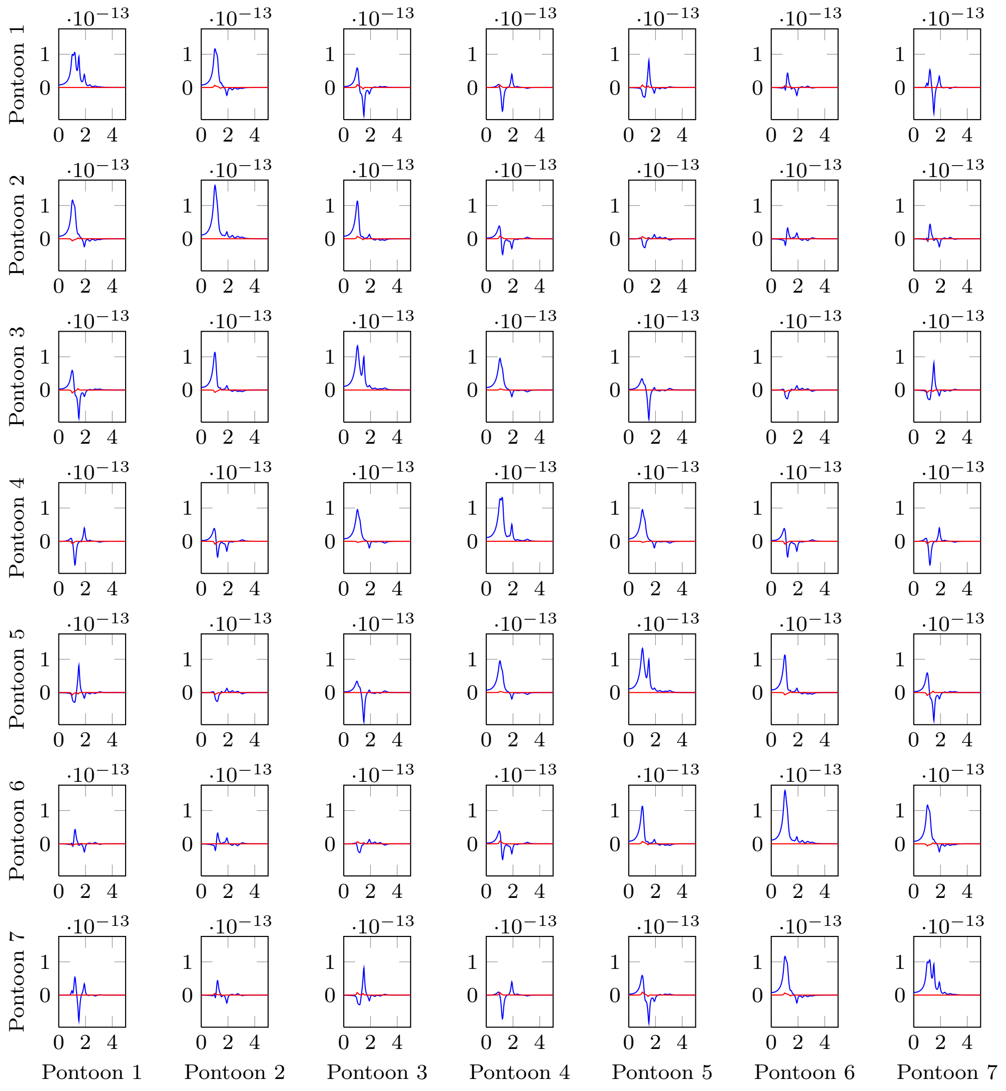


Figure C.1: *Cross-response spectra for the heave (DOF 3) of all pontoons on the Bergsøysund bridge. White noise loading with unit amplitude is used here. The x-axis shows the frequency, in rad/s, and the y-axis shows the cross-spectral density between the displacements, in* m^2*.*

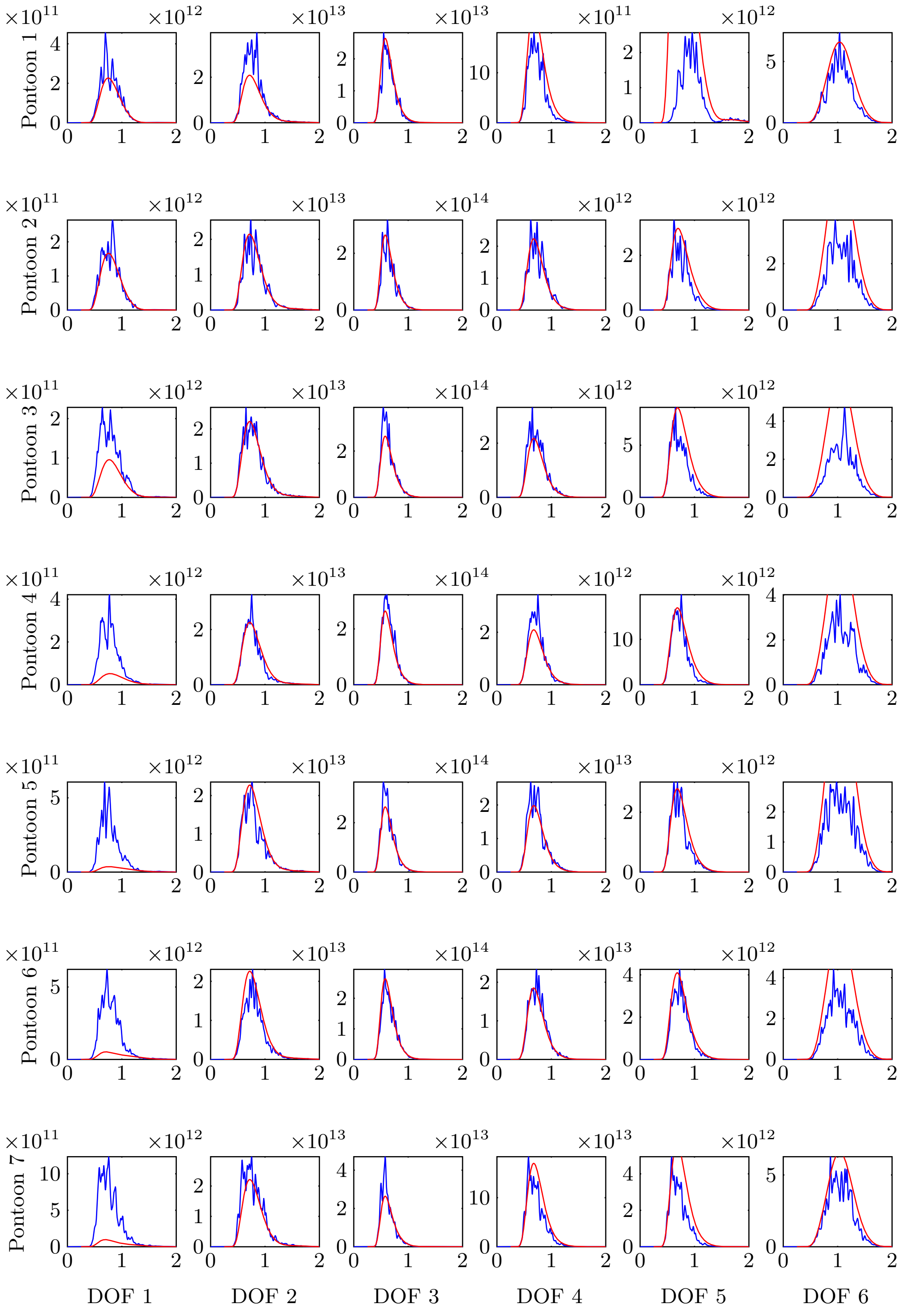


Figure C.2: *Auto-covariance spectra of load used as input. Blue plots are based on time series of the estimated load, while red plots are load spectra used in the frequency domain.*