Modal analysis techniques for (damped) floors

Explained and tested

Robert Giesen

Document name: Internship Report - Robert Giesen.docx
Date: 12/11/2017

Commissioned by:
MECAL Pantheon BV
Capitool 15
Enschede
The Netherlands

Modal analysis techniques for (damped) floors

Explained and tested

Document name:	Internship Report - Robert Giesen.docx
Date:	12/11/2017
Written by:	Robert Giesen, MECAL Pantheon BV, Enschede, The Netherlands
Reviewed by:	Servaas Bank, MECAL Pantheon BV, Enschede, The Netherlands
Approved by:	Servaas Bank, MECAL Pantheon BV, Enschede, The Netherlands
Document by:	MECAL Pantheon BV, Enschede, The Netherlands
Commissioned by:	MECAL Pantheon BV
Mailing list:	MECAL (1x), MECAL Pantheon BV (1x)
Pages incl. cover:	107
Keywords:	Modal analysis, Quadrature picking, Curve fitting
Language	English
Software version	Microsoft Word 2017

Contributing Authors	
Name Author	**Part(s) of this document**
Robert Giesen	All

Document change and history record			
Version	**Date**	**Page (s)**	**Short description**
1.0	12/11/2017	107	First Issue

Abstract

Modal description is a tool to describe the dynamic behaviour of a structure. Modal parameters can be found from a (simplified) theoretical spring-mass model or by measurements taken from the structure, in combination with certain fitting techniques.

The modal analysis procedure for the measurement and analysis of vibrations in a damped structure can be split into two complementing parts: the theoretical and the experimental analysis. The theoretical modal analysis makes a prediction of the system based on a simplified representation of the structure, while the experimental modal analysis determines the system by means of measurements and modal parameter estimation of the unknown data. Together they provide an accurate modal description of the structure, showing the natural frequencies at which resonance occurs and the shapes of deformation during vibrations. The methods derived during this research where checked using measurement data of real-life situations.

Three modal parameter estimation methods where attempted during this research. It was observed that the applicability of these methods, with regard to the accuracy, is heavily situation-dependent. The structure that is being measured and the desired results of the modal analysis must be assessed, after which a method must be chosen based on the assessment.

Table of contents

Chapter 1 Introduction 11
1.1 Assignment description 11
1.2 Research questions 11
1.3 Outline of the report 12
Chapter 2 Results 13
2.1 Roadmap 13
2.2 SEMICON FAB-X 15
2.3 SEMICON FAB-Y 18
2.4 SEMICON FAB-Z 21
Chapter 3 Conclusion 24
Chapter 4 Literature review 27
4.1 Passive vibration isolation 27
4.2 Active vibration cancellation 29
4.3 Modal analysis methods 29
4.3.1 Important considerations 31
4.4 Accurate measurements 34
Chapter 5 Theoretical modal analysis 37
5.1 Kinetics: Force and moment balances 37
5.2 Kinematics: Equations of motion 37
5.3 Vibration eigenproblem: Natural frequencies and modes 38
5.4 Frequency response function models 43
5.5 Single mode contribution 48
Chapter 6 Experimental modal analysis 51
6.1 Method 51
6.2 Conditions 51
Chapter 7 Quadrature picking 53
7.1 Frequency response functions matrix 54
7.2 Peak picking: Natural frequencies 55
7.2.1 Identification of the natural frequencies 56
7.2.2 Mode indicator functions 57
7.3 Quadrature picking: Natural modes and damping 58
7.3.1 Natural modes 58
7.3.2 Damping matrix 59

Chapter 8 Modal parameter estimation 63
8.1 Understanding the shape of the modal matrix 64
8.2 Basic model curve fitting 65
8.3 Single mode contribution scaling 67
8.4 Mean square error curve fitting 68
8.4.1 Scaling of the magnitude 70
8.4.2 Residual effects 70
8.4.3 MATLAB Optimisation 70
8.5 Other methods 71
Chapter 9 Error of the modal analysis 73
Chapter 10 Discussion and recommendations 79
Chapter 11 References 80
Appendix A MATLAB function EoM 81
Appendix B MATLAB function FRF 82
Appendix C MATLAB function ModeShapes 83
Appendix D MATLAB function GatherData 84
Appendix E MATLAB function HAvgAndCoh 86
Appendix F MATLAB function fft_cspectrum 87
Appendix G MATLAB function ModalFRF 88
Appendix H MATLAB function QuadraturePicking 90
Appendix I MATLAB function QuadraturePicking2D 93
Appendix J MATLAB script Curve fitting – Single mode contribution scaling 96
Appendix K MATLAB script Curve fitting – mean square error optimisation 97
Appendix L MATLAB script Compensation residues 99
Appendix M MATLAB GUI - QuadraturePicking 100

List of figures

Figure 2.1: Modal analysis roadmap. 14
Figure 2.2: Experimental modal analysis setup for SEMICON FAB-X. 15
Figure 2.3: Identification of the natural frequencies of SEMICON FAB-X. 15
Figure 2.4: Mode shapes for SEMICON FAB-X. 15
Figure 2.5: Frequency response functions and the corresponding curve fit for SEMICON FAB-X. 16
Figure 2.6: Driving point frequency response functions and the corresponding prediction for SEMICON FAB-X. 17
Figure 2.7: Experimental modal analysis setup for SEMICON FAB-Y. 18
Figure 2.8: Identification of the natural frequencies of SEMICON FAB-Y. 18
Figure 2.9: Mode shapes for SEMICON FAB-Y. 18
Figure 2.10: Frequency response functions and the corresponding curve fit for SEMICON FAB-Y. 19
Figure 2.11: Driving point frequency response functions and the corresponding prediction for SEMICON FAB-Y. 20
Figure 2.12: Experimental modal analysis setup for SEMICON FAB-Z. 21
Figure 2.13: Identification of the natural frequencies of SEMICON FAB-Z. 21
Figure 2.14: Mode shapes for SEMICON FAB-Z. 21
Figure 2.15: Frequency response functions and the corresponding curve fit for SEMICON FAB-Z. 22
Figure 2.16: Frequency response functions and the corresponding prediction for SEMICON FAB-Z. 23
Figure 4.1: Viscously damped mass-spring system subjected to external excitation [3]. 27
Figure 4.2: Dynamic vibration absorber [3]. 27
Figure 4.3: Magnitude of the response of the main system for three values of the damping coefficient. 28
Figure 4.4: Excitation of a beam under bending, mode shapes and the imaginary part of the frequency response functions for the complete system [2]. 30
Figure 4.5: Mode shapes of a beam excited on bending, derived from the middle and bottom rows of the frequency response function matrix [2]. 31
Figure 4.6: Frequency response functions (black), input power spectra (blue) and coherence spectra (red) for different hammer tips. 33
Figure 4.7: Aliasing. 35
Figure 5.1: Two-body model of a floor, problem definition. 37
Figure 5.2: Free-body-diagram floor model. 37
Figure 5.3: Mode shapes of the two-body system, derived from the natural modes. 42
Figure 5.4: Mode shapes of the two-body system, calculated by MATLAB. 42
Figure 5.5: Frequency response functions of the two-body system. 44
Figure 5.6: Modal frequency response functions of the two-body system. 49
Figure 5.7: Single mode contributions for the frequency response functions of the two-body system. 50
Figure 6.1: Roving hammer experiment [2]. 51
Figure 6.2: Roving sensor experiment [2]. 52
Figure 6.3: Impact hammer (left), acceleration sensor (middle), steel block with spikes (right). 52
Figure 7.1: Four-body model of a floor, problem definition. 53
Figure 7.2: Frequency response functions of the four-body system. 53
Figure 7.3: A floor divided into 36 measurement positions. 54
Figure 7.4: Visibility of the resonance peaks for the lightly damped (left) and heavily damped (right) cases of the four-body model. 55
Figure 7.5: Mode indicator functions for the lightly damped (left) and heavily damped (right) cases of the four-body model. 56
Figure 7.6: Ordinary mode indicator functions for the lightly damped (left) and heavily damped (right) cases of the four-body model. 57
Figure 7.7: Quadrature picking. 58

Figure 7.8: Mode shapes of the four-body model. 60
Figure 7.8: Quadrature picking the natural modes of the first two bodies of the four-body model. 61
Figure 7.9: Waterfall plot showing the contribution of each frequency response function to the natural modes of the first two bodies of the four-body model. 62
Figure 8.1: Regenerated frequency response functions from the modal description for the first row of the four-body model. 63
Figure 8.2: Mode shapes of a standard surface model. 64
Figure 8.3: MATLAB curve fitting tool (cftool) – Modal frequency response function fit for the first FRF of the four-body model. 66
Figure 8.4: MATLAB curve fitting tool (cftool) – System parameter frequency response function fit for the first FRF of the four-body model. 66
Figure 8.5: Regenerated frequency response functions from the scaled single mode contributions for the first row of the four-body model. 67
Figure 8.6: Regenerated frequency response functions from the mean square error curve fitting for the first row of the four-body model. 68
Figure 8.7: Regenerated frequency response functions from the mean square error curve fitting for the first row of the four-body model, with slightly different mass for each body. 69
Figure 8.8: Simple Exponential function-fit estimated by fmincon. 70
Figure 8.9: Other methods available for the experimental modal analysis of a system [21]. 72
Figure 9.1: lightly-damped single-body model, problem definition. 73
Figure 9.2: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the lightly-damped single-body model. 73
Figure 9.3: Extreme-damped single-body model, problem definition. 73
Figure 9.4: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the extreme-damped single-body model. 74
Figure 9.5: lightly-damped two-body model, problem definition. 74
Figure 9.6: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the lightly-damped two-body model. 74
Figure 9.7: Frequency response functions and the corresponding predictions for the lightly-damped two-body model. 75
Figure 9.8: Heavily-damped two-body model, problem definition. 75
Figure 9.9: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the heavily-damped two-body model. 76
Figure 9.10: Frequency response functions and the corresponding predictions for the heavily-damped two-body model. 76
Figure 9.11: Lightly-damped four-body model, problem definition. 76
Figure 9.12: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the lightly-damped four-body model. 77
Figure 9.13: Frequency response functions and the corresponding predictions for the lightly-damped four-body model. 77
Figure 9.14: Slightly-more-damped four-body model, problem definition. 77
Figure 9.15: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the slightly-more-damped four-body model. 78
Figure 11.1: Preview of the quadrature picking GUI. 100

Table 7.1: Frequency response functions matrix of the compliance. 54
Table 8.1: Difference in modal matrices derived from the equations of motion and by means of quadrature picking. 64
Table 11.1: MATLAB function EoM.m. 81
Table 11.2: MATLAB function FRF.m. 82
Table 11.3: MATLAB function ModeShapes.m. 83
Table 11.4: MATLAB function GatherData.m. 85
Table 11.5: MATLAB function HAvgAndCoh.m 86
Table 11.6: MATLAB function fft_cspectrum.m. 87
Table 11.7: MATLAB function ModalFRF.m 89
Table 11.8: MATLAB function QuadraturePicking.m. 92
Table 11.9: MATLAB function QuadraturePicking2D.m. 95
Table 11.10: MATLAB script Curve fitting – Single mode contribution scaling. 96
Table 11.11: MATLAB script Curve fitting – mean square error optimisation. 97
Table 11.12: MATLAB script Curve fitting – Optimisation function 98
Table 11.13: MATLAB script Compensation residues. 99
Table 11.14: MATLAB GUI – QuadraturePicking. 107

List of symbols

$z, \dot{z}, \ddot{z}$	Cartesian coordinates
$z_m, \dot{z}_m, \ddot{z}_m$	Natural coordinates
$q, \dot{q}, \ddot{q}$	Generalised coordinates
α	Angular acceleration
s	Laplace parameter
t	Time
$\boldsymbol{M}, m$	Mass matrix, Mass
$\boldsymbol{C}, c$	Damping matrix, Damping coefficient
$\boldsymbol{K}, k$	Stiffness matrix, Spring constant
$\boldsymbol{B}(s)$	System matrix
$\boldsymbol{D}(\omega)$	Dynamic matrix
$F(t)$	Applied force
$M(t)$	Applied moment
$Q(t)$	Modal force
F_c, F_k, F_t	Viscous damping force, Spring force, Transmitted force
$\frlap{\lvert F_t \rvert}{F}$	Transmissibility
δT	Virtual kinetic energy
δV	Virtual potential energy
δU, δW	Virtual work
$\mathcal{F}$	Rayleigh's dissipation function
ω	Excitation frequency
ω_n	Natural frequency
ξ	Damping ratio
$\boldsymbol{M}_m$	Modal mass matrix
$\boldsymbol{C}_m$	Modal damping matrix
$\boldsymbol{K}_m$	Eigenvalue matrix/modal stiffness matrix
$\boldsymbol{V}, \boldsymbol{V}_n$	Modal matrix, Mode vector
h	Frequency response function
$\boldsymbol{H}, \boldsymbol{H}_{Avg}, \boldsymbol{H}_{Im/Re}$	Frequency response functions matrix, Average frequency response functions matrix, Imaginary over real part of the frequency response functions matrix
A	Modal scaling constant
MIF	Mode indicator function
i	Imaginary number
$\lvert \dots \rvert$, $\lVert \dots \rVert$, $\overline{\dots}$	Absolute value, Determinant, Complex conjugate
$\sum \dots, \Delta \dots$	Summation, Difference
$\widehat{\dots}$	Normalised vector

Preface and acknowledgements

This report is commissioned by MECAL Pantheon BV in the Netherlands, as a conclusion of my fourteen weeks' internship from September 4, 2017 until December 11, 2017. During the second year of the master Mechanical Engineering (Mechanics of Solids, Surfaces and Systems) at the University of Twente, an internship at an external company must be conducted. During this internship, the student learns how to function independently in a real-life situation at graduate level and to apply and increase the knowledge and competences gained during the master program.

The courses in my master program are focused mainly on mechatronics and it is therefore that MECAL is the perfect employer for my internship.

About MECAL Pantheon BV

MECAL is a company founded in 1989 focused on analysis and product development in the wind energy and the high-tech systems. The company designs customised solutions for machine pedestals in semiconductor production facilities, complying with the ISO standards: ISO 9001, ISO 14001 and ISO 18001.

Acknowledgements

The internship opportunity I had with MECAL was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual that I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me through this internship period.

I express my deepest gratitude and special thanks to Servaas Bank, technical specialist who despite being extraordinarily busy with his duties, took time out to hear and guide me during the research and allowing me to carry out my project.

Chapter 1 Introduction

1.1 Assignment description

MECAL High-tech/Systems, designs and produces solutions for vibration problems in semiconductor fabrication plants. Some products are: stiff machine support frames or pedestals, structural building improvements, active vibration isolation pedestals and the EQUALIZER [1], an active vibration cancellation system which can generate a force that counteracts vibrations on a floor field.

For the best application of the EQUALIZER, the system must be measured in the form of frequency response functions. The frequency response functions are measured from one point on the structure to another, where an excitation force is applied to the system and the amplitude of the response measured. Division of the excitation force by the response results in the stiffness, while division of the response by the excitation force results in the compliance of the system.
These measurements are time-consuming and it is therefore worthwhile to derive a method, capable of making predictions for the frequency response functions of a part of the system.

The goal of this research is to be able to make a prediction for the positioning and the effectiveness of the Equalizer from MECAL. To be able to make such a prediction, the frequency response functions of the factory floor must be measured and analysed. There are three ways this can be done:

- Calculation of the modal description of the system from the experimental modal analysis.
- Tuning of a simple finite element model, according to the measurement data.
- Measuring the required frequency response functions directly.

The experimental modal analysis results in measurement data of the frequency response functions, from which the modal description of the system can be derived. This modal description gives the dynamic behaviour of the system, in a similar way as the equations of motion. While in theory, this gives accurate predictions. It must however be possible to derive the modal description from the measurement data.
The tuning of a simple finite element model is on the other hand a more straightforward method, where a standard frequency response function will be fitted to the measurement data.
As a last resort, the frequency response functions can be measured directly, giving the most accurate results, but also being the most time-consuming.

These methods will be investigated here and recommendations will be derived on when to use which method.

1.2 Research questions

The research questions are divided into main questions and sub-questions:

- Is it possible to use only the data from a single roving hammer or roving sensor experiment, to derive the modal description of the systems?
  - What are the steps necessary to perform a roving hammer/sensor experiment?
  - Which measurement data results from the roving hammer/sensor experiment?
  - Which methods can be used to derive the modal description of the system using measurement data?

- Can the data from a single roving hammer or roving sensor experiment be used to make an accurate prediction of the complete vibrational behaviour of the system?
  - Which methods can be used to make a prediction for the modal parameters of the system?
  - Which system parameters influence the accuracy of the prediction?
  - How do additional measurements influence the accuracy of the prediction?
  - How does damping influence the accuracy of the prediction?

- Which method should be used in a certain situation, such that the accuracy of the prediction is as high as possible?
  - To what extent can the methods be used, such that the accuracy is still within reasonable bounds?

1.3 Outline of the report

Because of the difficulty of the modal analysis and the involved formulas, the results and conclusions will be reported first. These first chapters will serve as a common thread through the derivations in Chapter 5 to Chapter 8. Having a result to work towards proved to be an understandable way of explaining the modal analysis described in this report.

In the next chapters, the modal analysis will be explained and the corresponding equations will be derived. It is necessary to get a clear understanding of each step in the derivation, to be able to comprehend the entire modal analysis. In chronological order, the following four steps can be distinguished:

The literature review in Chapter 4 contains background information on the subject. The passive and active vibration control, modal analysis methods and on how to take accurate measurements will be explained here.

Chapter 5 describes the steps and calculations necessary for the modal analysis of a structure. For each step a 'simplified' example will be worked out, which serves as an overview of the calculations necessary. During this analysis, kinematic models of the structure are derived using the Newton-Euler or Lagrange's method.

0 to Chapter 8 describe the methods used to take measurements of a structure and how to process this measurement data to derive the modal description. By measuring the vibrations in the structure, after excitation with a certain force, the modal description of the system can be derived. In combination with the theoretical modal analysis, this experimental analysis provides a far more accurate description of the real-life system than the theoretical analysis on its own.

The quadrature picking technique, described in Chapter 7, can then be used to derive a modal description of the structure. Subsequently, the frequency response of the structure will be derived from both models, which supposedly results in the same responses.

The curve fitting method described in Chapter 8 can be used to reconstruct the measurement data, using only a small part of the measurement system. Predictions can then be made for the rest of the frequency response functions.

Application of the modal analysis on real-life situations will provide a clear view on well the method works and the accuracy of the modal analysis method. The error of the modal analysis is described in Chapter 9.

Chapter 2 Results

As explained in the outline, the results will be presented first. Several experimental modal analyses were carried out during this research. Chapter 2 and Chapter 3 discuss the results of these experiments. The structures analysed are machine support frames. These frames were installed to provide a high stiffness suspension, as well as active cancellation from the floor vibrations.

2.1 Roadmap

To get to the results, the roadmap of the modal analysis, Figure 2.1, was followed. Each step in this roadmap is also described in the corresponding chapter of this report. The reader is referred to Chapter 5 to Chapter 8 to gain more insight on the derivations of these steps.

For each experimental modal analysis, the mode shapes of the structure and the plots used to find the natural frequencies of the structure are shown, as well as the setup consisting of the measurement positions. The resulting frequency response function plots of the estimations and predictions for the measured data show the measured frequency response functions, the estimation/prediction made by the modal analysis and the estimation/prediction including compensation for residual effects. The prediction plots show predictions for the unmeasured frequency response functions. To check these results, the unmeasured frequency response functions where measured and plotted against the predictions.

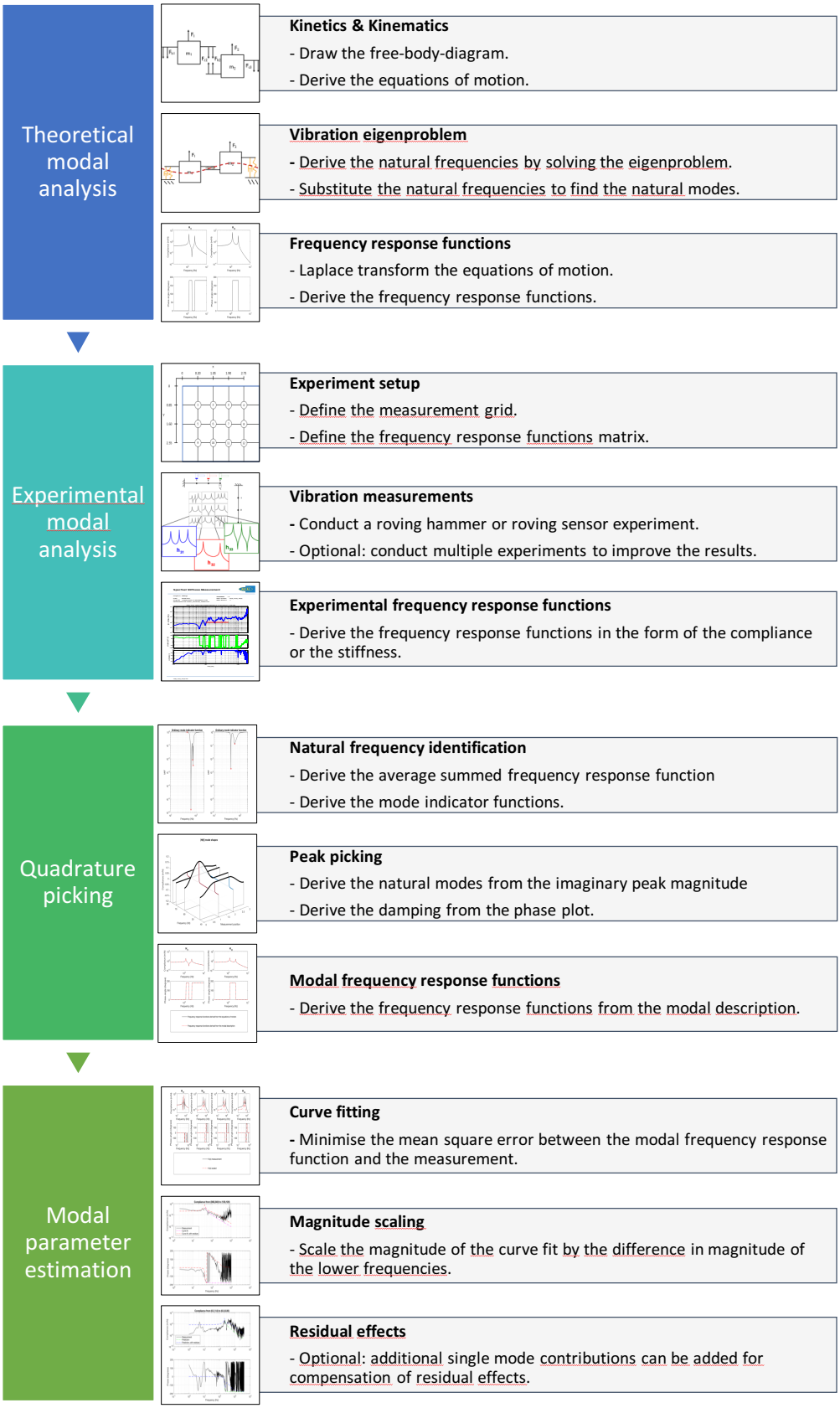


Figure 2.1: Modal analysis roadmap.

2.2 SEMICON FAB-X

The measurement positions for SEMICON FAB-X are shown in Figure 2.2. A roving hammer experiment was conducted here, where the frequency response functions are measured from each excitation position to the sensor position at location 16 $(360,240)$. Besides the roving hammer experiment, the following driving point measurements where conducted:

- Location 8 $(120,120)$
- Location 10 $(360,120)$
- Location 20 $(120,360)$
- Location 21 $(240,360)$
- Location 22 $(360,360)$
- Location 26 $(120,480)$
- Location 28 $(360,480)$
- Location 34 $(360,600)$

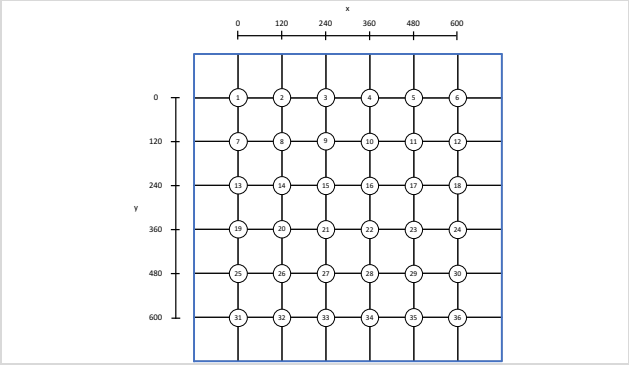


Figure 2.2: Experimental modal analysis setup for SEMICON FAB-X.

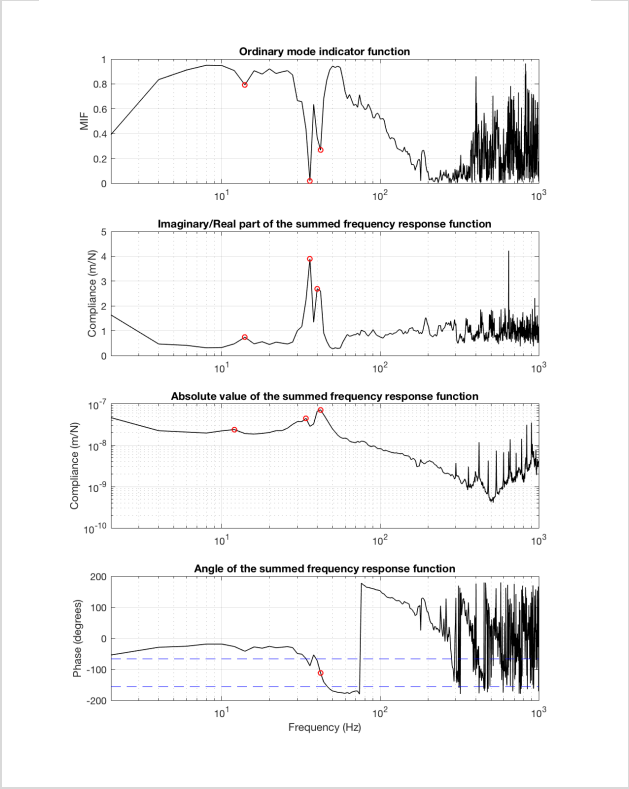


Figure 2.3: Identification of the natural frequencies of SEMICON FAB-X.

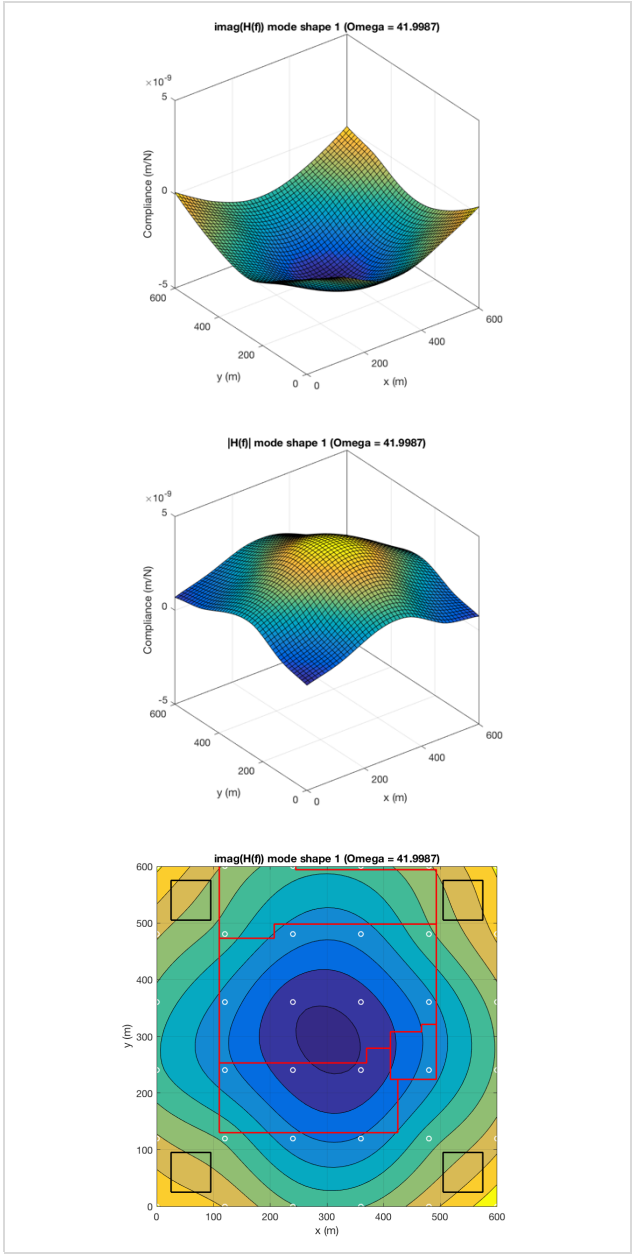


Figure 2.4: Mode shapes for SEMICON FAB-X.

Additionally, the mode shape plot of Figure 2.4 shows the steel frame (red) inside the pedestal and the columns (black) supporting the pedestal. The measurement positions are denoted by white circles.

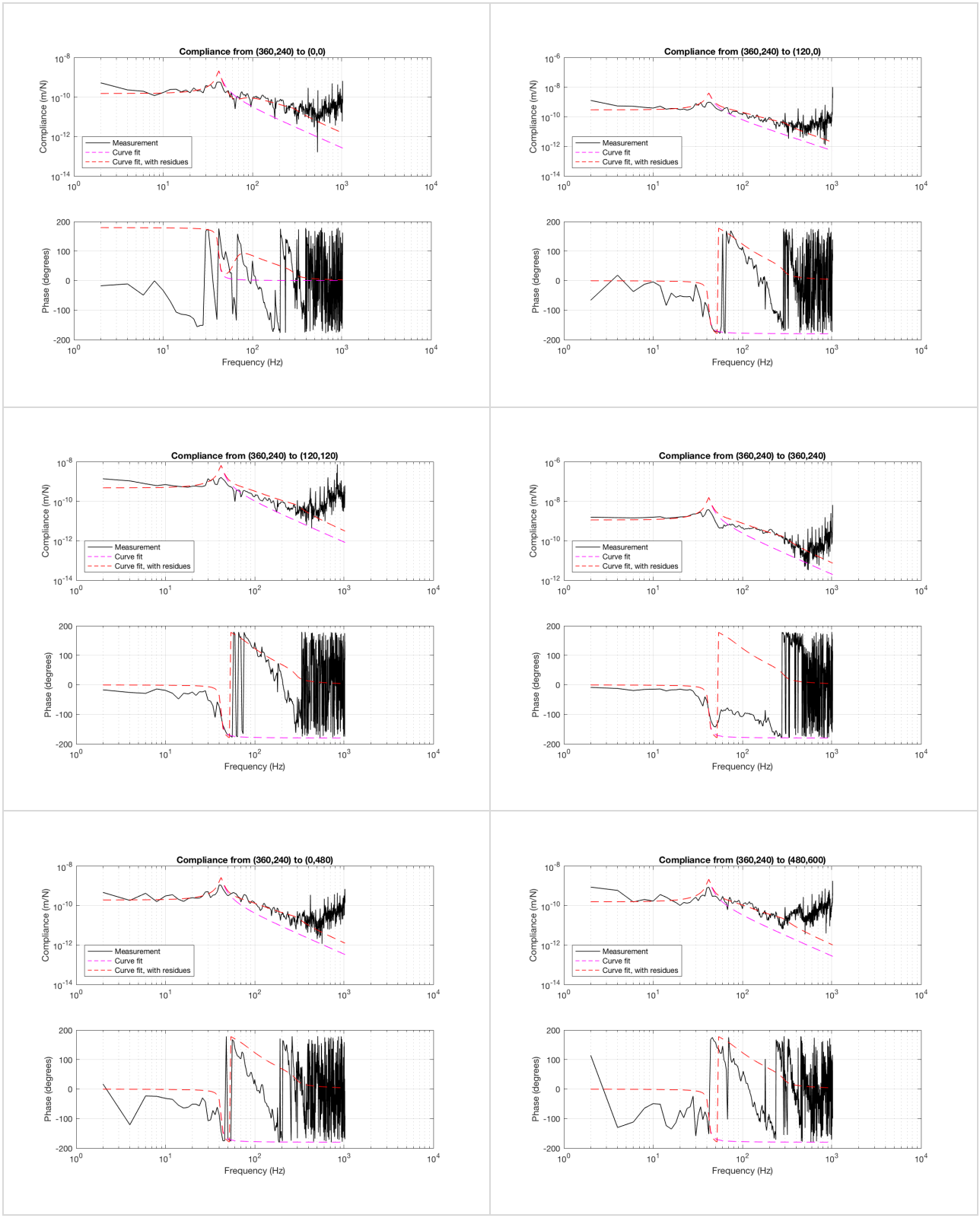


Figure 2.5: Frequency response functions and the corresponding curve fit for SEMICON FAB-X.

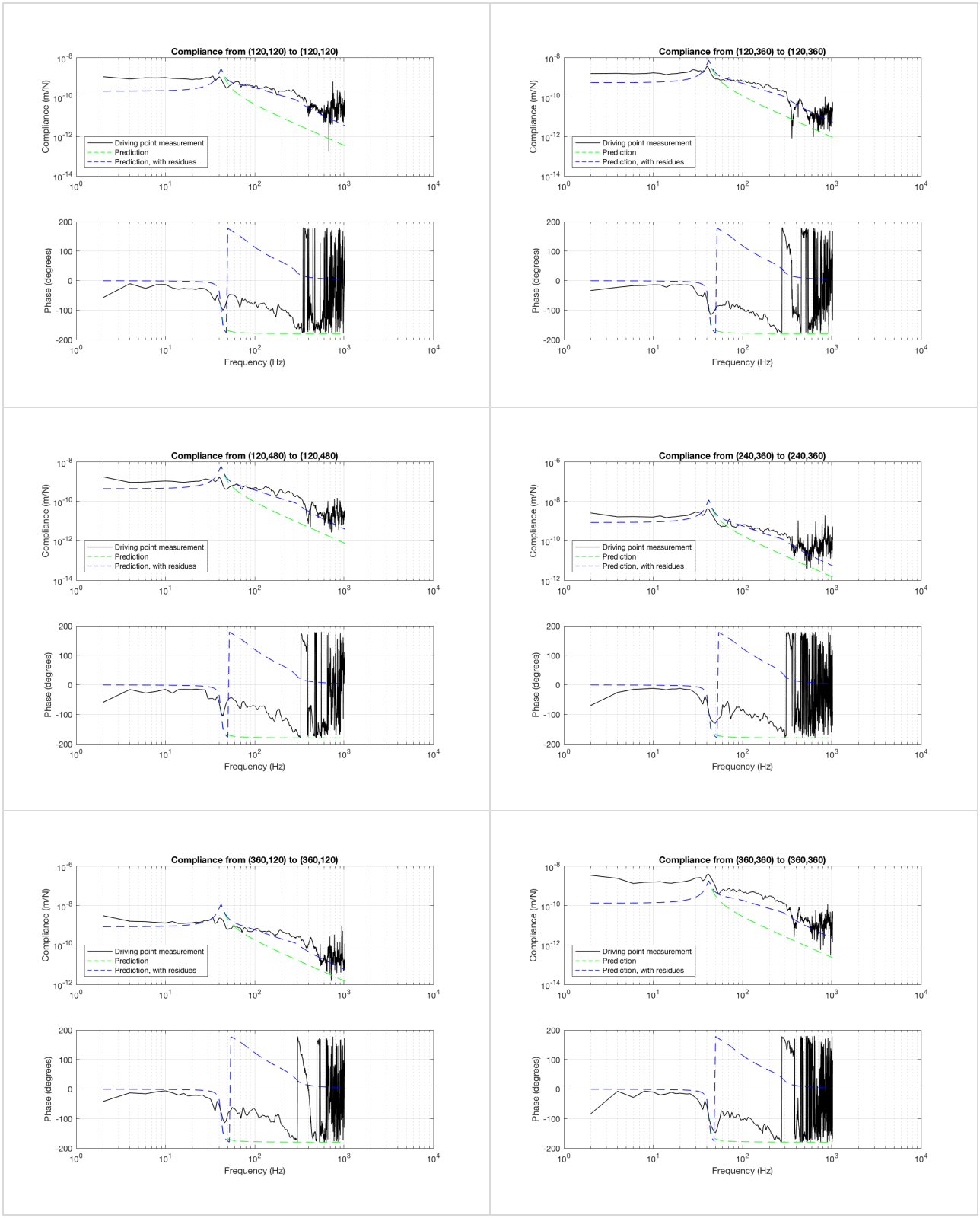


Figure 2.6: Driving point frequency response functions and the corresponding prediction for SEMICON FAB-X.

2.3 SEMICON FAB-Y

The measurement positions for SEMICON FAB-Y are shown in Figure 2.7. A roving hammer experiment was conducted here, where the frequency response functions are measured from each excitation position to the sensor position at location 14 $(360,240)$. Besides the roving hammer experiment, the following driving point measurements where conducted:

- Location 2 $(120,0)$
- Location 8 $(240,120)$
- Location 14 $(360,240)$
- Location 20 $(480,360)$

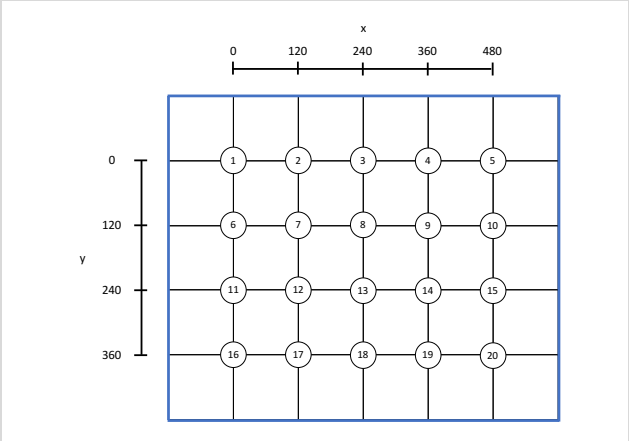


Figure 2.7: Experimental modal analysis setup for SEMICON FAB-Y.

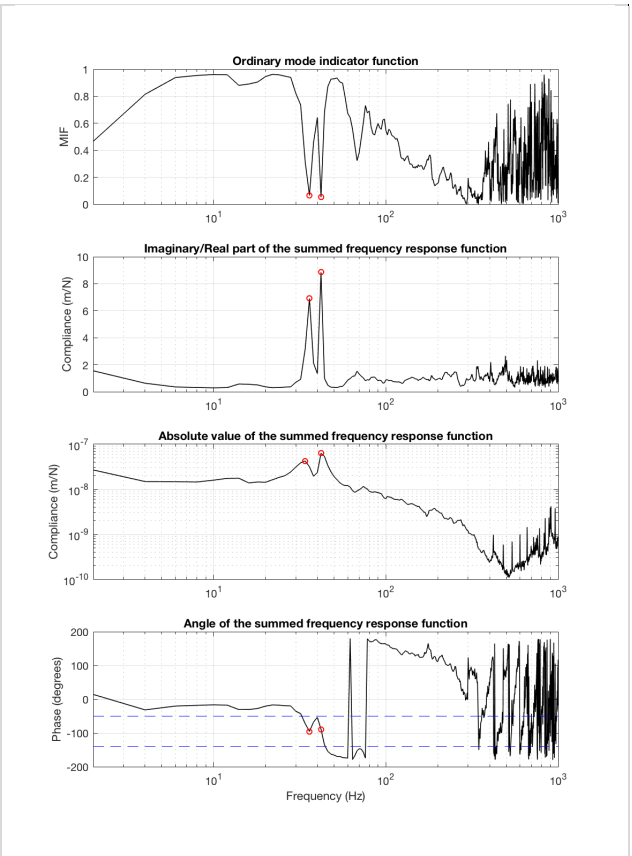


Figure 2.8: Identification of the natural frequencies of SEMICON FAB-Y.

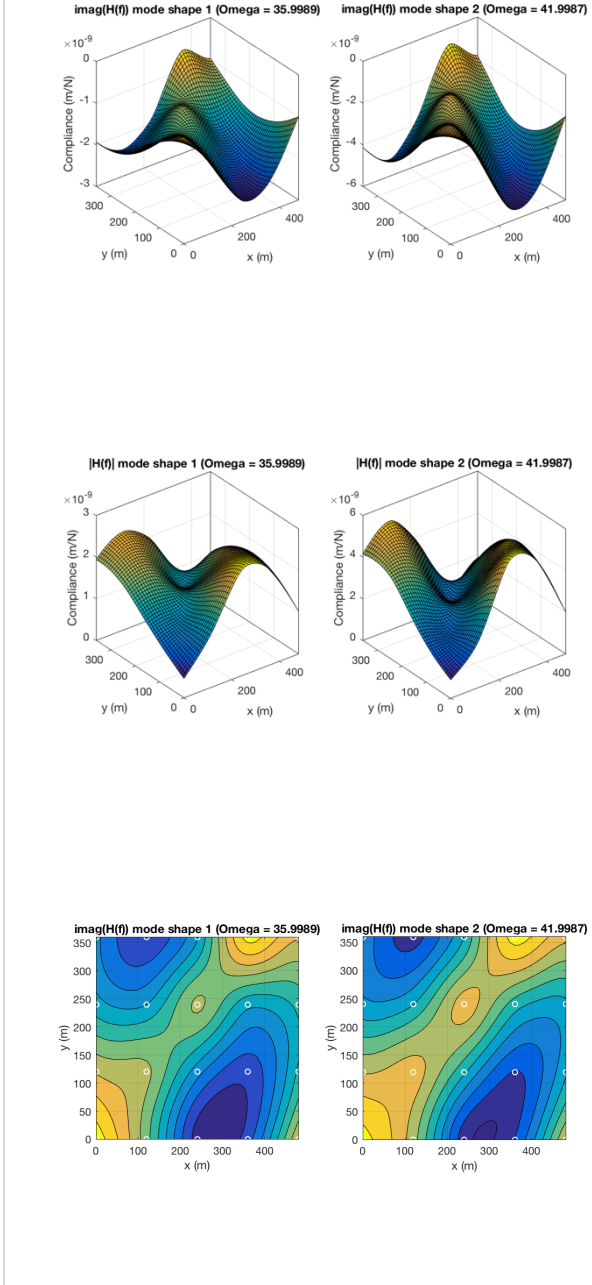


Figure 2.9: Mode shapes for SEMICON FAB-Y.

The measurement positions are denoted by white circles in the mode shape plot, Figure 2.9. The mode shape plot shows two almost identical mode shapes. This can be explained as being a result of taking measurement of a part of the system only and not the entire system [2]. These modes may seem identical for the top plate of the machine support frame, but other parts of the frame probably differ in mode shape.

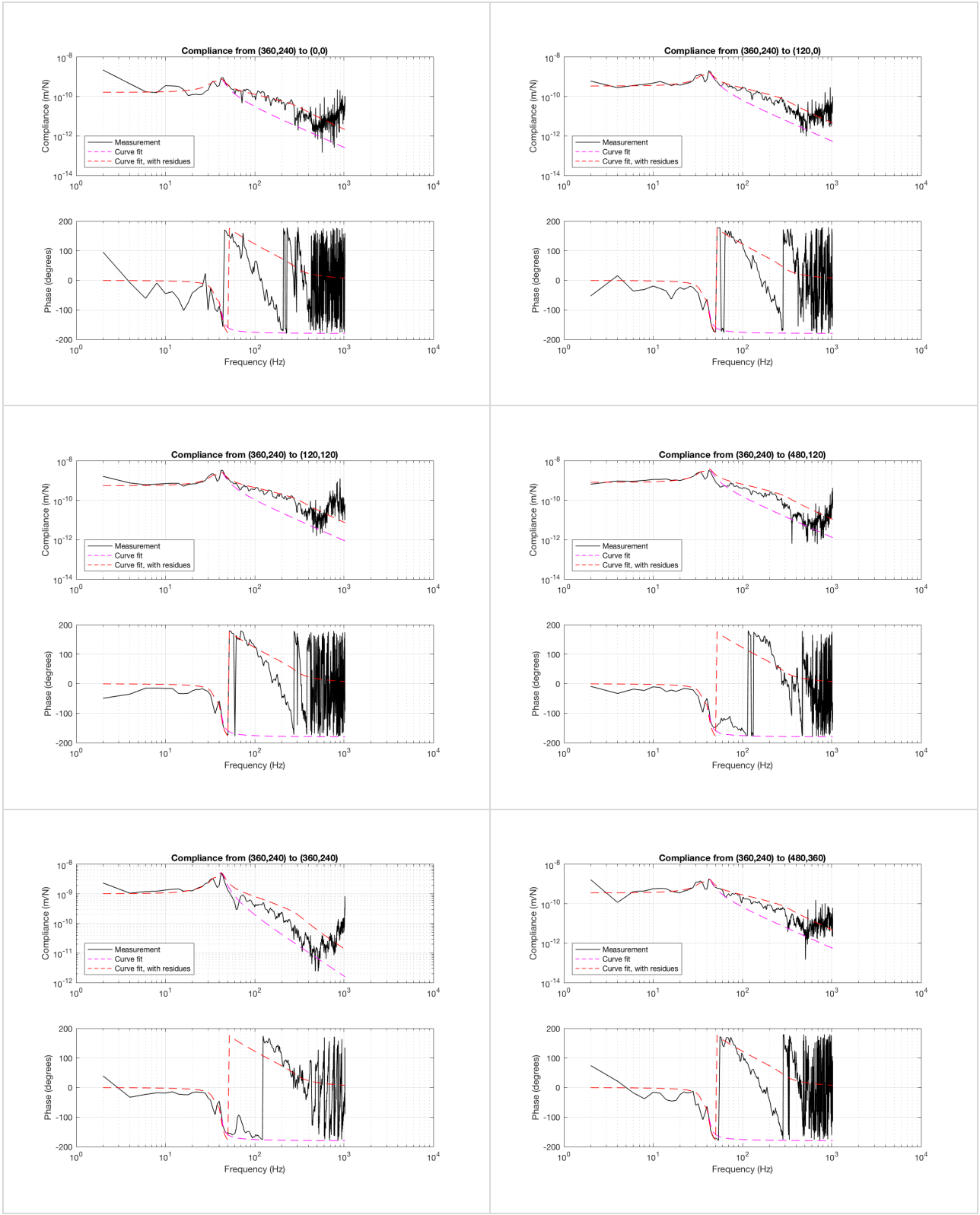


Figure 2.10: Frequency response functions and the corresponding curve fit for SEMICON FAB-Y.

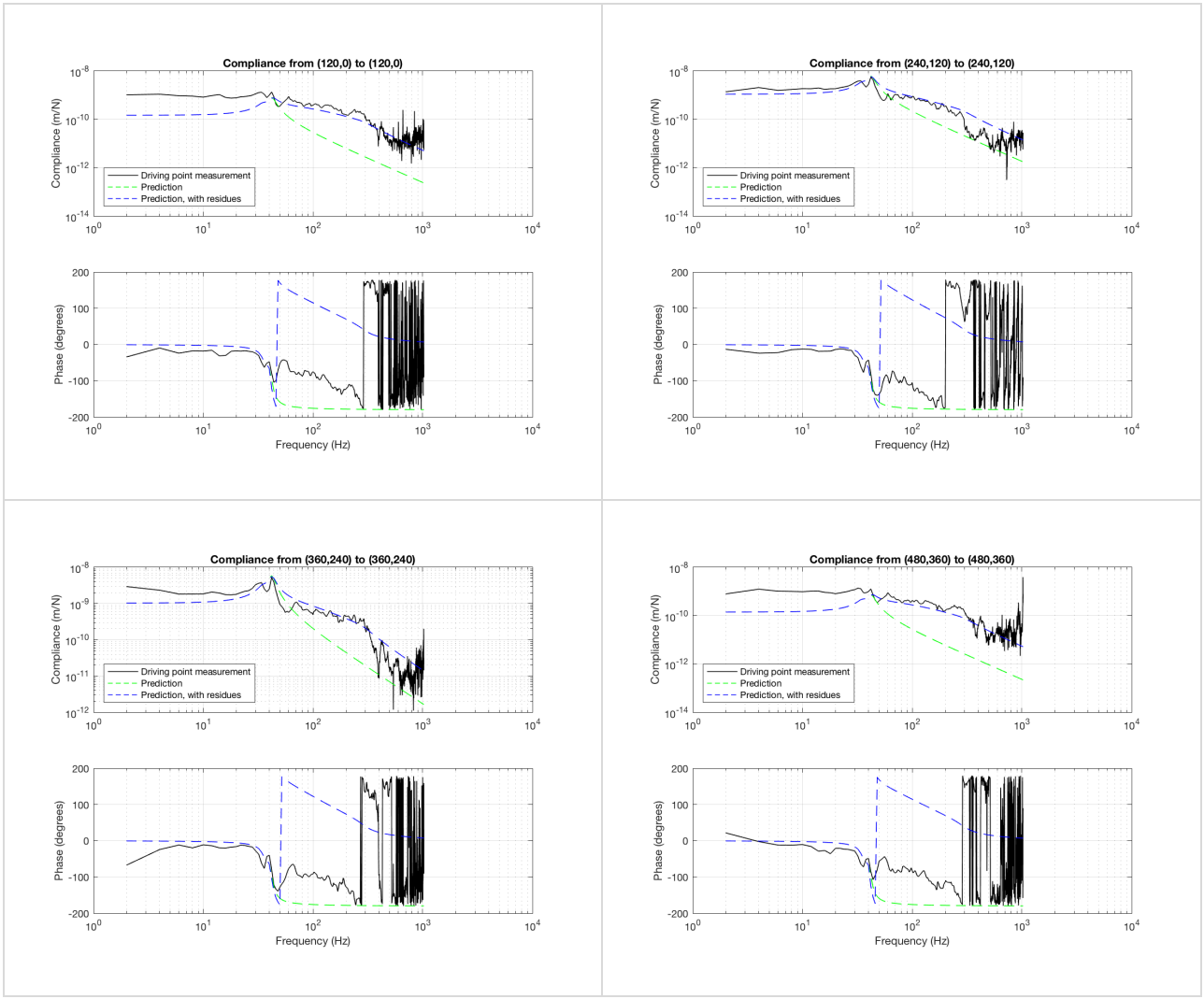


Figure 2.11: Driving point frequency response functions and the corresponding prediction for SEMICON FAB-Y.

2.4 SEMICON FAB-Z

The measurement positions for SEMICON FAB-Z are shown in Figure 2.12. All frequency response functions of the entire system were measured during this experiment.

The measurement positions are again denoted by a white circle in the mode shape plot, Figure 2.14.

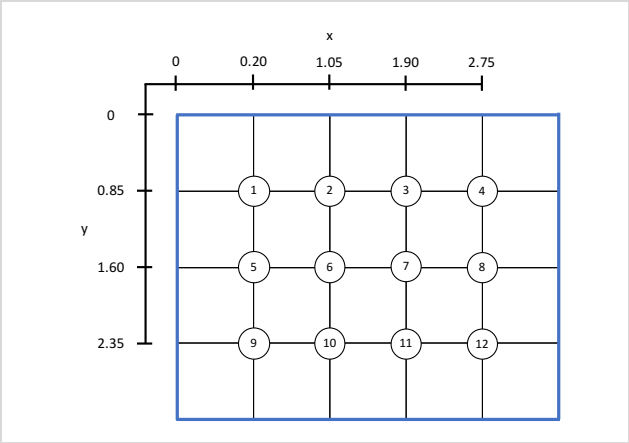


Figure 2.12: Experimental modal analysis setup for SEMICON FAB-Z.

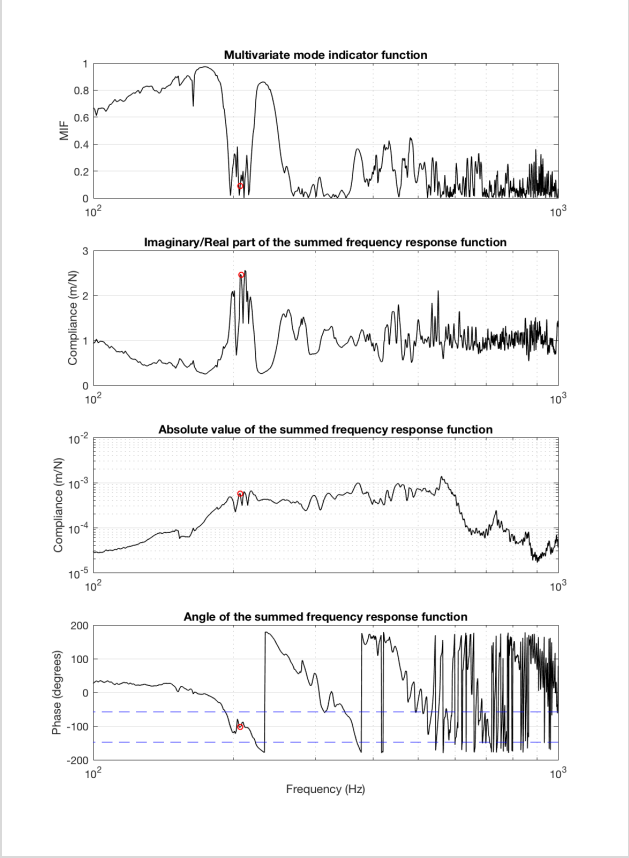


Figure 2.13: Identification of the natural frequencies of SEMICON FAB-Z.

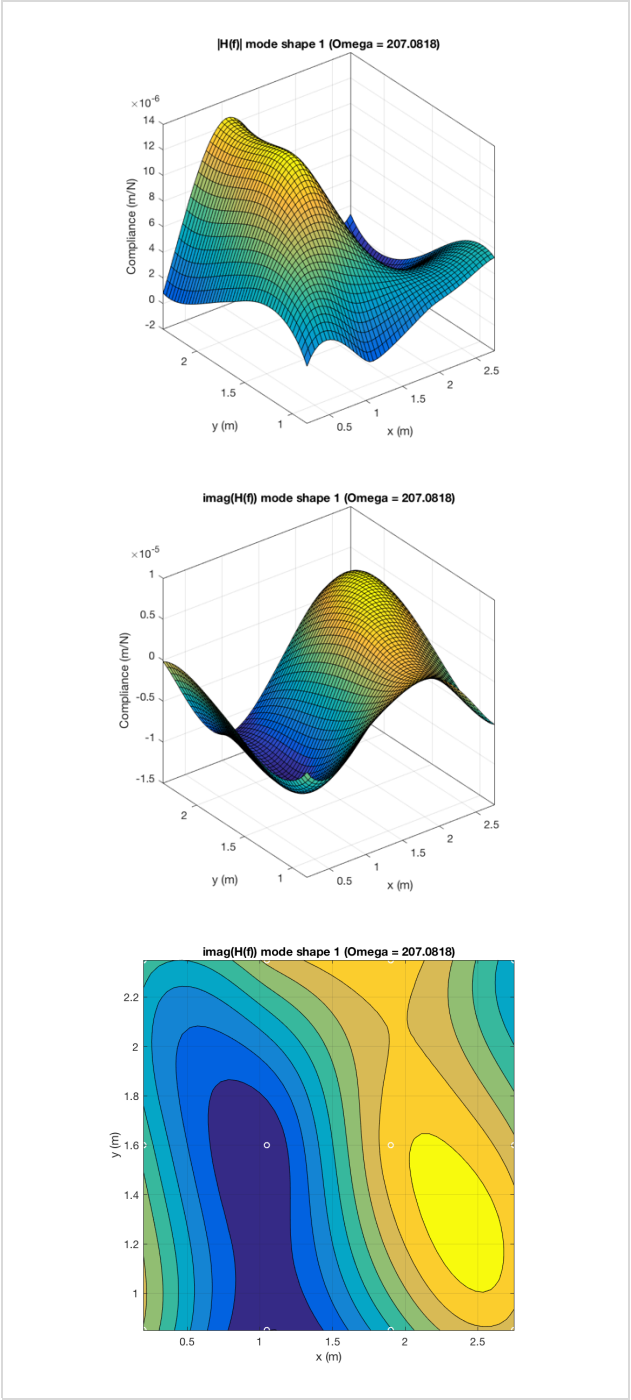


Figure 2.14: Mode shapes for SEMICON FAB-Z.

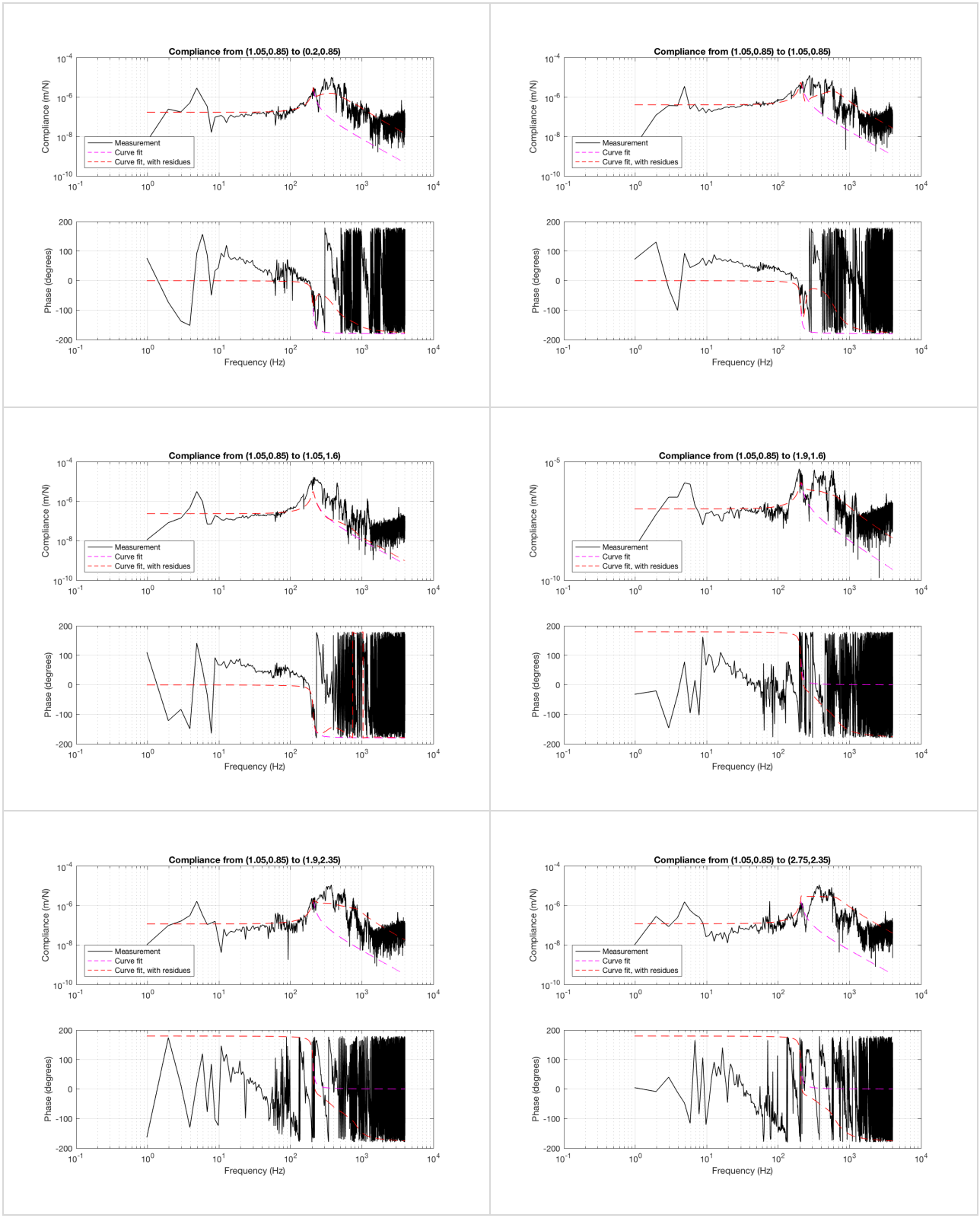


Figure 2.15: Frequency response functions and the corresponding curve fit for SEMICON FAB-Z.

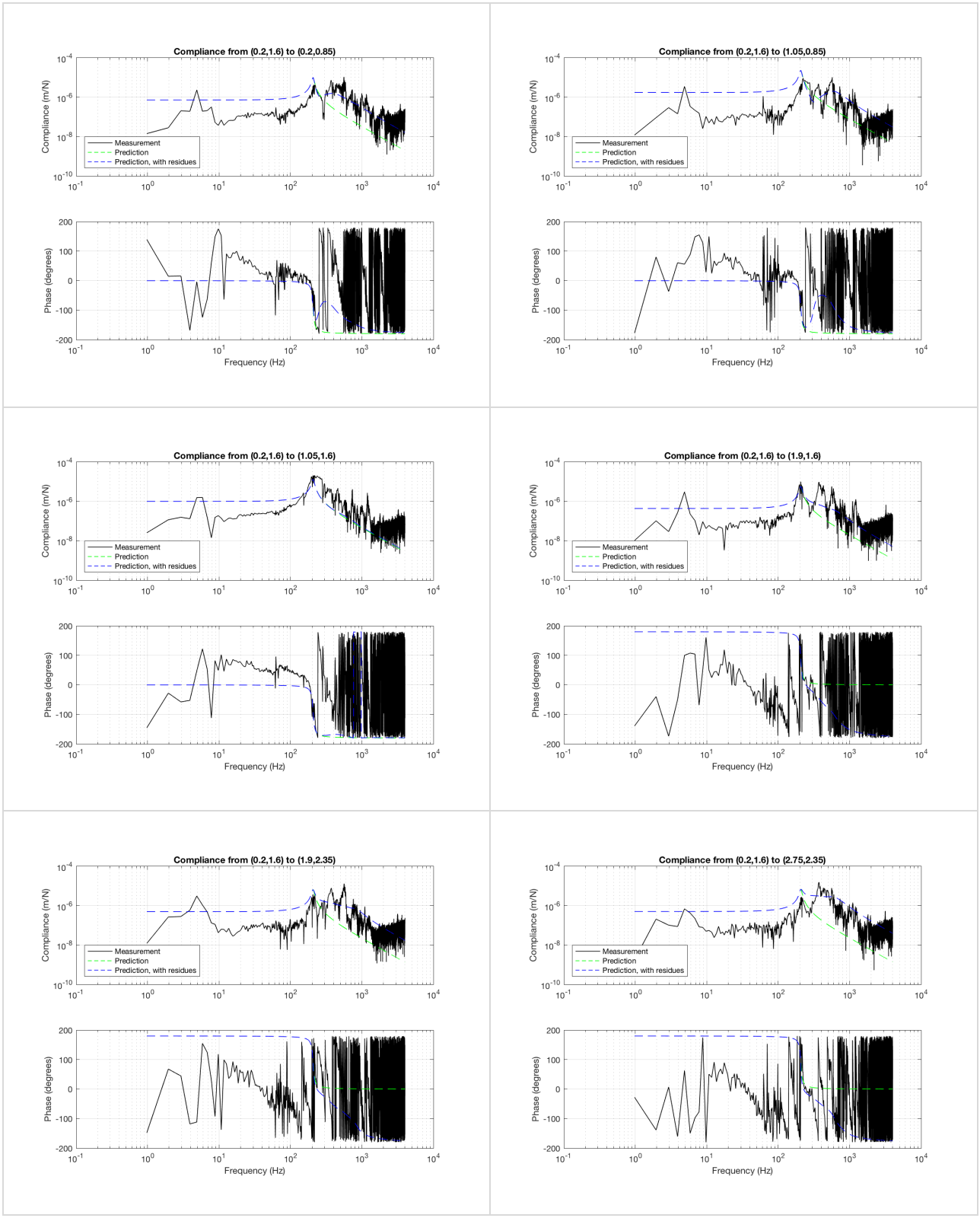


Figure 2.16: Frequency response functions and the corresponding prediction for SEMICON FAB-Z.

Chapter 3 Conclusion

The modal analysis techniques described in this report give a clear view on how to derive the modal parameters of a system from the experimental measurements.

The theoretical modal analysis shows that in three steps, the frequency response functions of a structure can be derived in an accurate way. The first step in this process is the derivation of the equations of motion:

1. $\boldsymbol{M}\,\ddot{\boldsymbol{z}} + \boldsymbol{C}\,\dot{\boldsymbol{z}} + \boldsymbol{K}\,\boldsymbol{z} = \boldsymbol{F}(t)$

Both methods to derive these equations (Newton-Euler and Lagrange) gave exactly the same results, therefore these methods can be used to check the correctness of one another.

The second step is solving the vibration eigenproblem, resulting in the modal description of the system:

2. $det\big(\boldsymbol{D}(\omega)\big) = 0$

This step gave additional problems, as it became obvious that only undamped problems can be solved with relative ease, except for proportional damping, where the damping matrix is uncoupled. Any form of damping results in a fourth-order problem, which is impossible to solve analytically. Numerical methods should be used to solve such a problem. One way to circumvent this is by making use of quadrature picking to derive the modal description.

The third and last step is the derivation of the frequency response functions:

3. $\frac{\boldsymbol{Z}(s)}{\boldsymbol{F}(s)} = (-\omega^2\,\boldsymbol{M} + i\omega\,\boldsymbol{C} + \boldsymbol{K})^{-1}$ and $\frac{\boldsymbol{Z}_m(s)}{\boldsymbol{Q}(s)} = (-\omega^2\boldsymbol{I} + i\omega\,\boldsymbol{C}_m + \boldsymbol{K}_m)^{-1}$

The frequency response functions derived from the equations of motion and from the modal description both result in exactly the same functions, provided that the modal description of the system is perfectly available. This showed that the theoretical modal analysis indeed results in the correct models of the system. It was however assumed that all system parameters where available, which in a real-life situation is never the case. It was found that small deviations from the exact system parameters, resulted in large errors in the frequency response functions. However, scaling-by-hand is always a possibility to increase the accuracy and counter these errors in system parameters.

One useful property of the modal frequency response functions became evident, which is the fact that these equations uncouple as a result of the natural coordinate transformation. The motion of the bodies can then be described for each body individually.

Experimental modal analysis was conducted to measure the vibration behaviour of a structure. While the setup of the experiment is rather straightforward, care must be taken that the entire system is measured to prevent any loss in data in the mode shapes of the structure and thus having mode shapes which look identical. If a sensor or excitation position is chosen on a node of a mode shapes, it also becomes impossible to measure the respective mode.

The quadrature picking method derived next, gave an easy to understand method to derive the modal parameters from the measurement data.

The quadrature picking method consists of selecting the resonance peaks in the measurement data, where the frequency of the peak is equal to the natural frequency of the system and the imaginary part of these peaks equal to the corresponding index in the corresponding mode vector. The damping is gained from either the half-power point or the phase plot. It was found that the mode indicator functions can give better insight in the measured resonance peaks, showing peaks that might not be visible from the frequency response functions alone. The derivation of the quadrature picking method answered the first research question:

- Is it possible to use only the data from a single roving hammer or roving sensor experiment, to derive the modal description of the systems?

It is indeed possible to derive the modal description of the entire system from a single roving hammer/sensor experiment. The mode shapes of the structure could be derived from a single row or column of the frequency response functions matrix.

This method resulted in a modal description of the structure, which accurately gave the mode shapes of the system. It did however not result in the exact modal description of the system, as was expected from the theoretical modal analysis. Only the imaginary part of the modal matrix was found and the diagonal terms of the damping matrix. Also, because the frequency response functions are summations of single mode contributions, quadrature picking does not result in the 'pure' parameters for a single mode only, but rather parameters resulting from a combination of modes. When trying to reconstruct the frequency response functions from this description, it was found that the functions where completely off, having large errors in both the shape and magnitude. It was therefore necessary to derive a method to scale these frequency response functions, namely curve fitting.

The parameter estimation gave a variety of methods to derive the frequency response functions of a structure from the experimental modal analysis, of which three methods where attempted:

- Calculation of the modal description of the system from the experimental modal analysis.
- Tuning of a simple finite element model, according to the measurement data.
- Measuring the required frequency response functions directly.

It was quickly found that this part of the modal analysis is also the most complex part, where not all methods result in accurate frequency response functions and that there is no general method which can be used for all cases. Each system must be analysed individually and depending on the expected results from the modal analysis, a modal parameter estimation method must be chosen, answering the second research question:

- Can the data from a single roving hammer or roving sensor experiment be used to make an accurate prediction of the complete vibrational behaviour of the system?

These predictions can be made, but the accuracy depends strongly on the structure and the measurements taken. It was found that, when the resonance peaks are accurately measured and clearly visible, a single row or column is enough to make a prediction for the frequency response functions of the entire system. This is a result from the definition of the frequency response functions matrix.

The experimental modal analysis of both theoretical and real-life situations gave valuable information about the accuracy of the entire modal analysis method. The calculation of the error made answered the third research question:

- Which method should be used in a certain situation, such that the accuracy of the prediction is as high as possible?

While the mean square error curve fitting technique gave the best way of predicting any unmeasured frequency response functions, the accuracy is mainly depended on the optimisation algorithm used. A good initial estimation, resulting from the quadrature picking method, is therefore absolutely necessary. This is not always the case however. The accuracy could be increased by taking into account more of the residuals. These residual effects could be compensated for by addition of the standard frequency response function for each residual, to the frequency response functions of the structure.

The main problem that occurred during the modal analysis was the visibility of the resonance peaks. If peaks are left out of the calculations, only a basic model can be fit. This fit might or might not give an accurate estimation, depending on the type and complexity of the structure.
This complexity of the system was also found to be a good indication for which method should be used for the modal parameter estimation. The tuning of a simple finite element model seems to be the most accurate estimation method even if the damping is extremely high. This method should however only be used for the estimation of a model with a single resonance peak, as complex models cannot be found this way. Care should be taken that the frequency response function found this way consists of parameters which do not have to be equal to the theoretical system parameters. If a prediction of unmeasured frequency response functions is to be made, the mean square error curve fitting method should be used instead.

The prediction made by the mean square error curve fitting gives only an indication of the expected frequency response function. If these results are to be derived very accurately, it is better to take more time for the experimental modal analysis and measure the frequency response functions directly.

Chapter 4 Literature review

4.1 Passive vibration isolation

In many applications, it is not only of interest to study the behaviour of the system itself, such that the vibration amplitude of the motion can be seen and is within an acceptable range, but it is also demanded that the system transmits as little vibration to the environment as possible [3]. In these cases, it is desired to isolate heavy vibrations of a part of a system from the rest of it. Such an analysis can be done as soon as the forced vibration response of the system is calculated.

Figure 4.1: Viscously damped mass-spring system subjected to external excitation [3].

Considering the viscously damped mass-spring system that is excited by an external force, Figure 4.1. The transmitted force acted upon the base is the sum of the spring force and the viscous damping force:

$$F_t = kx(t) + c\dot{x}(t)$$

The ratio of the magnitude of the transmitted force to the applied force is known as the transmissibility and can be calculated from:

$$\frac{|F_t|}{F} = \sqrt{\frac{1 + \left(2\xi\frac{\omega}{\omega_n}\right)^2}{\left(1 - \left(\frac{\omega}{\omega_n}\right)^2\right)^2 + \left(2\xi\frac{\omega}{\omega_n}\right)^2}}$$

From this equation can be concluded that the force transmitted to the base is smaller than the force acting on the system when the excitation frequency is larger than the square root of two times the natural frequency of the system. However, in this frequency range, increasing the damping has a negative influence on the vibration isolation. If the target is to minimise the transmitted force for $\omega > \omega_n\sqrt{2}$, it is best to have no damping at all.

Vibration absorbers

Especially when the system is subjected to a harmonic excitation of constant frequency causing undesirable vibrations, vibration absorbers are often used [3]. Three types of vibration absorbers exist: the undamped (connected by springs) and damped (connected by dampers) vibration absorbers and the dynamic (connected by springs and dampers) vibration absorber, shown in Figure 4.2.

Figure 4.2: Dynamic vibration absorber [3].

The left figure shows a machine connected to the fixed world by a spring. The machine is subjected to an external force, such that it causes unacceptable vibrations of the machine. Reduction of these vibrations by eliminating the external force or changing the system parameters might not be possible. Therefore, a dynamic vibration absorber can be connected to the machine, as shown in the right figure.

The equations of motion of the combined system can be derived as:

$$\begin{bmatrix} m_m & 0 \\ 0 & m_a \end{bmatrix} \begin{Bmatrix} \ddot{x}_m \\ \ddot{x}_a \end{Bmatrix} + \begin{bmatrix} c & -c \\ -c & c \end{bmatrix} \begin{Bmatrix} \dot{x}_m \\ \dot{x}_a \end{Bmatrix} + \begin{bmatrix} k_m + k_a & -k_a \\ -k_a & k_a \end{bmatrix} \begin{Bmatrix} x_m \\ x_a \end{Bmatrix} = \begin{Bmatrix} F e^{i\omega t} \\ 0 \end{Bmatrix}$$

Figure 4.3 shows the plot of the frequency response function of the system, for three values of the damping coefficient. This plot raises a single question: 'how can the vibration absorber be designed such that the peak value of the curve is as low as possible?'.

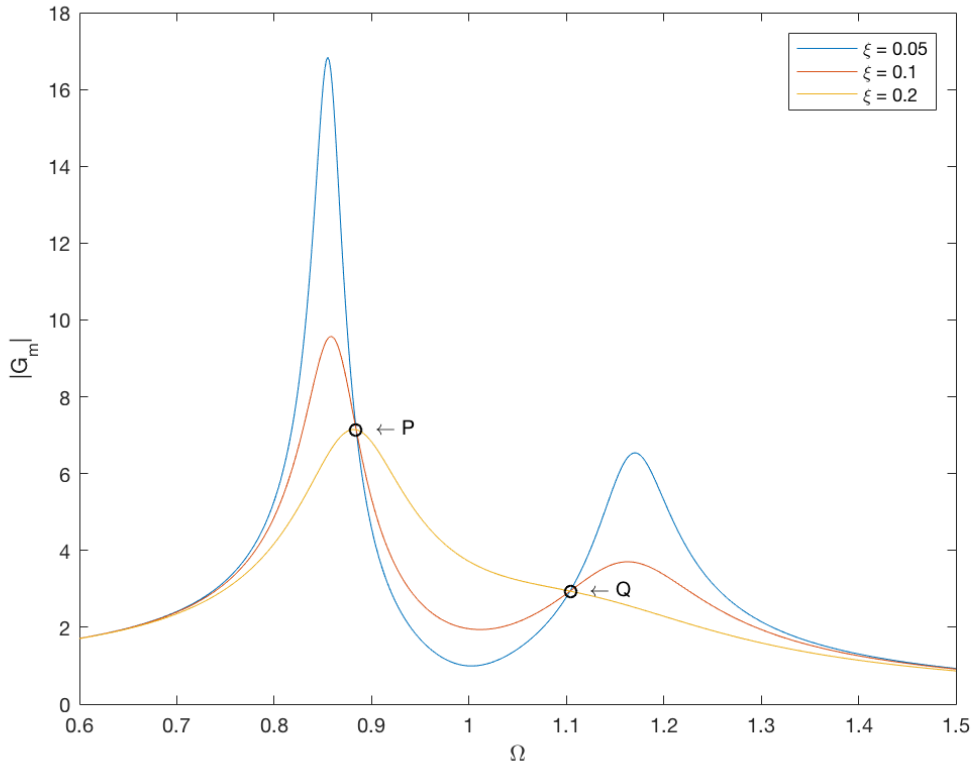


Figure 4.3: Magnitude of the response of the main system for three values of the damping coefficient.

The figure shows two common intersection points P and Q, from which the value does not depend on the damping coefficient. If the design parameters are chosen in such a way that one of the peaks goes through one of the intersections, the vibration absorber has been optimally designed. However, the maximum value of the curve would be further reduced when the common points P and Q would be of identical magnitude. This means that it may be acceptable to have an increased value of Q, if that means that P will be reduced. Because the intersection points are independent of the damping coefficient and the mass ratio is set for the system, the only parameter influencing these points is the ratio of natural frequencies Ω_a. The most favourable situation is obtained when first the ratio of natural frequencies is chosen such that the magnitude of P and Q is equal and then the damping factor is chosen such that the peak of the curve pass through one of these points.

4.2 Active vibration cancellation

Active vibration control is aimed at reducing the vibration level of a mechanical structure [4]. Contrary to passive methods, like vibration absorbers, shock mounts and base isolation, active vibration control is based on the application of a secondary vibration to the structure, resulting in minimum residual vibrations. Active application of force in an equal and opposite fashion to the forces imposed by external vibrations result in this secondary vibration. The active vibration control system must therefore constantly measure either the vibrations in the structure or the force applied by the external sources.
With this type of force application, a precision industrial process can be maintained on a platform essentially vibration-free. Many precision industrial processes cannot take place if the machinery is being affected by vibration. For example, the production of semiconductor wafers requires that the machines used for the photolithography steps are used in an essentially vibration-free environment or the sub-micrometre features will be blurred.

The typical active vibration control system uses several components:

- A platform suspended by several active drivers (that may use voice coils, hydraulics, pneumatics, piezo-electric or other techniques).
- Accelerometers that measure acceleration in three degrees-of-freedom.
- An electronic amplifier system that amplifies and inverts the signals from the accelerometers. A PID controller can be used to get better performance than a simple inverting amplifier.
- For very large systems, pneumatic or hydraulic components that provide the high drive power are required.

MECAL Equalizer

The Equalizer from MECAL is a type of active vibration cancellation system that is designed to counteract vibrations, especially in the 20 Hz to 80 Hz frequency band. The Equalizer is capable of reducing vibrations over a large floor area by a factor of 3 to 5.

The vibrations on the machine pedestal can result from two sources: external forces and forces from the precision machine itself. A soft suspension would provide isolation for the vibration from the floor, however making the vibrations from the machine worse. An active system will in this case help to achieve a higher level of vibration isolation, while keeping some of the stiffness of the pedestal.
The second option is a stiff suspension, which results in the best isolation of the vibrations from the machine. This does offer little to no isolation from the floor vibrations however. The Equalizer designed by MECAL makes this system active, resulting in both a high stiffness suspension and active cancellation from the floor vibrations.

4.3 Modal analysis methods

The dynamic behaviour of structures can be measured using so called experimental modal analysis methods. According to Mark H. Richardson [5], two fundamentally different methods of modal analysis exist: the normal mode method and the frequency response function method.

Normal mode method

The objective of the normal mode method is to excite the natural modes of the structure, by means of a device called a ‘shaker’. A shaker can output a harmonic excitation with a certain frequency. The frequency of the harmonic excitation will be set equal to a natural mode of the structure, also called the natural frequency. The process of adjusting the amplitude and frequency of the shaker to excite a specific natural frequency is called ‘modal tuning’.

Once a mode is properly excited, the excitation amplitude of the system can then be measured using measurement equipment (i.e. accelerometers), at many points on the structure. This is called 'modal dwell'.

A variation on this type of method is the frequency sweep, where the harmonic excitation frequency can be swept over a certain range of frequencies. This method is used to find the natural frequencies of the structure.

To measure the damping coefficient, the shaker is shut off to simulate an impulse response of the structure at the frequency of the mode. Ideally, the structure should exhibit a damped harmonic response at all points, with a frequency equal to the excited natural frequency. Most of the time, this is not the case and the structure will show a beating of several natural modes. The beating phenomenon can be explained as a type of interference between two vibrations with different frequencies. When this occurs, amplification and damping of the combined vibration alternate each other.

The are several problems which make this modal analysis difficult and time consuming:

- The location of excitation is difficult to predict without forehand knowledge about the modes of vibration of the structure. This is of importance as the point of intersection of the deformed state with the equilibrium state has zero amplitude and therefore does not move during the vibration of the structure at the corresponding natural mode. If the shaker is positioned at such an intersection point, it gets 'pushed' up and down instead of having zero amplitude. Therefore, changing the mode shape at the corresponding eigenfrequency.
- It is extremely difficult to excite closely coupled modes, one at a time.
- Since all the mode data is collected during the modal dwell, the structure must be instrumented with enough accelerometers, so that all degrees of freedom can be measured at once.

Frequency response function method

The frequency response function method uses the Fast Fourier Transform algorithm to derive frequency response functions from measurement data of various points on the structure.
During the experiment, the structure is excited by means of an impact force applied by a hammer or shaker. This is done on a single point on the structure. The response of the structure can then be measured in multiple points on the structure at the same time. This data about the response and the input force applied by the impact hammer can then be used to calculate the frequency response functions.

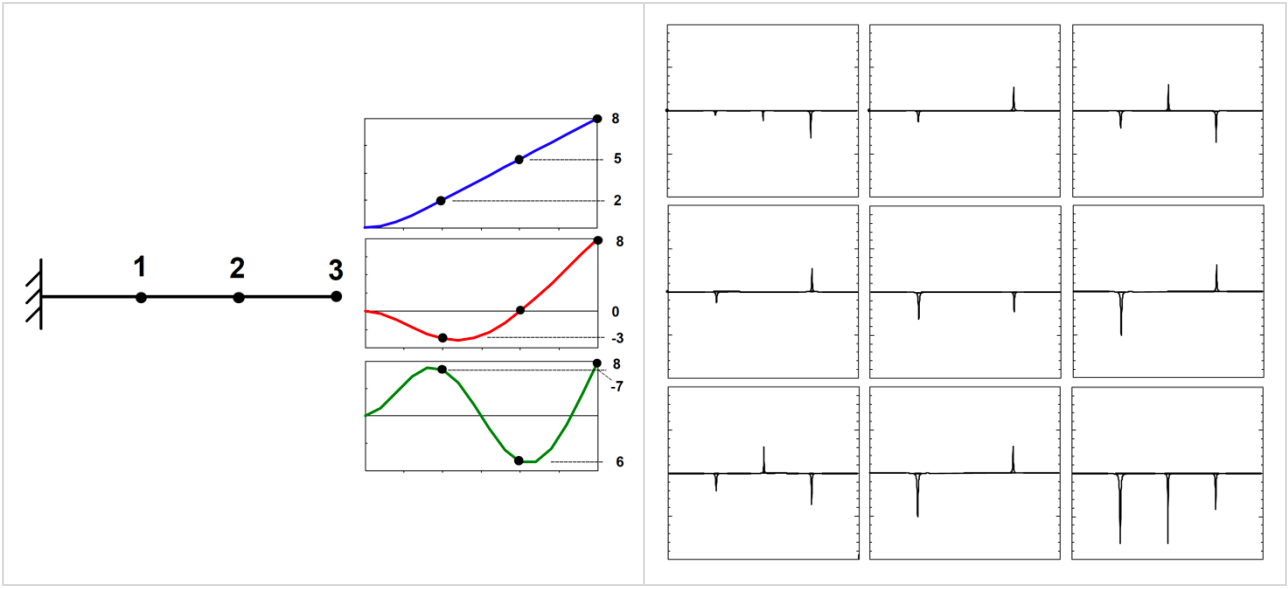


Figure 4.4: Excitation of a beam under bending, mode shapes and the imaginary part of the frequency response functions for the complete system [2].

Modal parameters are identified by performing further computations (i.e. "curve fitting") on this set of frequency response function measurements. Modal frequencies correspond to peaks in the imaginary part of the frequency response functions. A peak should exist at the same frequency in all measurements, except those measured at "node" points, where the modal amplitude is zero. The width of the modal peak is related to the damping of the mode. That is, the wider the peak, the higher the modal damping. The mode shape is obtained by assembling the peak values at the same frequency from all measurements.

Consider a beam being excited on bending, Figure 4.4. If the excitation of three points on the beam are measured, a total of nine possible input to output frequency response functions can then be derived [2]. By means of peak picking, the mode shapes of the beam can be derived from the amplitude of the peaks. This can be done for each row or column of the matrix individually. Figure 4.5 shows the mode shapes derived from the middle and bottom rows of the frequency response function matrix. These figures show some interesting features of the frequency response function matrix. While it seems to be possible to derive the mode shapes from a single row or column of the matrix. If the measurement points are located exactly at a node of the system, the magnitude of the frequency response function becomes zero and it therefore becomes impossible to extract the mode shape.

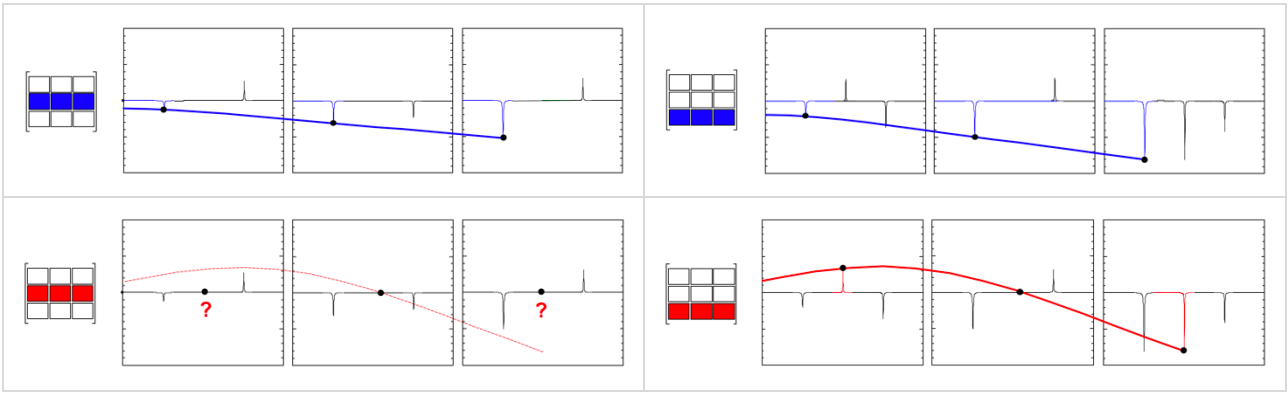


Figure 4.5: Mode shapes of a beam excited on bending, derived from the middle and bottom rows of the frequency response function matrix [2].

Another important note is that the magnitude of the peaks for different rows or columns of the frequency response function matrix are not the same, only the shape derived from these peaks is the same. Scaling is required to find the exact same value for each modal vector from different rows or columns.

Now can be concluded that any row or any column of the frequency response function matrix can be used to estimate any mode of the system, provided that the reference is not located at the node of a mode.
The frequency response function method has several advantages over the normal mode method, of which the biggest advantage is the ability to only measure a single row or column of points, instead of the complete floor. This greatly reduces the time and amount of measurements needed for analysis to complete. This method will therefore be further investigated.

4.3.1 Important considerations

As a conclusion of the modal analysis methods literature review, some important considerations will be explained here [2].

Roving hammer/sensor measurements

A roving hammer with a stationary accelerometer is one way to run an impact test that is commonly used. The other way an impact test can be performed is by keeping the impact hammer stationary and moving the accelerometer.

Both are acceptable ways for experimental modal analysis and because of reciprocity, there is no difference from a theoretical standpoint. Considering the measurements made, a single row of the frequency response function matrix will be filled for the roving sensor experiment and for the roving hammer experiment, a single column.

Driving point measurement

The measurement where the response point and direction are the same as the excitation point and direction is called a driving point measurement. For a driving point measurement, several items can be noted:

- All resonances are separated by anti-resonances.
- The phase loses 180 degrees of phase passing over a resonance and gain 180 degrees of phase passing over an anti-resonance.
- The peaks in the imaginary part of the frequency response function must all point in the same direction.

The drive point measurement can be viewed as a summation of all the modes or as the contribution due to each mode. The driving point measurements fill the diagonal of the frequency response function matrix.

Windowing

One important problem that occurs during the discrete sampling of a continuous system is aliasing, as explained in chapter 4.4. In order to minimise the error made by aliasing, weighting functions called windows can be applied. The most common windows for modal testing today are the Rectangular window, the Hanning window, the Flat Top window for shaker testing and the Force/Exponential window for impact testing. Without going into all the detail, windows always distort the peak amplitude measured and always give the appearance of more damping then what actually exists in the measured frequency response function, two very important properties that are being estimated from the measured functions. It is therefore preferable not using windows at all.
To get around not using windows on measured frequency response functions form a modal test is, by making sure the Nyquist-Shannon sampling theorem is always satisfied. Therefore, the sampling rate should be high enough for the system to be measured. For signals such as: pseudo-random, burst random, sine chirp, and digital stepped sine this requirement is, under most conditions, always satisfied and therefore are leakage free and do not require the use of a window.

Averaging multiple hits

During the experimental modal analysis, multiple hits must be measured for each individual roving hammer/sensor experiment. Noise and other disturbances can always influence a measurement, averaging multiple hits and even rejecting certain hits results in far more accurate measurements.
A measure for the reliability of the measurement is the coherence spectrum of the measurement. A coherence of one means that the measurement is accurate while a coherence below one means that there is probably some noise influencing the measurement. Care should be taken however when analysing the coherence of a measurement, as the coherence can only be calculated when multiple measurements were taken. In case of a single measurement, the coherence is always equal to one [2]. It should never be used as an indication of how many hits must be measured for each measurement.

Influence of the hammer tip

The input force spectrum exerted on a structure is a combination of the stiffness of the hammer/tip as well as the stiffness of the structure. Basically, the input power spectrum is controlled by the length of time of the impact pulse. A long pulse in the time domain, results in a short or narrow frequency spectrum. A short pulse in the time domain, results in a wide frequency spectrum.

Figure 4.6 shows a measurement where the hammer tip was changed for each measurement. The top figure shows the measurement where a very soft tip was used to excite the structure, the middle figure the measurement where a hard tip was used and the bottom figure a measurement for an in-between tip.

As can be seen in top figure, a very soft tip results in the power spectrum having significant roll-off at 400 Hz. The coherence also starts to drop at this frequency. The problem here is that there is not enough excitation at higher frequencies to cause the structure to respond. If there is not much input, then there is not much output. For a hard tip can be seen that the input power spectrum is extremely flat over all frequencies, however the coherence is not particularly good for this measurement.

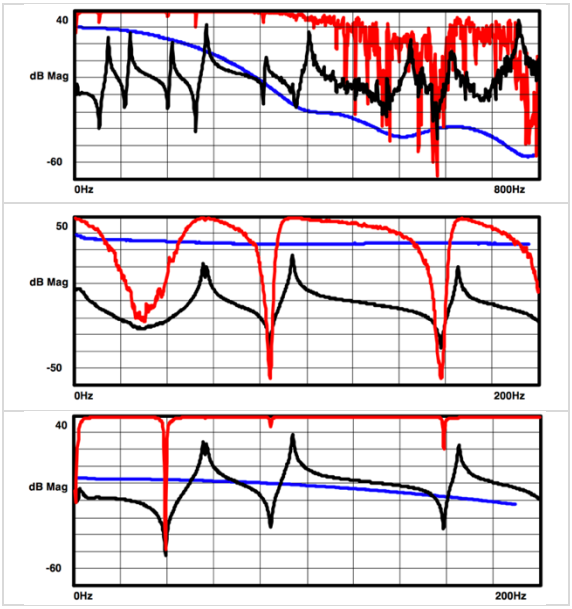


Figure 4.6: Frequency response functions (black), input power spectra (blue) and coherence spectra (red) for different hammer tips.

The following conclusion where drawn by Job Lansink, as a result of a research on the influence of the hammer tip [6]:

- The length of time of the impact pulse increases as the stiffness of the hammer tip is decreased. The coherence increases for low frequencies and decreases for high frequencies.
- Increasing the diameter of the tip results in a higher tip stiffness. The coherence decreases for low frequencies and increases for high frequencies. The stability of the measurement increases.
- Increasing the thickness of the tip lowers the average stiffness, resulting in an increase of the impact pulse time. The low frequency part increases, while the crossover frequency decreases.
- A parabolic tip results in a coherence which is favourable for both high and low frequencies.

It is therefore recommended to test the performance of the hammer tip for each experiment and changing the tip when necessary.

Heavily damped systems

According to Pete Avitabile [2], it is possible to extract heavily damped modes from the frequency response functions. It is however necessary to know the natural frequency of the damped modes, as these peaks will be less prominent in the frequency response graph. When the natural frequencies are known, a good measurement must be done, so that the damped mode can be extracted.

Real and complex modes

Characteristics of real modes:

- The mode shape is described by a standing wave, which has the presence of a fixed stationary node point.
- All points pass through their maxima, minima and zero at the same instant in time.
- All points are either totally in-phase or out-of-phase with any other point on the structure.
- The mode shapes from the undamped case are the same as the proportionally damped case. These mode shapes uncouple the system.

Characteristics of complex modes:

- The mode shape is described by a traveling wave and appears to have a moving node point on the structure.
- All points do not pass through their maxima, minima and zero at the same instant in time. Points appear to lag behind other points.
- The different DOFs will have some general phase relationship, that will not necessarily be in-phase or 180 degrees out-of-phase with other DOFs.
- The mode shapes from the undamped case will not uncouple the damping matrix.

Rotated modes

Most often when modes of a structure are very closely spaced, the mode shapes that satisfy the system can be linear combinations of each other. Therefore, the shapes might be rotated from what might have been expected. The only real requirement is that the modes of the system are orthogonal with respect to the system mass and stiffness matrices. Each of the modes of the system is unique.

This issue occurs with structures that have double symmetry and when either repeated roots or pseudo repeated roots occur. Another time it can happen is when using different numerical solution algorithms. Because the solution will typically iterate to a set of solution vectors, there is no reason why the vectors should converge towards a particular reference coordinate system.

Anti-resonance peaks

The imaginary part of the frequency response functions must all have the same direction and, in this condition, an anti-resonance exists between each mode. This is due to the fact that the magnitude of the frequency response function of mode 1 and mode 2 is equal at the anti-resonant frequency. But at this frequency, while the magnitudes are equal, the phase is 180 degrees out of phase with each other. This implies that the sum of mode 1 and mode 2 are equal and opposite. Therefore, the function trends towards zero.

Now can be concluded that when the imaginary part of each mode has an opposite sign and the phase is not necessarily out of phase then, when the modes add, anti-resonance does not occur. So, each measurement can have anti-resonances or no anti- resonances depending on the direction of the imaginary part of the frequency response function.

4.4 Accurate measurements

According to the *'Engineer's Guide to Accurate Sensor Measurements'* by National Instruments [7], accurate sensor measurements depend on a few signal conditioning requirements. After acquiring the data, additional signal processing must be performed in order to display the date in a more meaningful format. Vibration signals are commonly converted to the frequency spectrum, which shows how the signal change over a range of frequencies.

Signal amplification

The amplification of the signal produced by the accelerometer is important for the accuracy of the measurement, because it is very susceptible to noise. The reason for this is that the charge produced by an accelerometer is very small, therefore a low amount of noise can heavily influence this signal. The solution for this problem is to either choose sensitive accelerometer, with integrated amplifier, or use an external amplifier with noise reduction.

The most commonly used sensor for vibration measurements is the piezoelectric accelerometer. This type of sensor uses the piezoelectric effect to measure changes in acceleration by converting it to an electrical charge. This conversion is possible because a piezoelectric material will output a voltage when is being deformed. A change in this voltage output means a change in acceleration of the object the sensor is attached to. Since piezoelectric accelerometers are high-impedance sources, a charge-sensitive amplifier with low noise, a high input impedance, and a low output impedance must be designed.

A subclass of the piezoelectric accelerometers, called the Integrated Electronic Piezoelectric sensors (IEPE), integrate the charge amplifier or voltage amplifier close to the sensor to ensure better noise immunity and more convenient packaging. However, these sensors require 4 – 20 mA current excitation to operate the circuitry inside them.

AC Coupling

Enabling IEPE signal conditioning generates a DC voltage offset equal to the product of the excitation current and sensor impedance. The signal acquired from the sensor thus consists of both AC and DC components. AC coupling or capacitive coupling involves using a capacitor to filter out the DC signal component from a signal with both AC and DC components [8]. AC coupling is useful because the DC component of a signal acts as a voltage offset. Removing it from the signal can increase the resolution of signal measurements and the usable dynamic range of the channel. When implemented in software, AC coupling can remove erroneous DC data that invalidates signal processing integration and measurement results. AC coupling also attenuates the long-term DC drift that sensors have due to age and temperature effect.

Grounding

Improper grounding of the accelerometer can result in an increase in noise. This is a direct result from the created ground loops and can be countered by grounding either the system input or the accelerometer. If the sensor is grounded, it must be connected differentially. If the sensor is floating, the input of the measurement system should be connected to the ground.

Anti-aliasing filters

Aliasing is a very common problem which occurs during the discrete sampling of a continuous system. This problem can be explained as having too few time instances at which a measurement is taken.

According to the Nyquist-Shannon sampling theorem, the highest frequency that can be analysed is the Nyquist frequency. Any frequency greater than the Nyquist frequency, appears as a frequency between zero and the Nyquist frequency after sampling. Without detailed knowledge of the original signal, these frequencies cannot be distinguished from frequencies that actually lie between zero and the Nyquist frequency.

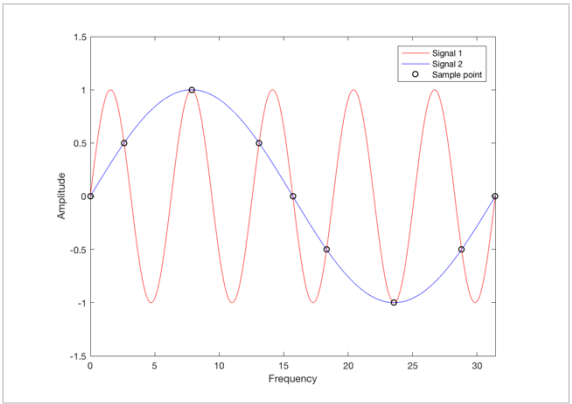


Figure 4.7: Aliasing.

Figure 4.7 shows the result of aliasing. In this figure, the red signal is the original signal that should be sampled, while the blue signal is the output signal after sampling. The figure clearly shows that both signals are very different.

Two solutions exist for the problem of aliasing, namely: choosing a higher sampling rate, or making use of a low-pass filter. The sampling rate should be at least twice the Nyquist frequency, to prevent aliasing. A low-pass filter only lets the low frequency content of the signal go through. The high frequency content will be blocked, which is most of the time the noise.

Dynamic range

The dynamic range of a sensor is the ratio between the largest and smallest signals a device can measure. This is of importance, because if the signal is too small or too large, the sensor cannot measure the signal. The dynamic range is a direct property of the used sensor and can therefore only be influenced by choosing a different sensor.

Maintaining signal quality

When very long cables are used, the added capacitance in the cable can affect the frequency response of the sensor by filtering some of the high-frequency content. In addition, noise and distortion may seep into the measurement signal if there is insufficient current to drive the cable capacitance. In general, cables should not be longer than 30 m if the frequency range is larger than 10 kHz.

The effect of long cables, can be experimentally determined by analysis of the high-frequency electrical characteristics. The procedure is as follows:

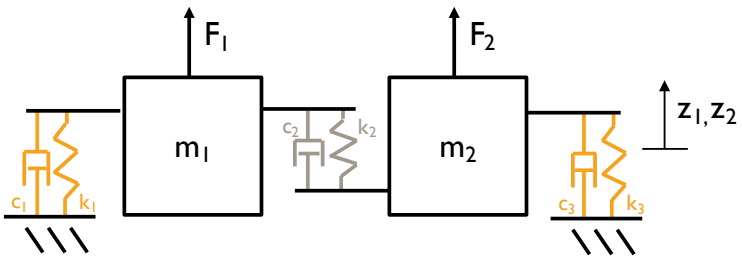
- Use a function generator to supply the maximum amplitude of the expected signal into a unity-gain, low-output impedance amplifier in series with the sensor.
- Compare the ratio of the original signal to the ratio of the signal measured on the scope. If the signal is attenuated, then the current used to drive the signal must be increased until a ratio of 1:1 is achieved.
- Care should be taken not to supply excessive current over short cable runs or when testing at elevated temperatures. Any current not used by the cable is used to power the internal electronics, and creates heat that might cause the sensor to exceed its maximum temperature specification.

Chapter 5 Theoretical modal analysis

5.1 Kinetics: Force and moment balances

The first step in the modal analysis is the derivation of the Equations of Motion of the system. These equations describe the motion of the system, as a result of an input force. There are several ways in which these equations can be derived. The Newton-Euler method and Lagrange's method are the two methods which are generally used for this derivation [3]. Both methods result in the same equations of motion and can therefore be used to check the derived equations of motion.

The two-body model of the factory floor, shown in Figure 5.1, will be derived on the right-side of the page, as an example of the modal analysis.



$$m_1 = 2\ kg \qquad k_1 = 6\ N/m$$
$$m_2 = 1\ kg \qquad k_2 = 3\ N/m$$
$$k_3 = 0\ N/m$$
$$c_1 = c_2 = c_3 = 0\ N \cdot s/m$$

Figure 5.1: Two-body model of a floor, problem definition.

5.2 Kinematics: Equations of motion

To get insight into the problem, a free-body-diagram must be drawn. This diagram consists of all bodies, disconnected from each other [3]. The forces present must be drawn to complete the problem definition.

Newton-Euler method

The applied forces can be summed by making use of Newton's second law of motion. These equations of summed forces are the force and moment balances and must be derived for each body individually:

$$\Sigma \boldsymbol{F} = m\ddot{\boldsymbol{z}} \quad and \quad \Sigma \boldsymbol{M} = I\boldsymbol{\alpha}$$

Because the two-body model only has motion in the z-direction, the moment balances do not have to be derived.

Dynamic model of a floor

The problem solved here is the simplified model of a floor. The floor is divided into three bodies, each connected to each other by means of a spring and damper.

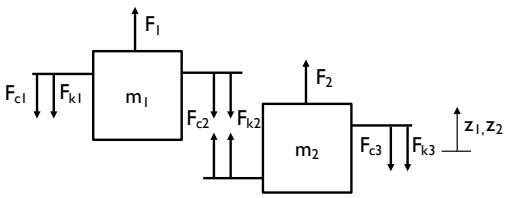


Figure 5.2: Free-body-diagram floor model.

Summation of the forces on the first body:

$$\Sigma F_1 = F_1 - c_1\,\dot{z}_1 - c_2\,(\dot{z}_1 - \dot{z}_2) - k_1\,z_1 - k_2\,(z_1 - z_2) = m_1\,\ddot{z}_1$$

Substitution of Hooke's law for springs and the equation for viscous damping completes the moment and force balances:

$$F_k = k \cdot \Delta z \quad and \quad F_c = c \cdot \Delta \dot{z}$$

The equations of motion follow from these moment and force balances. Rewriting them into the standard matrix form:

$$\boldsymbol{M}\,\ddot{\boldsymbol{z}} + \boldsymbol{C}\,\dot{\boldsymbol{z}} + \boldsymbol{K}\,\boldsymbol{z} = \boldsymbol{F}(t)$$

As explained above, the second method to derive the equations of motion is Lagrange's method. This method can be used the check whether the equations of motion are correct, but is not necessary for the modal analysis.

Summation of the forces on the second body:

$$\Sigma F_2 = F_2(t) + c_2\,(\dot{z}_1 - \dot{z}_2) - c_3\,(\dot{z}_2 - \dot{z}_3) + k_2\,(z_1 - z_2) - k_3\,(z_2 - z_3) = m_2\,\ddot{z}_2$$

Rewriting the force balances in standard matrix form:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\begin{Bmatrix} \ddot{z}_1 \\ \ddot{z}_2 \end{Bmatrix} + \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 + c_3 \end{bmatrix}\begin{Bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{Bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix}\begin{Bmatrix} z_1 \\ z_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

Substitution of the numerical values for the mass, stiffness and damping coefficients:

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\begin{Bmatrix} \ddot{z}_1 \\ \ddot{z}_2 \end{Bmatrix} + \begin{bmatrix} 9 & -3 \\ -3 & 3 \end{bmatrix}\begin{Bmatrix} z_1 \\ z_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

Additional theory

Lagrange's method

The basis for Lagrange's method is Hamilton's principle:

$$\int_{t_1}^{t_2} [-\delta T + \delta V - \delta U]\,dt = 0$$

Where δT is the change in kinetic energy of the system, δV the change in potential energy and δU the change in virtual work as a result of external forces.

Lagrange's equation for a system with one degree of freedom is then defined as:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} + \frac{\partial V}{\partial q_i} + \frac{\partial \mathcal{F}}{\partial \dot{q}_i} = \frac{\delta W_i}{\delta q}, \qquad i = 1, \dots, N_{DOF}$$

With q the generalised coordinate, $\mathcal{F}$ Rayleigh's dissipation function and $\frac{\delta W_i}{\delta q}$ the generalised force.

The same result can be gained from Lagrange's method:

$$T = \frac{1}{2} m_1\,\dot{z}_1{}^2 + \frac{1}{2} m_2\,\dot{z}_2{}^2$$
$$V = \frac{1}{2} k_1\,z_1{}^2 + \frac{1}{2} k_2 (z_1 - z_2)^2 + \frac{1}{2} k_3\,z_2{}^2$$
$$\mathcal{F} = \frac{1}{2} c_1\,\dot{z}_1{}^2 + \frac{1}{2} c_2 (\dot{z}_1 - \dot{z}_2)^2 + \frac{1}{2} c_3\,\dot{z}_2{}^2$$
$$\delta W = F_1(t)\,\delta z_1 + F_2(t)\,\delta z_2$$

Because this is a system with two degrees of freedom, Lagrange's equation must be used twice, for $q_1 = z_1$ and $q_2 = z_2$:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{z}_1}\right) = m_1\,\ddot{z}_1, \qquad \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{z}_2}\right) = m_2\,\ddot{z}_2$$
$$\frac{\partial T}{\partial z_1} = 0, \qquad \frac{\partial T}{\partial z_2} = 0$$
$$\frac{\partial V}{\partial z_1} = k_1\,z_1 + k_2(z_1 - z_2), \qquad \frac{\partial V}{\partial z_2} = k_3\,z_2 - k_2(z_1 - z_2)$$
$$\frac{\partial \mathcal{F}}{\partial \dot{z}_1} = c_1\,\dot{z}_1 + c_2(\dot{z}_1 - \dot{z}_2), \qquad \frac{\partial \mathcal{F}}{\partial \dot{z}_2} = c_3\,\dot{z}_2 - c_2(\dot{z}_1 - \dot{z}_2)$$
$$\frac{\delta W_1}{\delta q} = F_1(t), \qquad \frac{\delta W_2}{\delta q} = F_2(t)$$

Substitution in Lagrange's equation results in the same equations of motion as was obtained from the Newton-Euler method.

5.3 Vibration eigenproblem: Natural frequencies and modes

The next problem is the derivation of the eigenvalues and the eigenvectors, also called free-vibrations natural mode shapes. The square-root of the eigenvalue results in the eigenfrequency, or natural frequency of the system. Excitation of the system in its natural frequency results in resonance. The shape of the system during the vibration is called the mode shape, described by the eigenvectors, or natural modes.

Harmonic oscillation

A common assumption for spring-damper systems is harmonic oscillation. This assumption can mathematically be described in the form of trigonometric functions or complex exponentials:

$$\boldsymbol{z}(t) = \boldsymbol{Z}\cos(\omega t - \alpha) \quad or \quad \boldsymbol{z}(t) = \boldsymbol{Z}\, e^{i\omega t}$$

Substitution of these functions in the equations of motion results in the eigenproblem [9]:

$$(-\omega^2\boldsymbol{M} + i\omega\,\boldsymbol{C} + \boldsymbol{K})\,\boldsymbol{Z}\, e^{i\omega t} = \boldsymbol{F}(t)$$

This is a problem of the form:

$$\boldsymbol{D}(\omega)\,\boldsymbol{Z} = \boldsymbol{0}$$

In this equation $\boldsymbol{D}(\omega)$ is the dynamic matrix and ω the natural frequencies.
If for now the applied force is assumed to be zero, the problem can be solved by equating the determinant of the dynamic matrix to zero:

$$det\big(\boldsymbol{D}(\omega)\big) = 0$$

Natural frequencies

The solutions of the eigenproblem are the eigenvalues of the system:

$$\omega_1{}^2, \omega_2{}^2, \dots, \omega_i{}^2$$

These eigenvalues are stored on the diagonal of the eigenvalue matrix:

$$\boldsymbol{K}_m = \begin{bmatrix} \omega_1{}^2 & 0 & 0 \\ 0 & \omega_2{}^2 & 0 \\ 0 & 0 & \omega_3{}^2 \end{bmatrix}$$

The eigenvalue matrix is called $\boldsymbol{K}_m$ here instead of $\boldsymbol{\Lambda}$, which is how it is usually denoted in the literature. As will later be explained, $\boldsymbol{K}_m$ is the modal stiffness matrix of the system.

From the eigenvalues, the natural frequencies or eigenfrequencies of the system can be calculated, which are just the square roots of the eigenvalues.

Taking the derivatives of the trigonometric functions for harmonic oscillation:

$$\boldsymbol{z}(t) = \boldsymbol{Z}\, e^{i\omega t}$$
$$\dot{\boldsymbol{z}}(t) = i\omega\,\boldsymbol{Z}\, e^{i\omega t}$$
$$\ddot{\boldsymbol{z}}(t) = -\omega^2\,\boldsymbol{Z}\, e^{i\omega t}$$

Substitution of these functions into the equations of motion:

$$\left(-\omega^2\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} - i\omega\begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 + c_3 \end{bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix}\right)\begin{Bmatrix} Z_1 \\ Z_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

Substitution of the numerical values for the mass, stiffness and damping coefficients:

$$\left(-\omega^2\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 9 & -3 \\ -3 & 3 \end{bmatrix}\right)\begin{Bmatrix} Z_1 \\ Z_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

The damping was assumed to be zero here. This greatly simplifies the derivation and is therefore a favourable assumption. Care should be taken, because this assumption is not always rectifiable, especially not for factory floors.

The dynamic matrix can be derived as:

$$\boldsymbol{D}(\omega) = \left(-\omega^2\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 9 & -3 \\ -3 & 3 \end{bmatrix}\right)$$

Solving the eigenproblem, by equating the determinant of the dynamic matrix to zero:

$$det\left(\begin{bmatrix} 9 - 2\omega^2 & -3 \\ -3 & 3 - \omega^2 \end{bmatrix}\right) = 0$$

Resulting in:

$$(9 - 2\omega^2)(3 - \omega^2) - (-3)(-3) = 0$$

Solving this equation results in the eigenvalues of the system:

$$\omega_1{}^2 = 1.5, \qquad \omega_2{}^2 = 6.0$$

The eigenvalue matrix then becomes:

$$\mathbf{K}_\mathrm{m} = \begin{bmatrix} \omega_1{}^2 & 0 \\ 0 & \omega_2{}^2 \end{bmatrix} = \begin{bmatrix} 1.5 & 0 \\ 0 & 6.0 \end{bmatrix}$$

And the natural frequencies of the system:

$$\omega_1 = \sqrt{1.5}, \qquad \omega_2 = \sqrt{6.0}$$

Natural modes

The derivation of the natural modes or mode shapes is based on the eigenvectors of the system.

Three methods exist for the derivation of the natural modes. The first method is the easiest method when calculating the modes by hand. While the second method is more convenient when making use of MATLAB to calculate the eigenvectors. The third methods applies the MATLAB function `eig()` to solve the eigenproblem.

Remark: not all vibrational damping problems are solvable this way, only proportionally damped system. This will be explained during the natural coordinate transformation.

First method

The first problem is based on the eigenproblem, used to calculate the natural frequencies:

$$\boldsymbol{D}(\omega)\,\boldsymbol{Z} = \boldsymbol{0}$$

By back-substitution of the natural frequencies in the dynamic matrix, the natural modes can be calculated. This results in a system of equations, where each coordinate of the natural mode must be solved:

$$(-{\omega_n}^2\boldsymbol{M} + i\omega_n\boldsymbol{C} + \boldsymbol{K})\,\boldsymbol{V}_n = \boldsymbol{0}$$

Scaling of the natural modes is achieved by equating a single coordinate to one:

$$v_i = 1$$

This system must then be solved for each natural frequency ω_n independently and results in a natural mode belonging to each frequency:

$$\boldsymbol{V}_1 = \begin{Bmatrix} v_{11}=1 \\ v_{21} \\ v_{31} \end{Bmatrix}, \qquad \boldsymbol{V}_2 = \begin{Bmatrix} v_{12}=1 \\ v_{22} \\ v_{32} \end{Bmatrix}, \qquad \boldsymbol{V}_3 = \begin{Bmatrix} v_{13}=1 \\ v_{23} \\ v_{33} \end{Bmatrix}$$

These vectors form the columns of the modal matrix $\boldsymbol{V}$:

$$\boldsymbol{V} = \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}$$

First method

Back substitution of the natural frequencies in the dynamic equations of motion results in two separate equations, one for each natural mode:

$$\begin{bmatrix} 9-2\left(\sqrt{1.5}\right)^2 & -3 \\ -3 & 3-\left(\sqrt{1.5}\right)^2 \end{bmatrix} \begin{Bmatrix} v_{11} \\ v_{21} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

$$\begin{bmatrix} 9-2\left(\sqrt{6.0}\right)^2 & -3 \\ -3 & 3-\left(\sqrt{6.0}\right)^2 \end{bmatrix} \begin{Bmatrix} v_{12} \\ v_{22} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

The natural modes can then be calculated:

$$\boldsymbol{V}_1 = \begin{Bmatrix} v_{11}=1 \\ v_{21} \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}$$

$$\boldsymbol{V}_2 = \begin{Bmatrix} v_{12}=1 \\ v_{22} \end{Bmatrix} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}$$

This results in the modal matrix:

$$\mathbf{V} = \begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}$$

To satisfy the orthogonality conditions, the matrix must be scaled:

$$\boldsymbol{V}_m = \frac{\begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}}{\sqrt{\begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}^T \cdot \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}}} = \begin{bmatrix} 0.4082 & 0.5774 \\ 0.8165 & -0.5774 \end{bmatrix}$$

In the next section, the orthogonality conditions will be explained. To be able to satisfy these requirements, the modal matrix must be scaled according to:

$${\boldsymbol{V}_m}^T\boldsymbol{M}\,\boldsymbol{V}_m = \boldsymbol{1}$$

This can be achieved by application of the following equation:

$$\boldsymbol{V}_m = \frac{\boldsymbol{V}}{\sqrt{\boldsymbol{V}^T \boldsymbol{M}\, \boldsymbol{V}}}$$

Second method

The second method is based on the dynamic matrix $\boldsymbol{D}(\omega)$ [10]. Substitution of the natural frequencies results in:

$$\boldsymbol{D}(\omega_i) = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix}$$

This matrix can then be partitioned, resulting in the equation for the scaled eigenvectors around the first coordinate:

$$\boldsymbol{V}_i = \begin{Bmatrix} 1 \\ -[\boldsymbol{D}_{bb}(\omega_i)]^{-1}\, \boldsymbol{D}_{ba}(\omega_i) \end{Bmatrix}$$

Where,

$$\boldsymbol{D}_{ba}(\omega_i) = \begin{bmatrix} d_{21} \\ d_{31} \end{bmatrix}, \qquad \boldsymbol{D}_{bb}(\omega_i) = \begin{bmatrix} d_{22} & d_{23} \\ d_{32} & d_{33} \end{bmatrix}$$

Finally, the natural modes can be stored in the modal matrix, where each column defines a natural mode:

$$\boldsymbol{V} = \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix}$$

Again, the matrix must be scaled to satisfy the orthogonality conditions.

Third method

The MATLAB function `eig()` is able to calculate the eigenvalues and eigenvectors from the mass and stiffness matrices:

```
[Vm,Km] = eig(K,M);
```

The resulting eigenvectors already satisfy the orthogonality conditions.

Second method

Substitution of the natural frequencies in the dynamic matrix:

$$\boldsymbol{D}\left(\omega_1 = \sqrt{1.5}\right) = \begin{bmatrix} 9-3 & -3 \\ -3 & 3-1.5 \end{bmatrix}$$

$$\boldsymbol{D}\left(\omega_2 = \sqrt{6.0}\right) = \begin{bmatrix} 9-18 & -3 \\ -3 & 3-6 \end{bmatrix}$$

Partitioning these matrices:

$$\boldsymbol{D}_{ba}(\omega_1) = -3, \qquad \boldsymbol{D}_{bb}(\omega_1) = 3 - 1.5 = 1.5$$

$$\boldsymbol{D}_{ba}(\omega_2) = -3, \qquad \boldsymbol{D}_{bb}(\omega_2) = 3 - 6 = -3$$

Calculation of the natural modes:

$$\boldsymbol{V}_1 = \begin{Bmatrix} 1 \\ -[1.5]^{-1} \cdot -3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}$$

$$\boldsymbol{V}_2 = \begin{Bmatrix} 1 \\ -[-3]^{-1} \cdot -3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}$$

Resulting in the same modal matrix as method 1:

$$\boldsymbol{V} = \begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}$$

Again, to satisfy the orthogonality conditions, the matrix must be scaled:

$$\boldsymbol{V}_m = \frac{\begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}}{\sqrt{\begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}^T \cdot \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix}}} = \begin{bmatrix} 0.4082 & 0.5774 \\ 0.8165 & -0.5774 \end{bmatrix}$$

Third method

Application of the MATLAB function `eig()`:

```
K = [9,-3;-3,3];
M = [2,0;0,1];

[Vm,Km] = eig(K,M);
```

This results in the eigenvalues and eigenvectors:

$$\mathbf{K}_\mathrm{m} = \begin{bmatrix} 1.5 & 0 \\ 0 & 6.0 \end{bmatrix}$$

$$\boldsymbol{V}_m = \begin{bmatrix} -0.4082 & -0.5774 \\ -0.8165 & 0.5774 \end{bmatrix}$$

The resulting eigenvalues are therefore equal for all methods. The modal matrix is minus the matrix from methods 1 and 2. Because the entire matrix is multiplied by minus one, this is of no importance.

Fitting of the mode shapes

The mode shapes of the system can be derived by looking at the natural modes of the system. Each natural mode is a description of the mode shape of the system, for the corresponding natural frequency, Figure 5.3.

The mode shapes can also be fitted using MATLAB, by making use of the 'Shape-Preserving Piecewise Cubic Interpolation'-algorithm. This type of algorithm divides the data into intervals. On each subinterval, the polynomial $P(x)$ is a cubic Hermite interpolating polynomial for the given data points with specified slopes at the interpolation points, Figure 5.4. This mode fit algorithm is part of the QuadraturePicking MATLAB function (Appendix H). In case of mode shapes of a surface, a 5^{th} degree polynomial can be used as the best fit for the mode shape. This is because the 'Shape-Preserving Piecewise Cubic Interpolation'-algorithm is not available for a surface fit. This mode fit algorithm is part of the QuadraturePicking2D MATLAB function (Appendix I). At least fifteen measurements are needed for a 5^{th} degree polynomial fit. If this is not the case, the 'Biharmonic Spline Interpolation'-algorithm can be used.

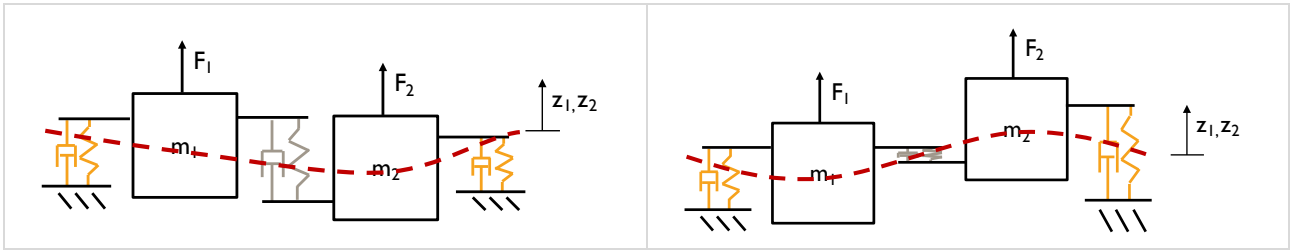


Figure 5.3: Mode shapes of the two-body system, derived from the natural modes.

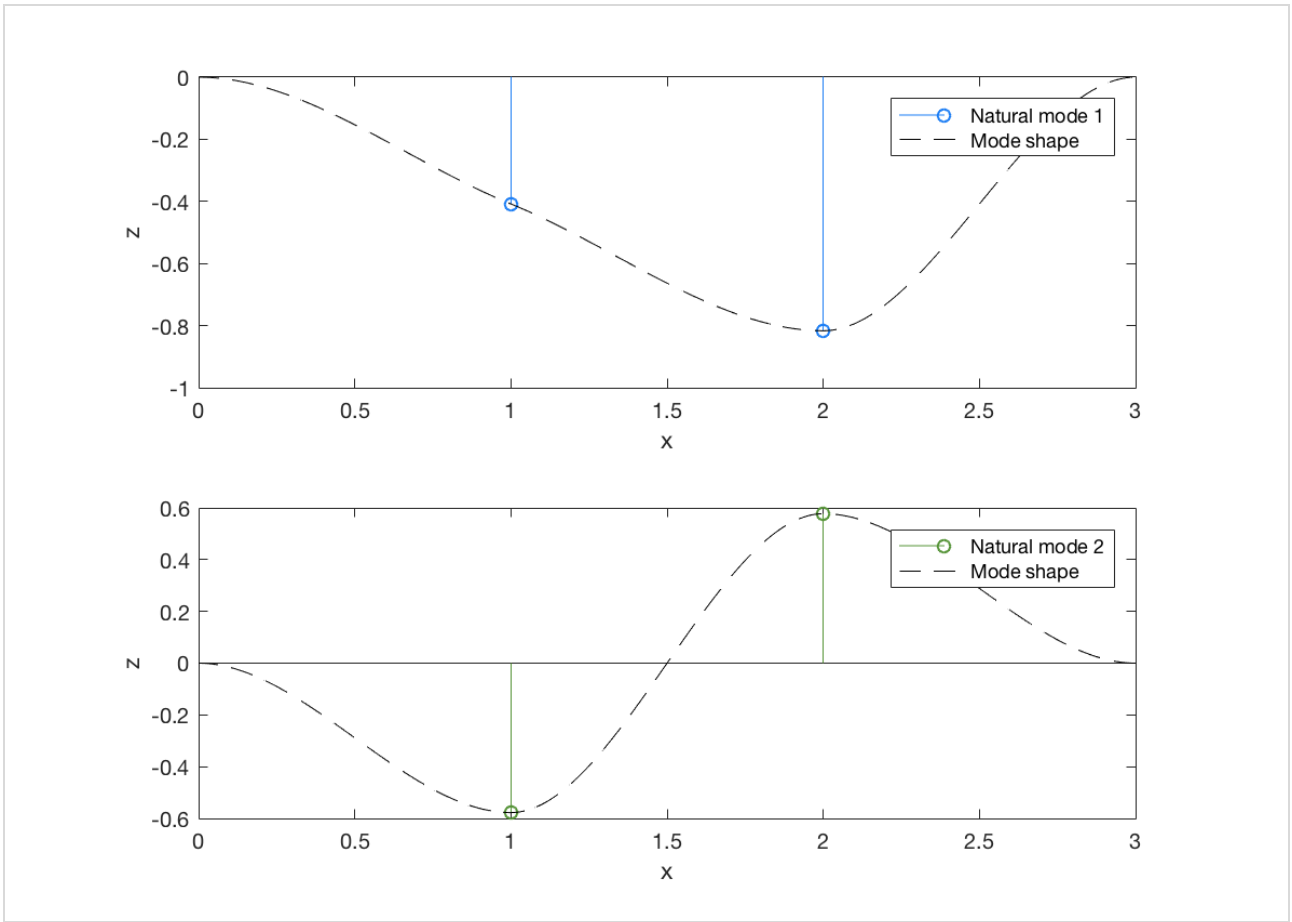


Figure 5.4: Mode shapes of the two-body system, calculated by MATLAB.

5.4 Frequency response function models

The frequency response function model is based on the implementation of the Fast Fourier Transform [11]. The first step is therefore the transformation of the equations of motion to the frequency domain.
The transformation from the time domain to the frequency domain is by means of the Laplace transformation. The Laplace transformation of a second order differential equation is done by substitution of the following Laplace variables, for the degrees of freedom:

$$\ddot{\boldsymbol{z}} = s^2\boldsymbol{Z}(s) = -\omega^2\,\boldsymbol{Z}(s)$$
$$\dot{\boldsymbol{z}} = s\,\boldsymbol{Z}(s) = i\omega\,\boldsymbol{Z}(s)$$
$$\boldsymbol{z} = \boldsymbol{Z}(s)$$

Substitution in the equations of motion of the system results in:

$$-\omega^2\,\boldsymbol{M}\,\boldsymbol{Z}(s) + i\omega\,\boldsymbol{C}\,\boldsymbol{Z}(s) + \boldsymbol{K}\,\boldsymbol{Z}(s) = \boldsymbol{F}(s)$$

Which is an equation of the form:

$$\boldsymbol{B}(s)\,\boldsymbol{Z}(s) = \boldsymbol{F}(s), \qquad where\ \boldsymbol{B}(s) = -\omega^2\,\boldsymbol{M} + i\omega\,\boldsymbol{C} + \boldsymbol{K}$$

Where $\boldsymbol{Z}(s)$ is the response of the system, $\boldsymbol{F}(s)$ is the applied force vector and $\boldsymbol{B}(s)$ is the system matrix.

Frequency response functions from equations of motion

The frequency response function is defined as either the response divided by the excitation force (compliance) or the excitation force divided by the response (stiffness).
Observation of the Laplace transformed equations of motion shows that:

$$\frac{\boldsymbol{Z}(s)}{\boldsymbol{F}(s)} = \boldsymbol{B}^{-1}(s) \qquad (compliance)$$
$$\frac{\boldsymbol{F}(s)}{\boldsymbol{Z}(s)} = \boldsymbol{B}(s) \qquad (stiffness)$$

Laplace transformation of the equations of motion for the two-body system:

$$\left(-\omega^2\begin{bmatrix}m_1 & 0\\ 0 & m_2\end{bmatrix} - i\omega\begin{bmatrix}c_1+c_2 & -c_2\\ -c_2 & c_2+c_3\end{bmatrix} + \begin{bmatrix}k_1+k_2 & -k_2\\ -k_2 & k_2+k_3\end{bmatrix}\right)\begin{Bmatrix}Z_1\\ Z_2\end{Bmatrix} = \begin{Bmatrix}F_1\\ F_2\end{Bmatrix}$$

Which is the same equation as the eigenproblem. Substitution of the numerical:

$$\left(-\omega^2\begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}9 & -3\\ -3 & 3\end{bmatrix}\right)\begin{Bmatrix}Z_1\\ Z_2\end{Bmatrix} = \begin{Bmatrix}F_1\\ F_2\end{Bmatrix}$$

The system matrix can be derived as:

$$\boldsymbol{B}(s) = \left(-\omega^2\begin{bmatrix}m_1 & 0\\ 0 & m_2\end{bmatrix} - \omega\begin{bmatrix}c_1+c_2 & -c_2\\ -c_2 & c_2+c_3\end{bmatrix} + \begin{bmatrix}k_1+k_2 & -k_2\\ -k_2 & k_2+k_3\end{bmatrix}\right)$$

Substitution of the numerical values for the mass, stiffness and damping coefficients:

$$\left(-\omega^2\begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}9 & -3\\ -3 & 3\end{bmatrix}\right)\begin{Bmatrix}Z_1\\ Z_2\end{Bmatrix} = \begin{Bmatrix}F_1\\ F_2\end{Bmatrix}$$

Rewriting this equation results in expressions for the compliance and the stiffness respectively:

$$\begin{Bmatrix}Z_1\\ Z_2\end{Bmatrix}\begin{Bmatrix}F_1\\ F_2\end{Bmatrix}^{-1} = \left(-\omega^2\begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}9 & -3\\ -3 & 3\end{bmatrix}\right)^{-1}$$

$$\begin{Bmatrix}Z_1\\ Z_2\end{Bmatrix}^{-1}\begin{Bmatrix}F_1\\ F_2\end{Bmatrix} = \left(-\omega^2\begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}9 & -3\\ -3 & 3\end{bmatrix}\right)$$

The response of the system to a unit force can then be derived as:

$$\begin{Bmatrix}Z_1\\ Z_2\end{Bmatrix} = \left(-\omega^2\begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}9 & -3\\ -3 & 3\end{bmatrix}\right)^{-1}\begin{Bmatrix}1\\ 0\end{Bmatrix}$$

$$\begin{Bmatrix}Z_3\\ Z_4\end{Bmatrix} = \left(-\omega^2\begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}9 & -3\\ -3 & 3\end{bmatrix}\right)^{-1}\begin{Bmatrix}0\\ 1\end{Bmatrix}$$

Plots of these responses show the four frequency response functions for this system. These plots are shown in Figure 5.5.

Therefore, taking the inverse of the system matrix results in the frequency response functions matrix $\boldsymbol{H}(s)$ of the compliance, which contains the frequency response functions from a point to all other points on the structure:

$$\boldsymbol{H}(s) = \boldsymbol{B}(s)^{-1}, \qquad where\ \boldsymbol{H}(s)\boldsymbol{F}(s) = \boldsymbol{Z}(s)$$

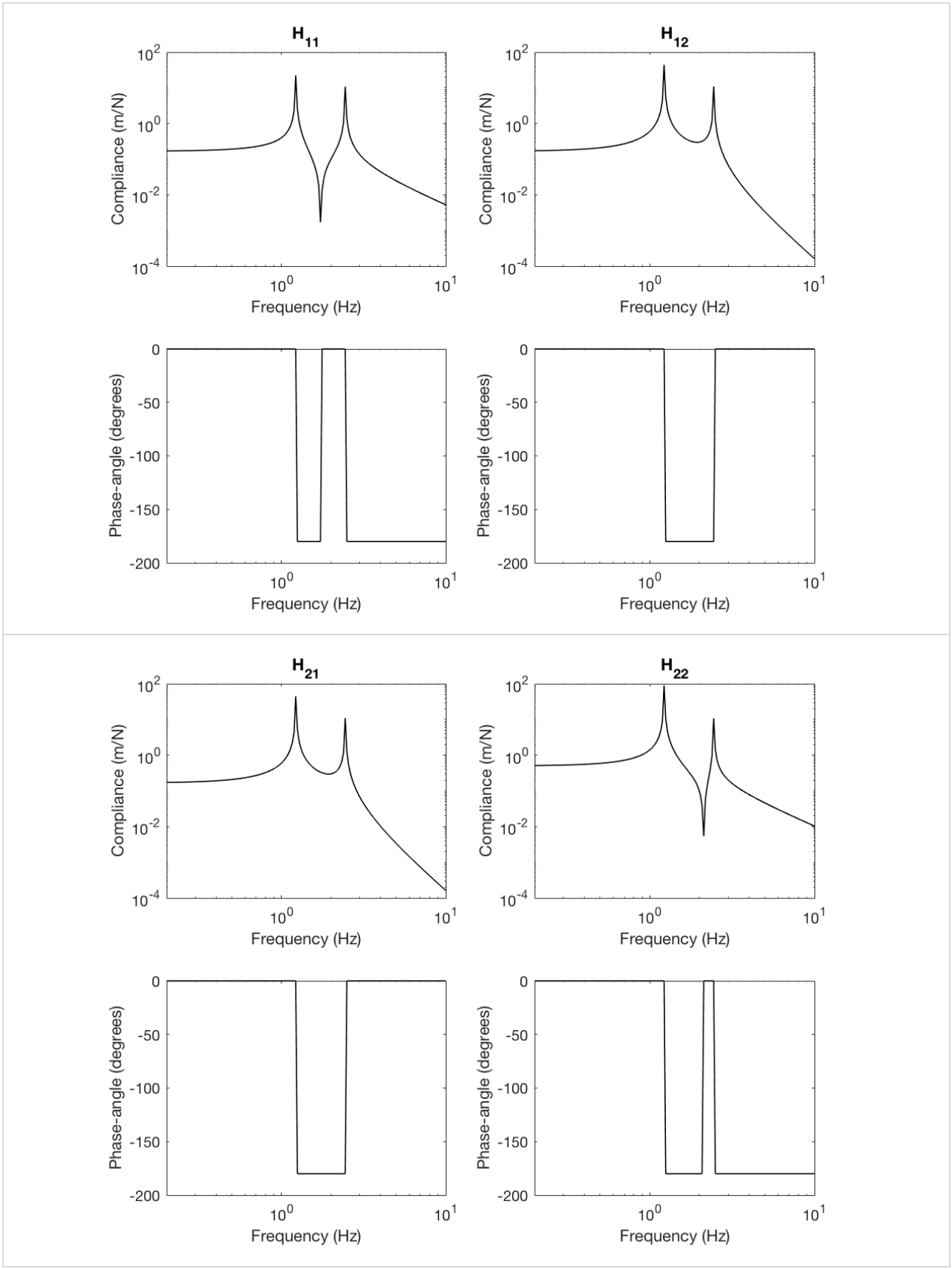


Figure 5.5: Frequency response functions of the two-body system.

Additional theory

The frequency response functions matrix can be calculated from the determinant $\|\boldsymbol{B}(s)\|$ and the conjugate $\overline{\boldsymbol{B}(s)}$ of $\boldsymbol{B}(s)$:

$$\boldsymbol{H}(s) = \frac{1}{\|\boldsymbol{B}(s)\|} \cdot \overline{\boldsymbol{B}(s)}$$

The frequency response functions matrix can then be separated into individual frequency response functions, which will all be polynomials with the same denominator:

$$\begin{Bmatrix} Z_1(s) \\ \vdots \\ Z_n(s) \end{Bmatrix} = \begin{bmatrix} h_{11}(s) & \dots & h_{1\mathrm{m}}(s) \\ \vdots & \ddots & \vdots \\ h_{\mathrm{n}1}(s) & \dots & h_{\mathrm{nm}}(s) \end{bmatrix} \cdot \begin{Bmatrix} F_1(s) \\ \vdots \\ F_{\mathrm{n}}(s) \end{Bmatrix}$$

Where,

$$h_{\mathrm{nm}}(s) = \frac{a_0\, s^{\mathrm{m}} + a_1\, s^{m-1} + \cdots + a_{\mathrm{m}}}{b_0\, s^{2\mathrm{n}} + b_1\, s^{2\mathrm{n}-1} + \cdots + b_{2\mathrm{n}}}$$

Frequency response functions from modal description

The derivation of the frequency response function model from the modal vectors is especially useful during the experimental modal analysis. The modal vectors are derived by means of quadrature picking (Chapter 7), after which the frequency response functions can be derived [12] [13].

The easiest way to derive the modal frequency response functions is by transformation to natural coordinates. This transformation is mainly used to uncouple the system, which means that each body can move independently. The uncoupling of the equations of motion is achieved by means of the multiplication with the modal matrix.

To transform the equations of motion to natural coordinates, the following equation must be substituted for the degrees of freedom:

$$\boldsymbol{z} = \boldsymbol{V}_m\, \boldsymbol{z}_m$$

Where $\boldsymbol{V}_m$ is the modal matrix and $\boldsymbol{z}_m$ the natural coordinate system. In the literature, $\boldsymbol{\eta}$ is used for the natural coordinates, but it was decided to use $\boldsymbol{z}_m$ instead.

Transformation of the equations of motion to natural coordinates results in:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} -0.41 & -0.58 \\ -0.82 & 0.58 \end{bmatrix} \begin{Bmatrix} \ddot{z}_{m,1} \\ \ddot{z}_{m,2} \end{Bmatrix} + \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 + c_3 \end{bmatrix} \begin{bmatrix} -0.41 & -0.58 \\ -0.82 & 0.58 \end{bmatrix} \begin{Bmatrix} \dot{z}_{m,1} \\ \dot{z}_{m,2} \end{Bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix} \begin{bmatrix} -0.41 & -0.58 \\ -0.82 & 0.58 \end{bmatrix} \begin{Bmatrix} z_{\mathrm{m},1} \\ z_{\mathrm{m},2} \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

Pre-multiplication of these equations with the transpose of the modal matrix and identification of the orthogonality conditions results in:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} \ddot{z}_{m,1} \\ \ddot{z}_{m,2} \end{Bmatrix} + \begin{bmatrix} 2.45\,\xi & 0 \\ 0 & 4.90\,\xi \end{bmatrix} \begin{Bmatrix} \dot{z}_{m,1} \\ \dot{z}_{m,2} \end{Bmatrix} + \begin{bmatrix} 1.5 & 0 \\ 0 & 6.0 \end{bmatrix} \begin{Bmatrix} z_{\mathrm{m},1} \\ z_{\mathrm{m},2} \end{Bmatrix} = \begin{Bmatrix} Q_1 \\ Q_2 \end{Bmatrix}$$

Where,

$$\boldsymbol{V}_m{}^T \boldsymbol{M}\, \boldsymbol{V}_m = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{V}_m{}^T \boldsymbol{K}\, \boldsymbol{V}_m = \begin{bmatrix} 1.5 & 0 \\ 0 & 6.0 \end{bmatrix}$$

$$\boldsymbol{V}_m{}^T \boldsymbol{C}\, \boldsymbol{V}_m = \begin{bmatrix} 2.45\,\xi & 0 \\ 0 & 4.90\,\xi \end{bmatrix}$$

$$\boldsymbol{V}_m{}^T \boldsymbol{F}(t) = \boldsymbol{Q}(t)$$

The results are the equations of motion in natural coordinates:

$$\boldsymbol{V}_m\, \boldsymbol{M}\, \ddot{\boldsymbol{z}}_m + \boldsymbol{V}_m\, \boldsymbol{C}\, \dot{\boldsymbol{z}}_m + \boldsymbol{V}_m\, \boldsymbol{K}\, \boldsymbol{z}_m = \boldsymbol{F}(t)$$

Pre-multiplication of these equations of motion with the transpose of the modal matrix results in:

$$\boldsymbol{V}_m{}^T \boldsymbol{M}\, \boldsymbol{V}_m\, \ddot{\boldsymbol{z}}_m + \boldsymbol{V}_m{}^T \boldsymbol{C}\, \boldsymbol{V}_m\, \dot{\boldsymbol{z}}_m + \boldsymbol{V}_m{}^T \boldsymbol{K}\, \boldsymbol{V}_m\, \boldsymbol{z}_m = \boldsymbol{V}_m{}^T \boldsymbol{F}(t)$$

Two orthogonality conditions can be identified here:

$$\boldsymbol{M}_m = \boldsymbol{V}_m{}^T \boldsymbol{M}\, \boldsymbol{V}_m = \mathbf{1}, \qquad \mathbf{K}_\mathrm{m} = \boldsymbol{V}_m{}^T \boldsymbol{K}\, \boldsymbol{V}_m = \boldsymbol{\omega_n}{}^2$$

These orthogonality conditions are a result of the distinct natural modes and will always result in the above identities. Substitution of the conditions results in the modal equations:

$$\ddot{\boldsymbol{z}}_m + \boldsymbol{V}_m{}^T \boldsymbol{C}\, \boldsymbol{V}_m\, \dot{\boldsymbol{z}}_m + \mathbf{K}_\mathrm{m}\, \boldsymbol{z}_m = \boldsymbol{V}_m{}^T\, \boldsymbol{F}(t)$$

There are still two factors left, the damping part of the equation and the applied forces. For the forces, the modal force vector can be defined:

$$\boldsymbol{Q}(t) = \boldsymbol{V}_m{}^T \boldsymbol{F}(t)$$

This results in:

$$\ddot{\boldsymbol{z}}_m + \boldsymbol{V}_m{}^T \boldsymbol{C}\, \boldsymbol{V}_m\, \dot{\boldsymbol{z}}_m + \mathbf{K}_\mathrm{m}\, \boldsymbol{z}_m = \boldsymbol{Q}(t)$$

Remark: until now it was assumed that the natural frequencies and modal matrix are perfectly available. During the experimental modal analysis, the natural frequencies and modal matrix are derived by means of quadrature picking. They are therefore never perfect and only an approximation of the reality. This problem will be addressed during the modal parameter estimation (Chapter 8).

Proportionally damping

For the damping, no general applicable formula can be defined however. Only for a single special case this is possible: proportionally damping.

Because the mass and stiffness matrices in natural coordinates are diagonal matrices, the system is only uncoupled when the damping matrix is also diagonal. Therefore, the damping factor $\boldsymbol{V}_m{}^T \boldsymbol{C}\, \boldsymbol{V}_m$ can be calculated to find out whether this is the case. The system is then called proportionally damped and can be solved using the above described techniques. Another way to find out whether a system is proportionally damped is by checking whether the damping matrix is a linear combination of the mass and stiffness matrices:

$$\boldsymbol{C} = a\boldsymbol{M} + b\boldsymbol{K}$$

Again, this results in a diagonal damping matrix. If the system is proportionally damped, the damping factor can be replaced with a diagonal matrix, where the diagonal terms are equal to $2\xi_i\omega_i$. The modal equations for a proportionally damped system, can now be written as:

$$\ddot{\boldsymbol{z}}_m + \boldsymbol{C}_m\, \dot{\boldsymbol{z}}_m + \mathbf{K}_\mathrm{m}\, \boldsymbol{z}_m = \boldsymbol{Q}(t), \qquad where\ \boldsymbol{C}_m = \boldsymbol{V}_m{}^T \boldsymbol{C}\, \boldsymbol{V}_m = \mathbf{2}\boldsymbol{\xi}\boldsymbol{\omega_n}$$

Natural equations of motion

$$\begin{aligned} \ddot{\boldsymbol{z}}_m + \boldsymbol{C}_m\, \dot{\boldsymbol{z}}_m + \boldsymbol{\omega_n}{}^2\, \boldsymbol{z}_m &= \boldsymbol{Q}(t) && (General) \\ \ddot{\boldsymbol{z}}_m + \boldsymbol{\omega_n}{}^2\, \boldsymbol{z}_m &= \boldsymbol{Q}(t) && (Undamped) \\ \ddot{\boldsymbol{z}}_m + \mathbf{2}\boldsymbol{\xi}\boldsymbol{\omega_n}\, \dot{\boldsymbol{z}}_m + \boldsymbol{\omega_n}{}^2\, \boldsymbol{z}_m &= \boldsymbol{Q}(t) && (Proportionally\ damped) \end{aligned}$$

Like the derivation of the frequency response functions from the equations of motion, the modal frequency response functions can be derived by means of Laplace transformation of the natural equations of motion:

$$-\omega^2\, \boldsymbol{Z}_m(s) + i\omega\, \boldsymbol{C}_m\, \boldsymbol{Z}_m(s) + \mathbf{K}_\mathrm{m}\, \boldsymbol{Z}_m(s) = \boldsymbol{Q}(s)$$

Rewriting this equation results in the modal equations of motion in natural coordinates:

$$(-\omega^2\, \boldsymbol{I} + i\omega\, \boldsymbol{C}_m + \boldsymbol{K}_m)\, \boldsymbol{Z}_m(s) = \boldsymbol{Q}(s)$$

Observation of the Laplace transformed equations of motion shows that:

$$\frac{\boldsymbol{Z}_m(s)}{\boldsymbol{Q}(s)} = (-\omega^2\, \boldsymbol{I} + i\omega\, \boldsymbol{C}_m + \boldsymbol{K}_m)^{-1} \quad (compliance)$$

$$\frac{\boldsymbol{Q}(s)}{\boldsymbol{Z}_m(s)} = (-\omega^2\, \boldsymbol{I} + i\omega\, \boldsymbol{C}_m + \boldsymbol{K}_m) \quad (stiffness)$$

To calculate the response of the system, the equation must be reverse transformed from modal coordinates to Cartesian coordinates:

$$\boldsymbol{Z}_m(t) = \boldsymbol{V}_m\, (-\omega^2\, \boldsymbol{I} + i\omega\, \boldsymbol{C}_m + \boldsymbol{K}_m)^{-1}$$

Second method

Another method for the derivation of the modal frequency response functions is based on the frequency response function model [14], derived from the equations of motion:

$$\boldsymbol{H}(s) = \boldsymbol{B}(s)^{-1}, \qquad where\ \boldsymbol{H}(s)\boldsymbol{F}(s) = \boldsymbol{Z}(s)$$

The orthogonality conditions derived during the transformation to natural coordinates and the approximation for the modal damping matrix can then be substituted.

Laplace transformation of the natural equations of motion:

$$\left(-\omega^2 \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + i\omega \begin{bmatrix}2.45\,\xi & 0\\ 0 & 4.90\,\xi\end{bmatrix} + \begin{bmatrix}1.5 & 0\\ 0 & 6.0\end{bmatrix}\right) \begin{Bmatrix}Z_{m,1}\\ Z_{m,2}\end{Bmatrix} = \begin{Bmatrix}Q_1\\ Q_2\end{Bmatrix}$$

Substitution of the numerical values for the mass, stiffness and damping coefficients:

$$\left(-\omega^2 \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}1.5 & 0\\ 0 & 6.0\end{bmatrix}\right) \begin{Bmatrix}Z_{m,1}\\ Z_{m,2}\end{Bmatrix} = \begin{Bmatrix}Q_1\\ Q_2\end{Bmatrix}$$

Rewriting this equation results in expressions for the compliance and the stiffness respectively:

$$\begin{Bmatrix}Z_{m,1}\\ Z_{m,2}\end{Bmatrix} \begin{Bmatrix}Q_1\\ Q_2\end{Bmatrix}^{-1} = \left(-\omega^2 \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}1.5 & 0\\ 0 & 6.0\end{bmatrix}\right)^{-1}$$

$$\begin{Bmatrix}Z_{m,1}\\ Z_{m,2}\end{Bmatrix}^{-1} \begin{Bmatrix}Q_1\\ Q_2\end{Bmatrix} = \left(-\omega^2 \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}1.5 & 0\\ 0 & 6.0\end{bmatrix}\right)$$

The response of the system to a unit force can then be derived as:

$$\begin{Bmatrix}Z_1\\ Z_2\end{Bmatrix} = \begin{bmatrix}-0.41 & -0.58\\ -0.82 & 0.58\end{bmatrix} \left(-\omega^2 \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}1.5 & 0\\ 0 & 6.0\end{bmatrix}\right)^{-1} \cdot \begin{bmatrix}-0.41 & -0.82\\ -0.58 & 0.58\end{bmatrix} \begin{Bmatrix}1\\ 0\end{Bmatrix}$$

$$\begin{Bmatrix}Z_3\\ Z_4\end{Bmatrix} = \begin{bmatrix}-0.41 & -0.58\\ -0.82 & 0.58\end{bmatrix} \left(-\omega^2 \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}1.5 & 0\\ 0 & 6.0\end{bmatrix}\right)^{-1} \cdot \begin{bmatrix}-0.41 & -0.82\\ -0.58 & 0.58\end{bmatrix} \begin{Bmatrix}0\\ 1\end{Bmatrix}$$

Plots of these responses show the four frequency response functions for this system. These plots are shown in Figure 5.7.

The orthogonality conditions are repeated here for clarity:

$$\boldsymbol{M}_m = \boldsymbol{V}_m{}^T \boldsymbol{M}\, \boldsymbol{V}_m = \boldsymbol{1}, \qquad \boldsymbol{K}_m = \boldsymbol{V}_m{}^T \boldsymbol{K}\, \boldsymbol{V}_m = \boldsymbol{\omega_n}^2, \qquad \boldsymbol{C}_m = \boldsymbol{V}_m{}^T \boldsymbol{C}\, \boldsymbol{V}_m = \boldsymbol{2\xi\omega_n}$$

This results in an equation which is dependent on the natural modes and natural frequencies:

$$\boldsymbol{B}(s)^{-1} = \left(-\omega^2\, \boldsymbol{V}_m\, \boldsymbol{V}_m{}^T + i\omega\, \boldsymbol{V}_m\, \boldsymbol{C}_m\, \boldsymbol{V}_m{}^T + \boldsymbol{V}_m\, \boldsymbol{K}_m\, \boldsymbol{V}_m{}^T\right)^{-1}$$

And a frequency response functions matrix of the form:

$$\boldsymbol{H}(s) = \boldsymbol{V}_m(-\omega^2 + i\omega\, \boldsymbol{C}_m + \boldsymbol{K}_m)^{-1}\boldsymbol{V}_m{}^T$$

Additional theory

Because the equation is still rather complex, each component of the frequency response function can be written as a summation:

$$\boldsymbol{H}_{ij}(\omega) = \sum_{k=1}^{N} \frac{{}_kV_{m,i}\ {}_kV_{m,j}}{(-\omega^2 + {}_kC_m\ i\omega + {}_kK_m)}$$

Where ω is the excitation frequency and ${}_kV_{m,i}$, ${}_kV_{m,j}$ the components of the k[th] natural mode. Which results in:

$$\boldsymbol{H}_{ij}(\omega) = \sum_{k=1}^{N} \frac{{}_kA_{ij}}{(-\omega^2 + {}_kC_m\ i\omega + {}_kK_m)}, \qquad where\ {}_kA_{ij} = {}_kV_{m,i}\ {}_kV_{m,j}$$

This frequency response function can then be written in matrix vector form (for two-degrees of freedom):

$$\begin{bmatrix} \boldsymbol{H}_{11}(\omega_1) \\ \vdots \\ \boldsymbol{H}_{11}(\omega_{max}) \end{bmatrix} = \begin{bmatrix} \left(-{\omega_1}^2 + C_{m,1}\ i\omega_1 + K_{m,1}\right) & \left(-{\omega_1}^2 + C_{m,2}\ i\omega_1 + K_{m,2}\right) \\ \vdots & \vdots \\ \left(-{\omega_{max}}^2 + C_{m,1}\ i\omega_{max} + K_{m,1}\right) & \left(-{\omega_{max}}^2 + C_{m,2}\ i\omega_{max} + K_{m,2}\right) \end{bmatrix} \begin{bmatrix} {}_1A_{11} \\ \vdots \\ {}_NA_{11} \end{bmatrix}$$

5.5 Single mode contribution

The single mode contribution or superposition principle says that, in case of an uncoupled system, a frequency response function can be defined as a sum of individual frequency response functions for each body. This concept can easily be derived from the equations of motion in frequency response function form:

$$\begin{Bmatrix} Z_1(s) \\ \vdots \\ Z_n(s) \end{Bmatrix} = \begin{bmatrix} h_{11}(s) & \dots & h_{1\mathrm{m}}(s) \\ \vdots & \ddots & \vdots \\ h_{\mathrm{n}1}(s) & \dots & h_{\mathrm{nm}}(s) \end{bmatrix} \cdot \begin{Bmatrix} F_1(s) \\ \vdots \\ F_{\mathrm{n}}(s) \end{Bmatrix}$$

Where each frequency response function in the frequency response functions matrix is a $N \times N$ matrix. This results in the following individual sub-equations for the responses of the system:

$$Z_1(s) = h_{11}(s)\ F_1(s) + \cdots + h_{\mathrm{nm}}(s)\ F_{\mathrm{n}}(s)$$

Each response can therefore be written as a sum of individual frequency response functions or single mode contributions.

Expansion of the response of the system to a unit force:

$$\begin{Bmatrix} Z_1 \\ Z_2 \end{Bmatrix} = \begin{bmatrix} -0.41 & -0.58 \\ -0.82 & 0.58 \end{bmatrix} \begin{bmatrix} -\frac{2}{3\,\omega^2} & 0 \\ 0 & -\frac{1}{6\,\omega^2} \end{bmatrix} \begin{Bmatrix} -0.41 \\ -0.82 \end{Bmatrix}$$

$$\begin{Bmatrix} Z_3 \\ Z_4 \end{Bmatrix} = \begin{bmatrix} -0.41 & -0.58 \\ -0.82 & 0.58 \end{bmatrix} \begin{bmatrix} -\frac{2}{3\,\omega^2} & 0 \\ 0 & -\frac{1}{6\,\omega^2} \end{bmatrix} \begin{Bmatrix} -0.58 \\ 0.58 \end{Bmatrix}$$

This results in four individual responses, from each body to a force on a single body:

$$Z_1 = 0.17\left(-\frac{2}{3\,\omega^2}\right) + 0.48\left(-\frac{1}{6\,\omega^2}\right)$$
$$Z_2 = 0.34\left(-\frac{2}{3\,\omega^2}\right) - 0.48\left(-\frac{1}{6\,\omega^2}\right)$$
$$Z_3 = 0.24\left(-\frac{2}{3\,\omega^2}\right) - 0.34\left(-\frac{1}{6\,\omega^2}\right)$$
$$Z_4 = 0.48\left(-\frac{2}{3\,\omega^2}\right) + 0.34\left(-\frac{1}{6\,\omega^2}\right)$$

These contributions correspond to a natural frequency of the system and therefore have a single resonance peak. Because the total system is a summation of the single mode contributions, the contributions are defined as standard transfer functions:

$$h = \frac{1}{(-\omega^2 + {}_kC_m\ i\omega + {}_kK_m)}$$

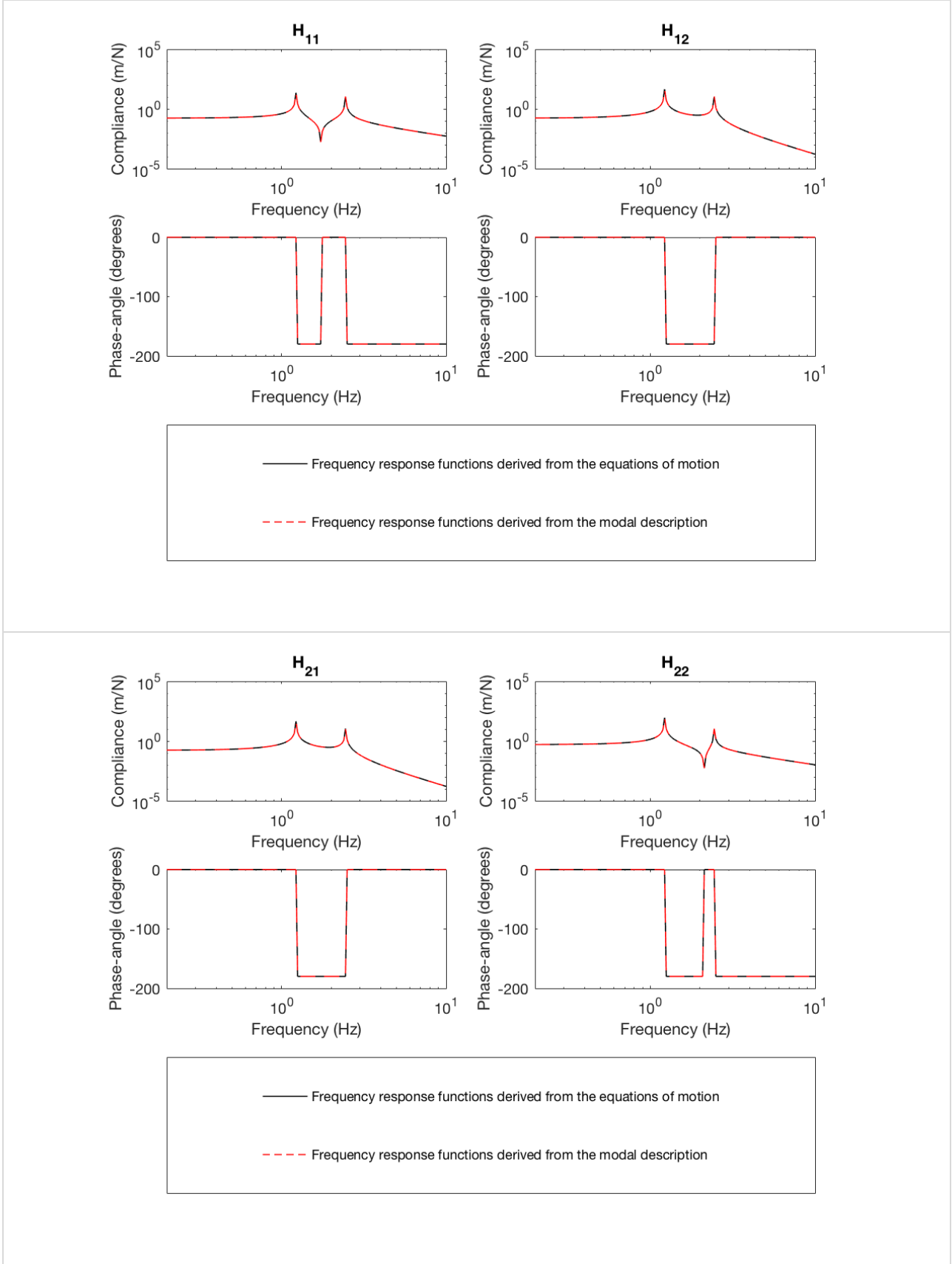


Figure 5.6: Modal frequency response functions of the two-body system.

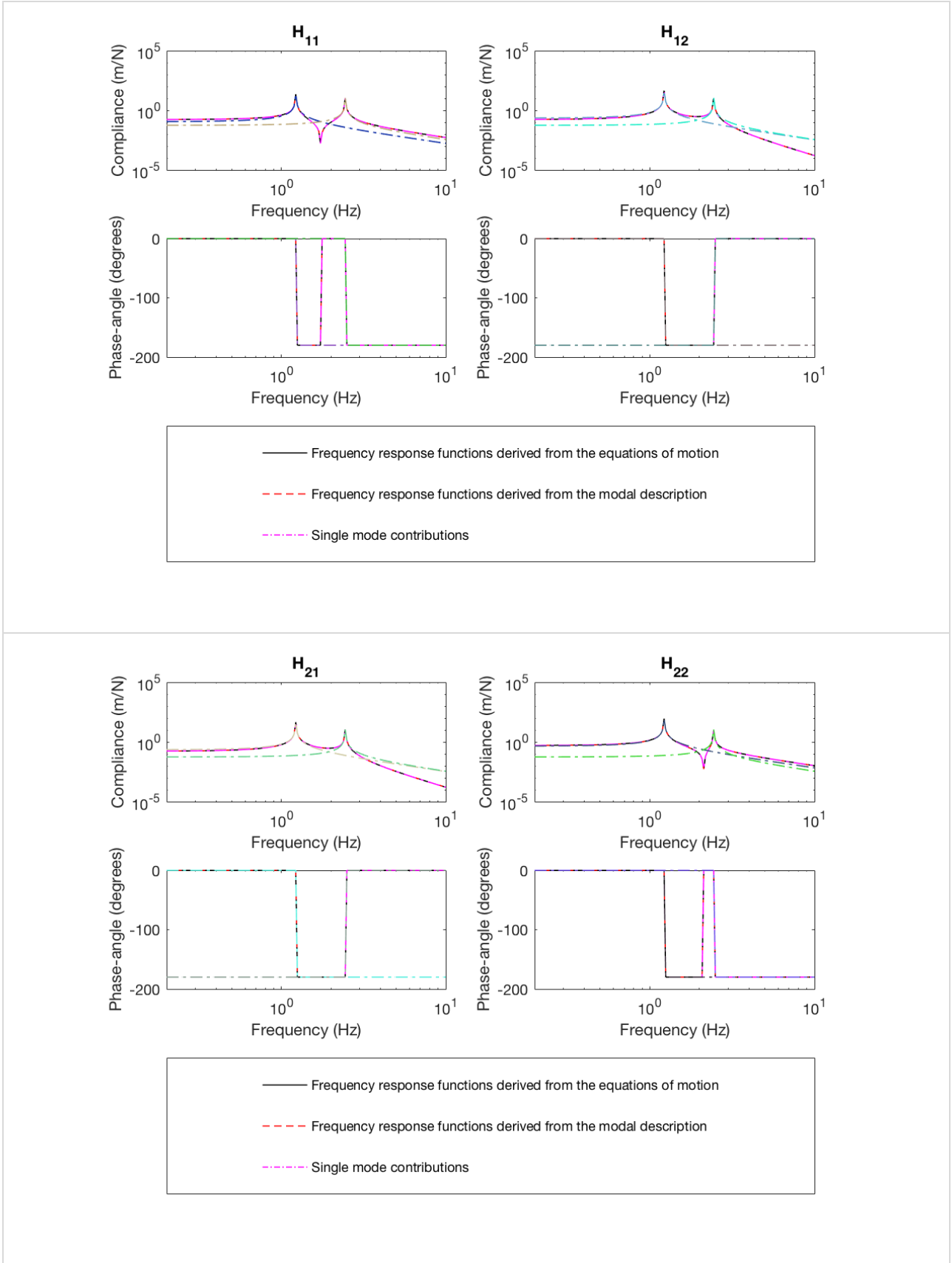


Figure 5.7: Single mode contributions for the frequency response functions of the two-body system.

Chapter 6 Experimental modal analysis

6.1 Method

The experimental modal analysis consists of measuring the vibration response of a system, after application of an excitation force. This excitation force can either be harmonic, applied by a shaker or an impulse applied by an excitation device, e.g. a shaker or an impact hammer.

First the measurement positions must be defined. This measurement grid defines the points where the acceleration sensors are placed and where the system is being excited. The vibration responses of the system are the frequency response functions from input excitation point to output measurement point. Which frequency response functions are measured depends on the type of experiment performed, as explained in section 7.1. For a roving hammer experiment, a single row of the matrix is measured, whereas a column of the matrix is measured in case of a roving sensor experiment. A choice must be made between performing a roving hammer or roving sensor experiment. This choice depends solely on which experiment is most convenient for the system, as it will give the same results for both experiments. Multiple acceleration sensors or shakers can be used to perform multiple roving hammer or sensor experiments respectively, at the same time.

Each measurement consists of ten excitations or 'hits' which can be averaged to improve the measurement and reduce the noise. Even if the coherence of the measurement is equal to one for all frequencies, multiple excitation must be measured and averaged, as explained in section 4.3.1.

The amplitude measured by the acceleration sensors and the input force measured by the acceleration sensors attached to the hammer or the built-in sensors of the shaker, are processed using SpecTest. The software calculates the frequency response functions from these input and output signals.

6.2 Conditions

Experimental setup

The setup for the experimental modal analysis is shown in Figure 6.1 for the roving hammer experiment and Figure 6.2 for the roving sensor experiment. These figures show the measurement grid for a beam being excited on bending and the frequency response functions that are being measured.

To set up the experiment, the measurement grid is marked on the system. The acceleration sensor(s) is/are positioned on these measurement points. The sensors, hammer and/or shaker are connected to the software and tested to confirm the proper working of the equipment. A high noise level indicates that something is wrong with the setup. The software is turned on and the experiment can be conducted. The filenames of the measurement data should contain the coordinates of both the input and output points, for easy access during the modal analysis.

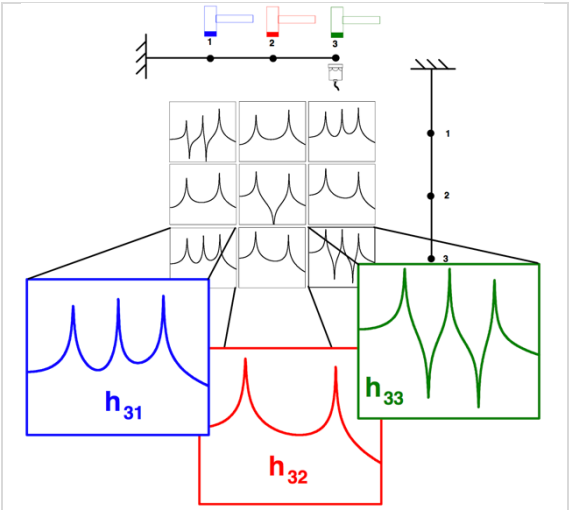


Figure 6.1: Roving hammer experiment [2].

Measurement equipment

The following equipment is necessary to conduct the experimental modal analysis:

- Acceleration sensors
- Impact hammer or shaker
- Data acquisition unit
- Software – MECAL SpecTest

When measuring the vibrations in a floor, the covering may block the output signal or apply extra damping to the system. To counteract this, a heavy block with spikes can be used as a platform on which the acceleration sensors are attached, Figure 6.3. The spikes go through the floorcovering, transmitting the vibrations directly to the acceleration sensors.

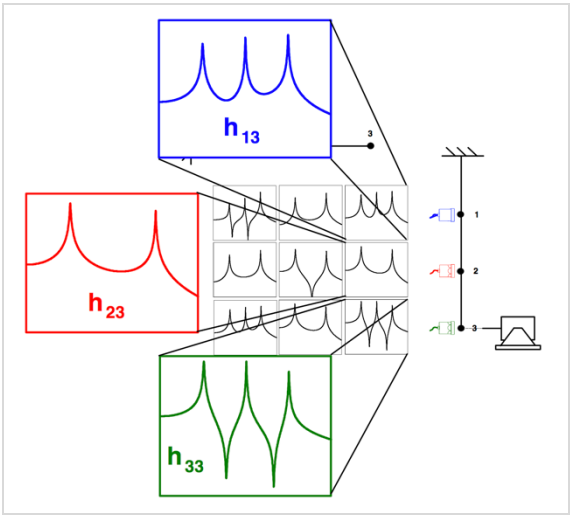


Figure 6.2: Roving sensor experiment [2].

Figure 6.3: Impact hammer (left), acceleration sensor (middle), steel block with spikes (right).

Additional considerations

During the definition of the measurement positions, care must be taken that the measurement grid represents the entire system and not only a sub-system. The reason for this is, when analysing the mode shapes of the system, modes may look identical [2]. This can be explained as the mode shapes of the sub-system looking identical, but with different mode shapes for the entire system. For example, if a machine support frame is measured that has a top and bottom plate, only measuring the top plate may result in these identical mode shapes. While the mode shape of the top plate look the same for both modes, the mode shape of the bottom plate probably differs. This mode shape of the bottom plate was not measured however and is therefore not visible during the modal analysis.

Another problem that might occur when defining the measurement grid is that measurement positions might be placed on a node of the system. These nodes are the points where the mode shapes pass the zero-amplitude line. Measurements of these nodes also result in zero amplitude. Most systems have standard mode shapes, these nodes can therefore be analysed beforehand, after which a measurement grid can be defined accordingly.

Chapter 7 Quadrature picking

During the theoretical modal analysis, it was assumed that the modal description of a system is available, so that the frequency response functions of the complete system can be derived. The next step in the modal analysis is therefore the derivation of the modal description from the measured data. To do this, a technique called quadrature picking is used. This is a method used to find the natural frequencies, natural modes and the damping of a system after an experimental modal analysis was performed [2]. The method involves first finding the resonance peaks of the measured frequency response functions, which give an indication for the natural frequencies of the system. The imaginary part of the frequency response functions can then be used to find the corresponding natural modes. In turn, these modes can be used to derive the system's damping coefficients, where the so called 'half-power points' serve as an indication for the magnitude of the damping.

The quadrature picking procedure will be explained by means of a simple, lightly damped, four-body model of a floor, Figure 7.1. The first row of the frequency response functions matrix is shown in Figure 7.2. It will be assumed that these functions where measured during an experimental modal analysis. While these are unrealistic results for an experimental modal analysis, it will make the explanation of the quadrature picking technique more clear.

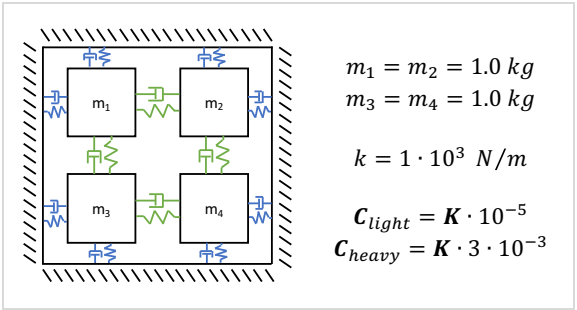


Figure 7.1: Four-body model of a floor, problem definition.

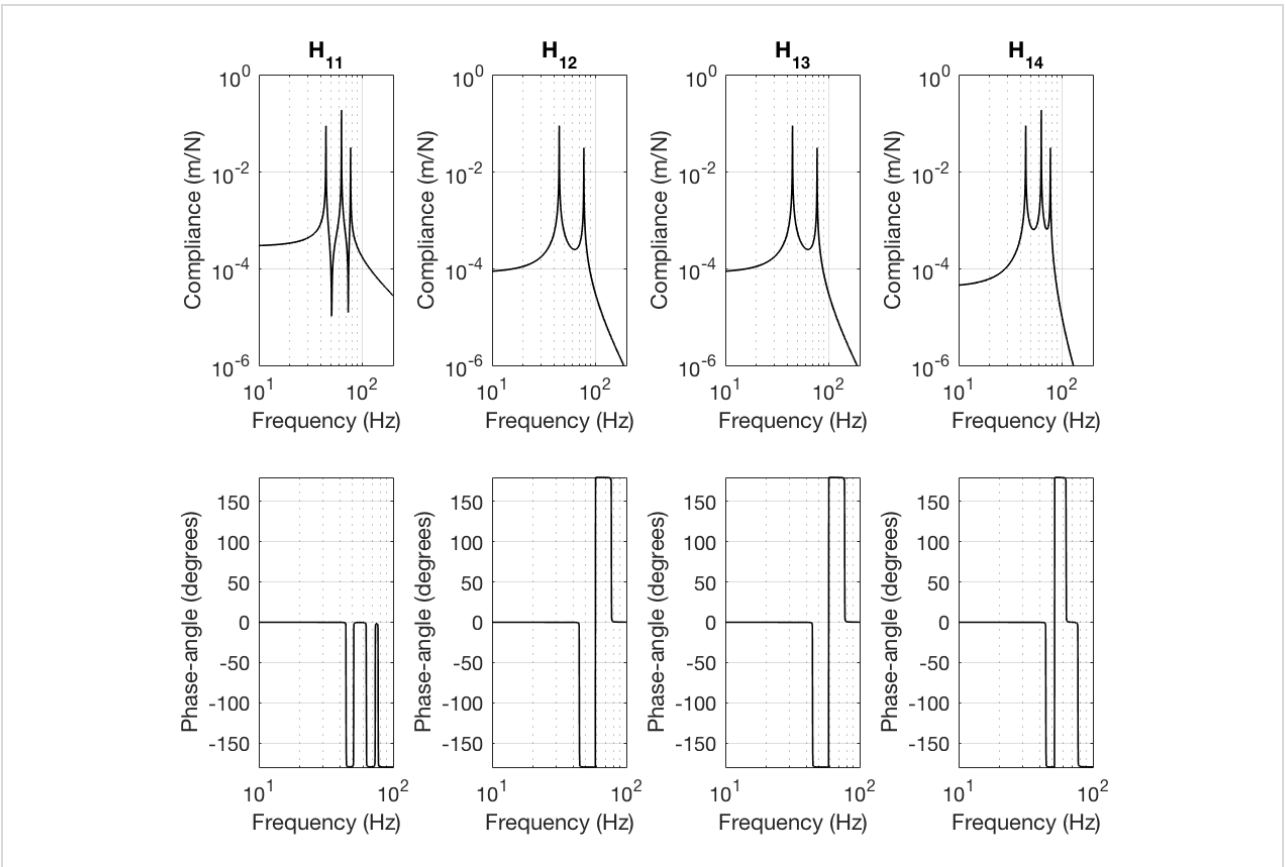


Figure 7.2: Frequency response functions of the four-body system.

The equations of motion can be derived as:

$$\begin{bmatrix} m_1 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 \\ 0 & 0 & m_3 & 0 \\ 0 & 0 & 0 & m_4 \end{bmatrix} \begin{Bmatrix} \ddot{z}_1 \\ \ddot{z}_2 \\ \ddot{z}_3 \\ \ddot{z}_4 \end{Bmatrix} + \begin{bmatrix} 4c & -c & -c & 0 \\ -c & 4c & 0 & -c \\ -c & 0 & 4c & -c \\ 0 & -c & -c & 4c \end{bmatrix} \begin{Bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \end{Bmatrix} + \begin{bmatrix} 4k & -k & -k & 0 \\ -k & 4k & 0 & -k \\ -k & 0 & 4k & -k \\ 0 & -k & -k & 4k \end{bmatrix} \begin{Bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix}$$

7.1 Frequency response functions matrix

The experimental modal analysis results in measurement data of the excitation force and the vibrations, in the form of accelerations, as discussed in 0. From this data, the frequency response functions from input force to output vibration can be calculated.

All frequency response functions will be stored in the frequency response functions matrix, Table 7.1. Each index of the matrix corresponds with the sensor and excitation positions of the measurement. The matrix has dimensions equal to the number of measurement points.
Care should be taken that, even for measurements on a plane, the matrix will always be 2-dimensional. This is a result of the definition of the modal matrix, where each mode vector forms a column of the matrix and each index of the mode vector contains the data of a point in the system. For a single roving sensor/hammer experiment, this means that one row or column of the frequency response functions matrix will be filled.

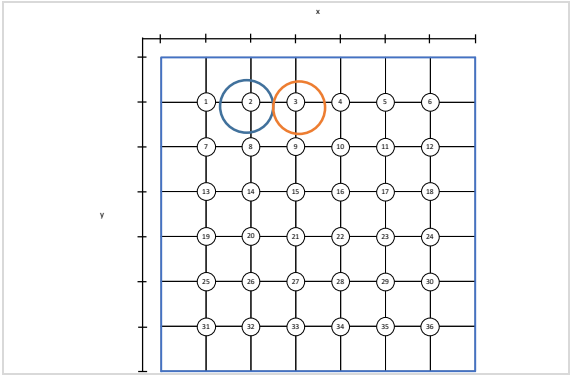


Figure 7.3: A floor divided into 36 measurement positions.

Roving sensor (excitation pos. 2) ↓	Hammer position						
Roving hammer (sensor pos. 3) → / Sensor position							
$\frac{z_1}{F_1}$	$\frac{z_1}{F_2}$	$\frac{z_1}{F_3}$	$\frac{z_1}{F_4}$	$\frac{z_1}{F_5}$	$\frac{z_1}{F_6}$		$\frac{z_1}{F_{36}}$
$\frac{z_2}{F_1}$	$\frac{z_2}{F_2}$	$\frac{z_2}{F_3}$	$\frac{z_2}{F_4}$	$\frac{z_2}{F_5}$	$\frac{z_2}{F_6}$		$\frac{z_2}{F_{36}}$
$\frac{z_3}{F_1}$	$\frac{z_3}{F_2}$	$\frac{z_3}{F_3}$	$\frac{z_3}{F_4}$	$\frac{z_3}{F_5}$	$\frac{z_3}{F_6}$		$\frac{z_3}{F_{36}}$
$\frac{z_4}{F_1}$	$\frac{z_4}{F_2}$	$\frac{z_4}{F_3}$	$\frac{z_4}{F_4}$	$\frac{z_4}{F_5}$	$\frac{z_4}{F_6}$	⋯	$\frac{z_4}{F_{36}}$
$\frac{z_5}{F_1}$	$\frac{z_5}{F_2}$	$\frac{z_5}{F_3}$	$\frac{z_5}{F_4}$	$\frac{z_5}{F_5}$	$\frac{z_5}{F_6}$		$\frac{z_5}{F_{36}}$
$\frac{z_6}{F_1}$	$\frac{z_6}{F_2}$	$\frac{z_6}{F_3}$	$\frac{z_6}{F_4}$	$\frac{z_6}{F_5}$	$\frac{z_6}{F_6}$		$\frac{z_6}{F_{36}}$
		⋮				⋱	⋮
$\frac{z_{36}}{F_1}$	$\frac{z_{36}}{F_2}$	$\frac{z_{36}}{F_3}$	$\frac{z_{36}}{F_4}$	$\frac{z_{36}}{F_5}$	$\frac{z_{36}}{F_6}$	⋯	$\frac{z_{36}}{F_{36}}$

Table 7.1: Frequency response functions matrix of the compliance.

Figure 7.3 shows a floor divided into 36 measurement positions. The frequency response functions matrix has dimensions: 36×36. If the complete system is to be measured, 1.296 measurements must be done! This shows that the amount of measurements needed grows quadratic and it quickly becomes impossible to measure the entire system.

Each column (blue) of the frequency response functions matrix shows the measurements for a constant excitation position (roving sensor), while each row (orange) shows the measurements for a constant sensor position (roving hammer). The driving point measurements (green) are the measurements where the response position is the same as the excitation position. These are located on the diagonal of the frequency response functions.

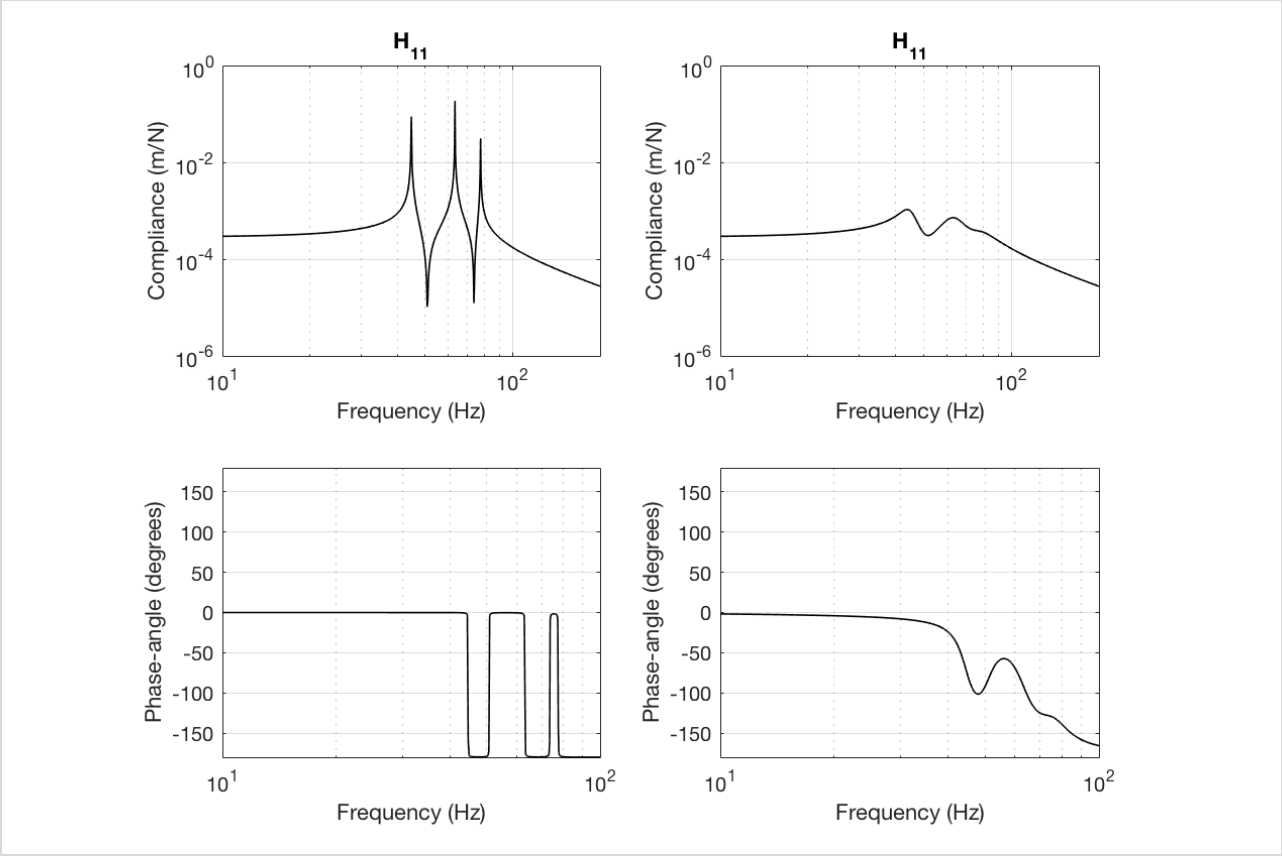


Figure 7.4: Visibility of the resonance peaks for the lightly damped (left) and heavily damped (right) cases of the four-body model.

7.2 Peak picking: Natural frequencies

The first step in quadrature picking is peak picking, where the resonance peaks are located and the corresponding natural frequency derived. For an undamped system, this is a rather straight forward method, because the peaks are clearly visible, Figure 7.4. This method however becomes increasingly difficult as the damping of the system increases, where peaks might completely disappear from the frequency response functions. Figure 7.4 shows this behaviour for the heavily damped case of the four-body system, where the third peak has almost 'disappeared'. Especially for these damped systems, the difficulty of peak picking lies in the identification of the resonance peaks.

One may also wonder: 'what happened to the fourth resonance peak?'. Because this system consists of four bodies, four resonance peaks should be visible. In this case, the third and fourth peaks have merged, therefore showing a single large peak, with an anti-resonance peak.

Several methods for the identification of the natural frequencies are available:

- Averaging the absolute value of all frequency responses functions in the frequency responses functions matrix.
- Averaging the imaginary part of the frequency responses functions divided by the real part.
- Derivation of a mode indicator function.

While the first two functions are useful, because they can also be used for quadrature picking, they may result in problems where not all modes may be identified and not all modes are genuine modes of the system [15]. In that case, the mode indicator functions can give more insight in the resonance peaks.

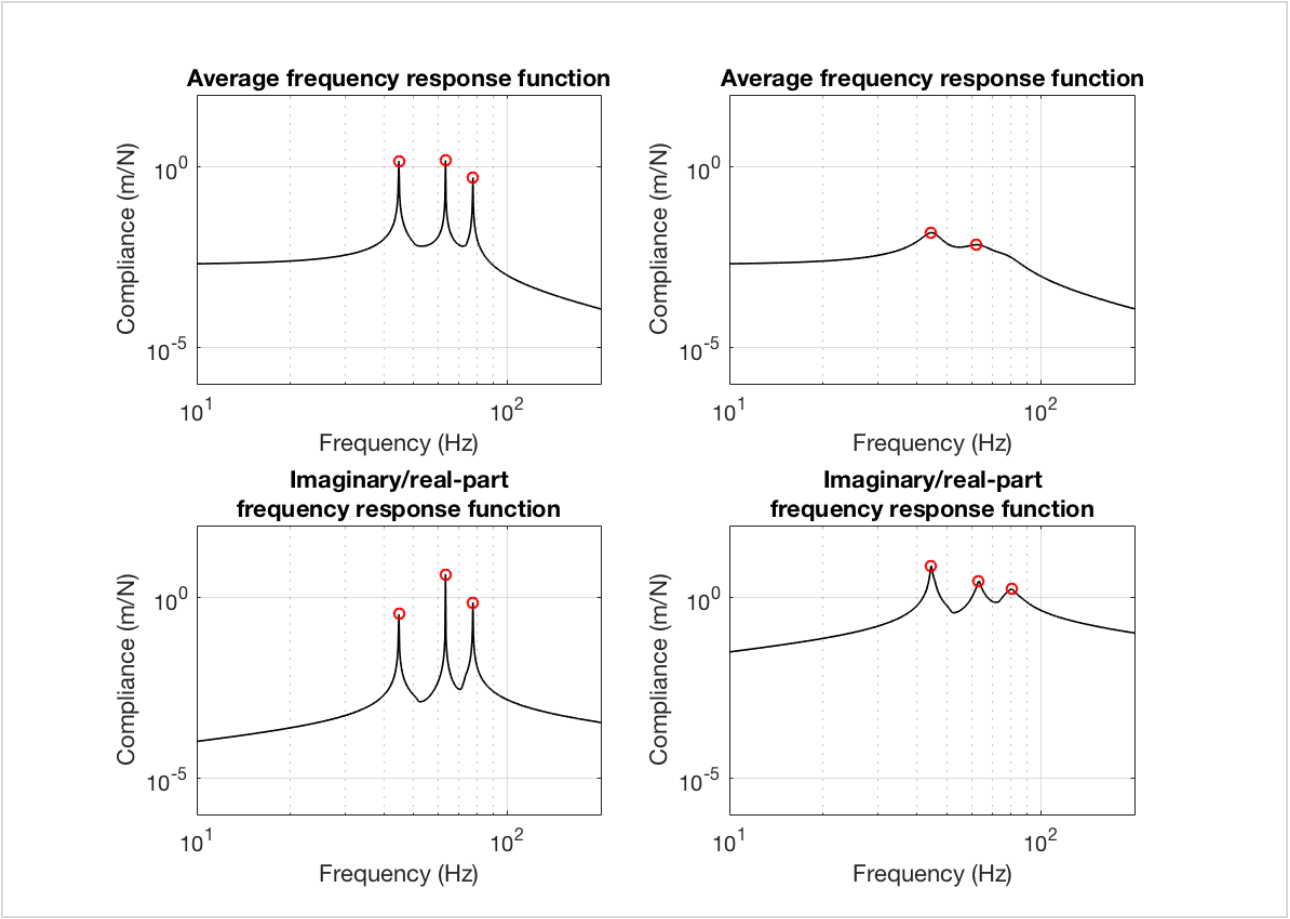


Figure 7.5: Mode indicator functions for the lightly damped (left) and heavily damped (right) cases of the four-body model.

7.2.1 Identification of the natural frequencies

The peak picking starts by analysis of the frequency response functions measured during the experimental modal analysis. These frequency response functions can either be the stiffness of the system (F/X) or the compliance (X/F). Analysis of each frequency response function individually may result in resonance peaks being overlooked, because the measurement location might coincide with a node of the corresponding mode shape. It is therefore necessary to sum all measurements to generate a single average frequency response function of the system:

$$H_{Avg}(\omega) = \sum_{j=1}^{M}\sum_{i=1}^{N}\left|H_{ij}(\omega)\right|$$

And for better amplification of the resonance peaks, the imaginary parts of the frequency response functions can be divided by the real parts:

$$H_{Im/Re}(\omega) = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N}\left|imag\left(H_{ij}(\omega)\right)\right|}{\sum_{j=1}^{M}\sum_{i=1}^{N}\left|real\left(H_{ij}(\omega)\right)\right|}$$

Figure 7.5 shows these functions for the measurement of the floor. The peaks of the function are indicated with red circles. It might be possible that not all circles indicate a natural frequency of the system. It is therefore important to carefully select the peaks of the functions manually to find the correct natural frequencies of the system.

7.2.2 Mode indicator functions

Several mode indicator functions exist [16], among these functions are the ordinary, multivariate and complex mode indicator functions. These three are the most commonly used functions for modal analysis. Basically, the mathematical formulation of the mode indicator function is that the real part of the frequency response function is divided by the magnitude of the frequency response function:

$$MIF(\omega) = \frac{\sum_{i=1}^{N} real\left(H_{ij}(\omega)\right) \cdot \left|H_{ij}(\omega)\right|}{\sum_{i=1}^{N}\left|H_{ij}(\omega)\right|^2} \qquad (Ordinary\ mode\ indicator\ function)$$

$$MMIF(\omega) = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N} real\left(H_{ij}(\omega)\right) \cdot \left|H_{ij}(\omega)\right|}{\sum_{j=1}^{M}\sum_{i=1}^{N}\left|H_{ij}(\omega)\right|^2} \qquad (Multivariate\ mode\ indicator\ function)$$

Because the real part rapidly passes through zero at resonance, the mode indicator function generally tends to have a much more abrupt change across a mode. The difference between the ordinary and the multivariate mode indicator functions is the amount of measurements used for the calculation of the function. For an ordinary mode indicator function, a single row or column is used, while for the multivariate multiple rows or columns are used. Of the indicator functions, the complex mode indicator function is the most accurate. It is based on the single value decomposition of the frequency response functions matrix to determine all the principal modes that are observed in the set of measurements. These complex mode indicator functions were not derived during this research, further research on this subject can improve the modal analysis.

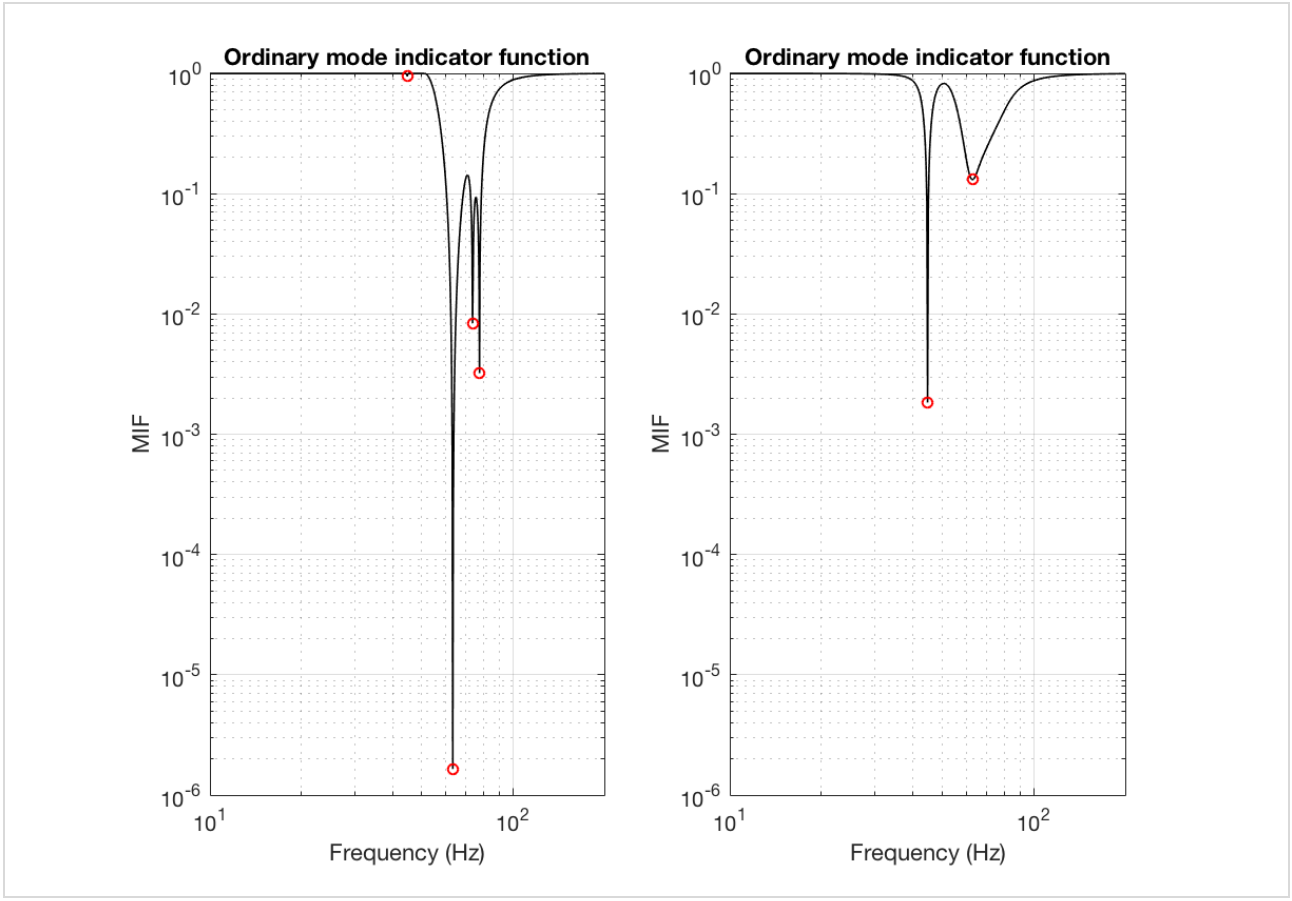


Figure 7.6: Ordinary mode indicator functions for the lightly damped (left) and heavily damped (right) cases of the four-body model.

The mode indicator function for the experimental modal analysis of the floor is shown in Figure 7.6. The function is equal to one for all frequencies, except for the natural frequencies where it exhibits local minima or maxima. For the lightly damped case, this mode indicator function shows the frequencies of all resonance peaks, including the fourth peak that was lost because of closely spaced peaks. For the heavily damped case, only two peaks are shown. The ordinary and multivariate mode indicator functions are not able to identify the resonance peaks in this case. The complex mode indicator function is therefore preferred when the system is heavily damped.

Making use of all identification functions ensures that no resonance peaks are missed because of close spacing or because of measurements taken on a node of the structure. The three figures together identify the four natural frequencies of the system: 44.75 Hz, 63.25 Hz, 73.50 Hz and 77.50 Hz.

A MATLAB GUI was made to interactively select the resonance peaks of a measurement. This program can be found in Appendix M.

7.3 Quadrature picking: Natural modes and damping

The quadrature picking method is shown in Figure 7.7. This figure shows that, by examination of the resonance peaks, the natural modes can be derived. The damping can be derived from the so-called half-power or bandwidth of the resonance peaks. This method is very easy to apply, but also comes with some limitations, as will be discussed at the end of this chapter.

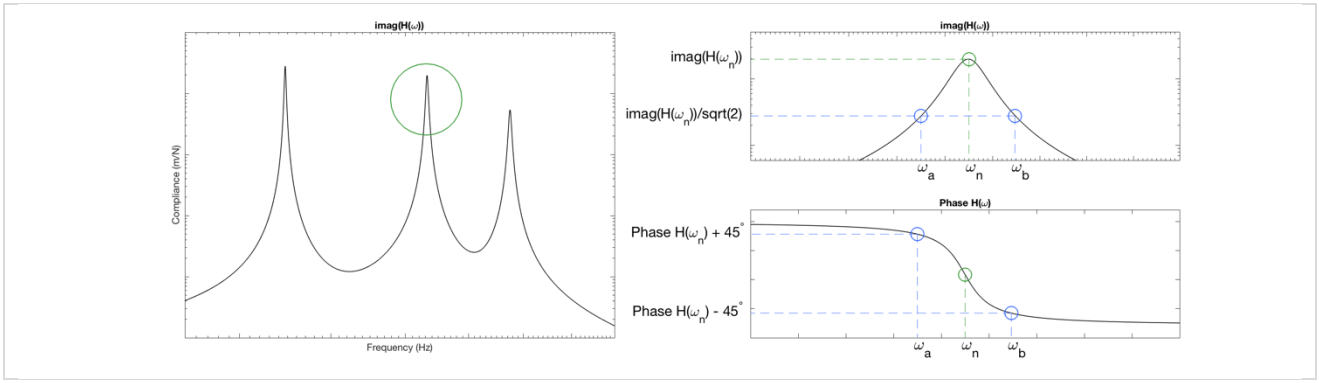


Figure 7.7: Quadrature picking.

7.3.1 Natural modes

The natural modes can be obtained from the imaginary part of the frequency response function. Most articles analyse the averaged absolute frequency response function of the system. This analysis does indeed result in the natural modes, however this results in absolute natural modes, where negative values in the mode vector are ignored. Therefore, it is better to take the imaginary part of the averaged frequency response function. This results in the correct natural modes, while still maintaining the negative values. As shown in Figure 7.7, the natural modes are just the magnitudes of the resonance peaks.

A surface fit of the first four mode shapes is shown in Figure 7.8. To get a better idea of how these mode shapes where constructed, line fits of the first two bodies can be plotted instead, Figure 7.9. These fits show how the modes where derived from the frequency response functions shown in Figure 7.2.

This modal analysis resulted in resonance peaks that are clearly visible and therefore easily indentified. If not all resonance peaks can be accuratly identified, it is better to leave them out of the modal matrix.

This can be done by adding zeros for the missing modes:

$$\boldsymbol{V} = \begin{bmatrix} v_{11} & 0 & 0 \\ v_{21} & 0 & 0 \\ v_{31} & 0 & 0 \end{bmatrix} \qquad (V_2 \; and \; V_3 \; not \; visible)$$

As a result of the way in which the single mode contributions are defined (5.5), these modes will be left out of the frequency response functions. This also means that the corresponding resonance peaks will be missing from the frequency response functions.
Finally, Figure 7.10 shows a so-called waterfall plot of the mode shapes. This plot gives extra insight on how the mode shapes are derived from the experimental modal analysis.

7.3.2 Damping matrix

The damping of each mode can be derived from the bandwidth or half-power of the resonance peak. The bandwidth of the peak is the intersection with the line at $\frac{1}{\sqrt{2}}$ times the magnitude of the peak, Figure 7.7. ω_a and ω_b are respectively the frequencies of the left and right intersections with the half-power line.

For closely space peaks it might not be possible to find the intersections with the half-power line, therefore it is also possible to select the left and right intersections from the phase diagram of the averaged frequency response function. The left intersection is then equal to the natural frequency minus $45°$ phase change. The right intersection is equal to the natural frequency plus $45°$ phase change. This results in the same left and right frequencies as was derived from the half-power line, however these are not influenced by the width of the peak.

The damping coefficients and damping matrix can be calculated using the following formula's:

$$\xi = \frac{\omega_b - \omega_a}{2\,\omega_n}, \qquad \boldsymbol{C} = \begin{bmatrix} \omega_1 * \xi_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \omega_m * \xi_m \end{bmatrix}$$

And again, unidentifiable resonance peaks can be left out of the calculations:

$$\boldsymbol{C} = \begin{bmatrix} \omega_1 * \xi_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad (V_2 \; and \; V_3 \; not \; visible)$$

Limitations

The quadrature picking method is a very comprehensible way of finding the natural frequencies and modes of a system, however there are some limitations [15]:

- It is assumed that only a single mode contributes to the response of the system at a natural frequency. This is generally not the case, even when modes a largely separated. Each peak in the frequency response function is therefore a combination of modes. This may result in errors in the derived system properties.
- Because the method is based on finding the magnitude of a resonance peak, errors may occur when the measurements of such peaks are inaccurate, which is often the case.
- The derived system properties can only contain a real part. It is not possible to extract the imaginary part of the system properties by means of quadrature picking.

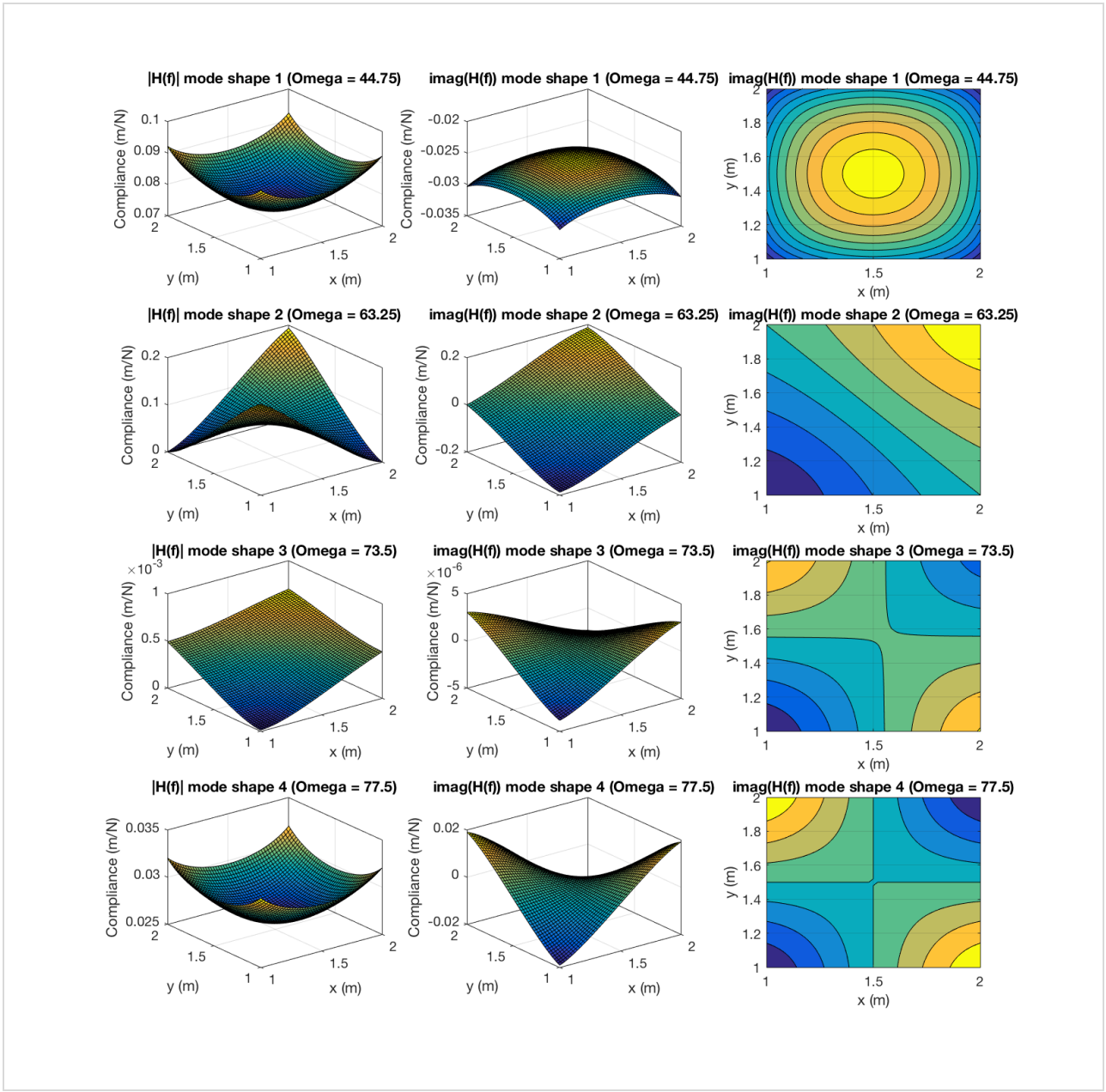


Figure 7.8: Mode shapes of the four-body model.

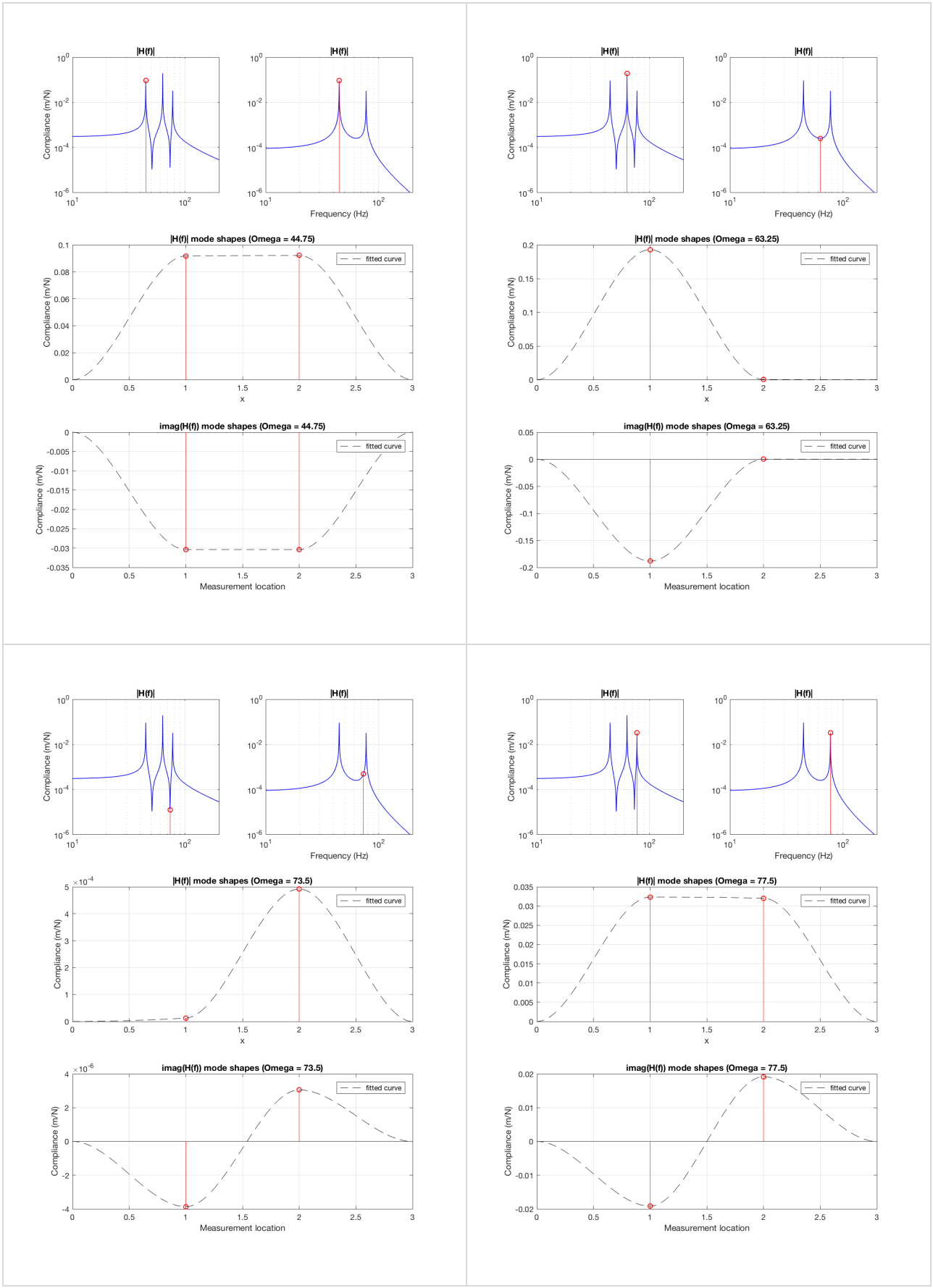


Figure 7.9: Quadrature picking the natural modes of the first two bodies of the four-body model.

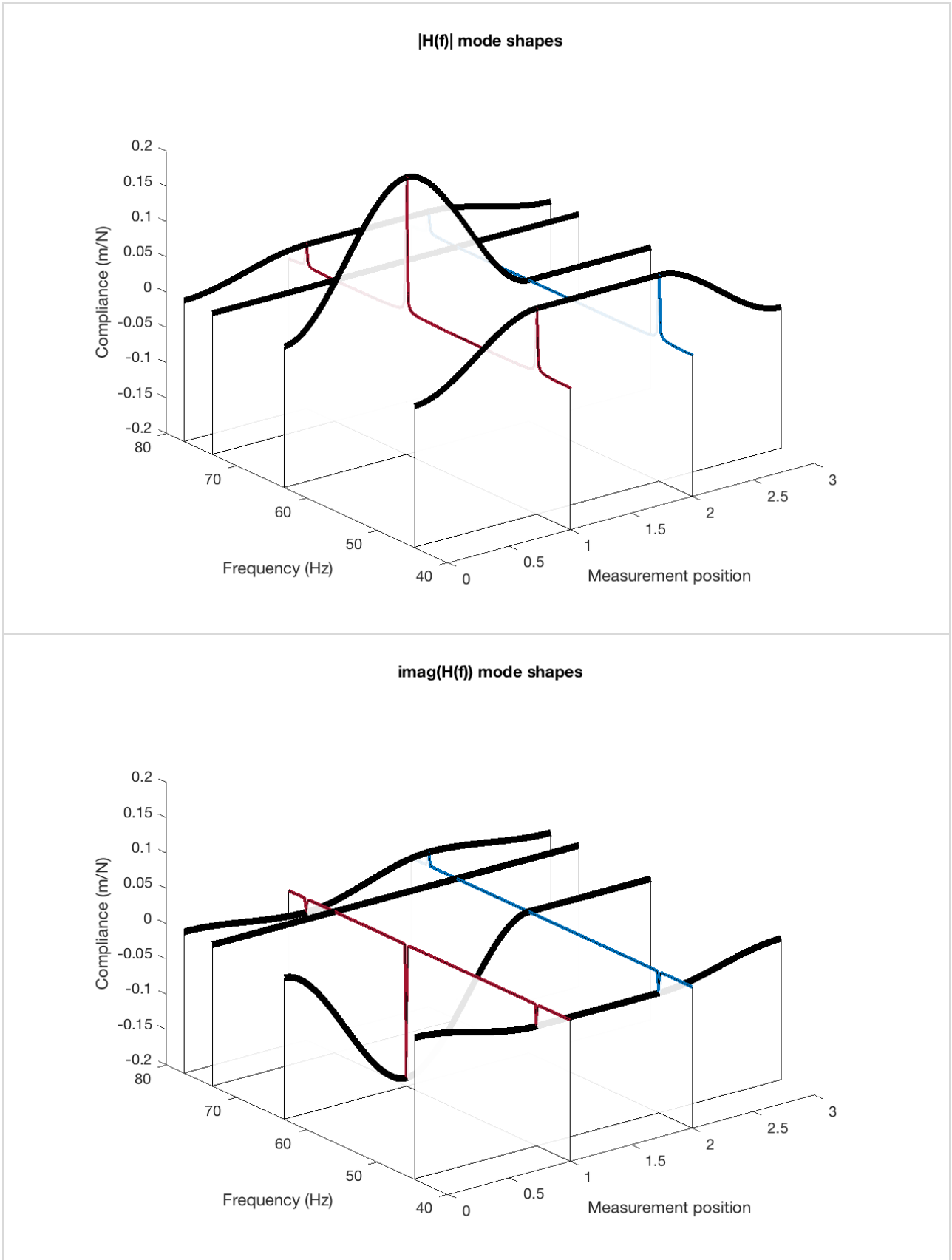


Figure 7.10: Waterfall plot showing the contribution of each frequency response function to the natural modes of the first two bodies of the four-body model.

Chapter 8 Modal parameter estimation

Modal parameter estimation is probably the most difficult part of the whole modal analysis. In Chapter 5, it was shown that the frequency response functions derived from the modal description of the system can result in a perfect description of the system. However, this is generally not the case in a real-life situation as the measurement data always contains noise and parasitic modes, where the structure vibrations in a different direction than the direction measured. Also, the quadrature picking method only finds the general shape of the modal matrix, Table 8.1. This means that the regenerated modal frequency response functions are not a perfect fit for the measured data, as shown in Figure 8.1. When only measuring the stiffness of the system or the general shape of the modes, it is perfectly fine to use the quadrature picking method without scaling to derive these characteristics [2]. This was also shown by Hugo Nauta [17]. If the complete system is to be derived from the measurement, a scaling must be applied however.

The last step in the modal analysis is therefore finding a model which describes the measured system accurately enough. The most commonly used method to derive the scaling of the system is by means of curve fitting [15], which is the most simplistic and clarifying way. Curve fitting is an optimisation technique, which minimises the mean square error between the measurement data and a fit function [18]. Simply saying, it's like drawing a line through the measurement data.
The fit function is determined by the measured system. This function was derived by the quadrature picking method, in the form of the modal description of the system. The frequency response functions matrix derived from the modal description is a simplified model of the measurement data.

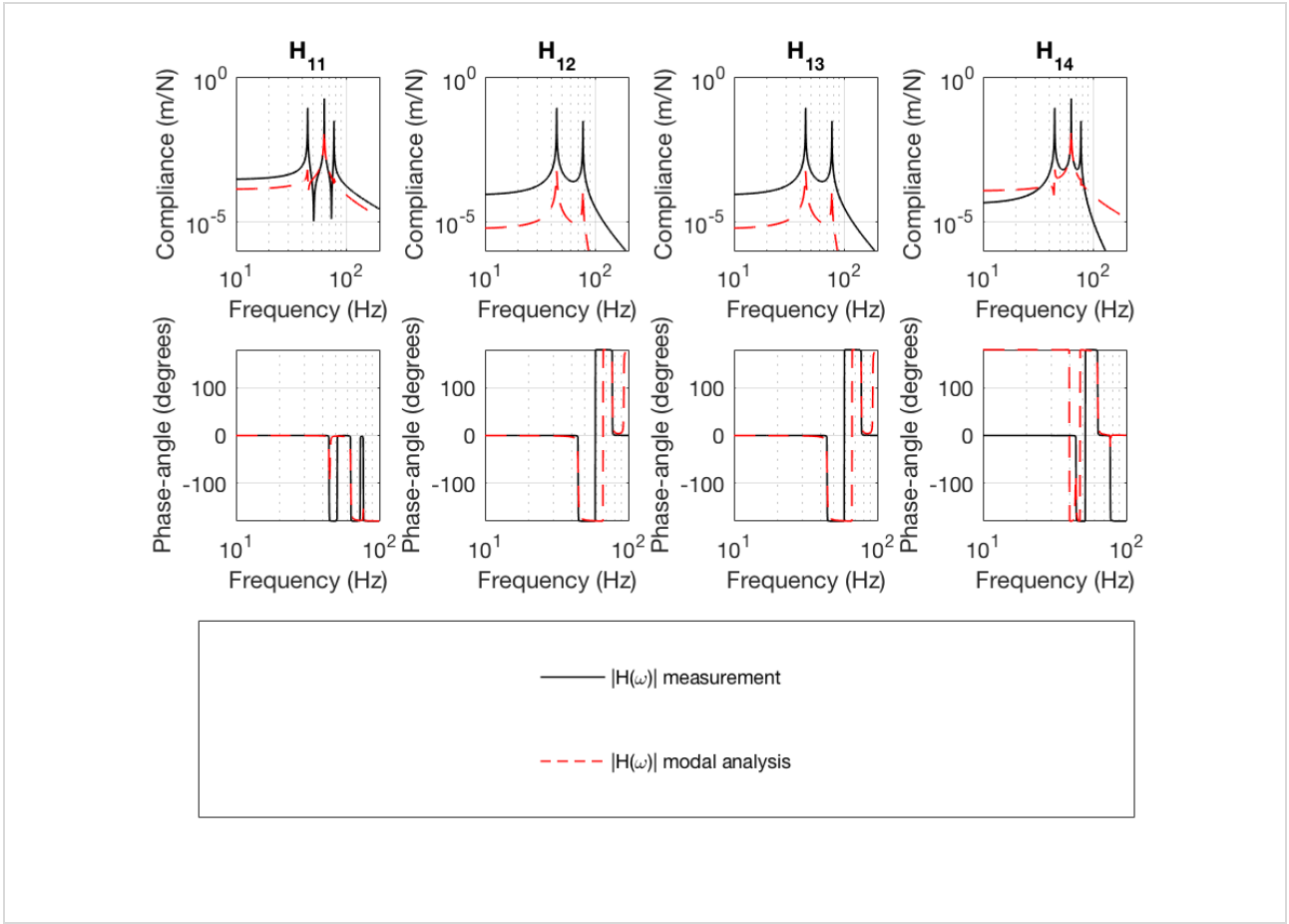


Figure 8.1: Regenerated frequency response functions from the modal description for the first row of the four-body model.

Three methods will be discussed in this chapter: basic model curve fitting, mean square optimisation curve fitting and single mode contribution scaling. Because these methods all make use of optimisation algorithms, the accuracy of the methods also greatly depends on the accuracy of these optimisation algorithms. It is therefore necessary to analyse the accuracy of the used algorithm, before it can be applied during the modal analysis.

The four-body model of Figure 7.1 will again be used to illustrate the modal parameter estimation.

	Equations of motion				***Quadrature picking***			
V	*-0.5000*	*0.5896*	*-0.3904*	*0.5000*	*-0.1140*	*-0.7071*	*0*	*-0.0719*
	-0.5000	*-0.3904*	*-0.5896*	*-0.5000*	*-0.1140*	*0*	*0*	*0.0719*
	-0.5000	*0.3904*	*0.5896*	*-0.5000*	*-0.1140*	*0*	*0*	*0.0719*
	-0.5000	*-0.5896*	*0.3904*	*0.5000*	*-0.1140*	*0.7071*	*0*	*-0.0719*

Table 8.1: Difference in modal matrices derived from the equations of motion and by means of quadrature picking.

8.1 Understanding the shape of the modal matrix

Before the modal matrix can be scaled, the shape of the matrix must be understood. This knowledge can later be used to pre-scale the modal matrix, before curve fitting is applied. This increases the accuracy of the curve fitting technique and lowers the error between the measured frequency response functions and the functions derived from the modal description.

Scaling of the plus/minus signs

The first step in the scaling of the modal matrix is to make sure the plus/minus signs for each individual coefficient of the modes are correct. If this is not the case, anti-resonance peaks might show up at the wrong frequencies in the frequency response function plots (4.3.1).

The scaling can be done by looking at the standard mode shapes of the structure, in this case a clamped plate. The first four mode shapes are shown in Figure 8.2. The theoretical modal matrix has the following shape:

$$\boldsymbol{V} = \begin{bmatrix} - & 0 & + & + \\ - & - & 0 & - \\ - & + & 0 & - \\ - & 0 & - & + \end{bmatrix}$$

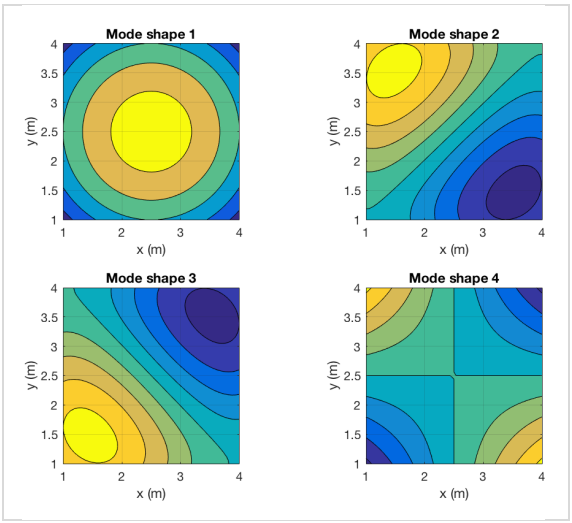


Figure 8.2: Mode shapes of a standard surface model.

Comparison of the columns with the corresponding mode shape shows a striking similarity:

- Mode 1: all coefficients in the same direction.
- Mode 2 and mode 3: two coefficients zero, two coefficients in opposite direction.
- Mode 4: two coefficients negative, two coefficients positive.

This means that the modal matrix can be scaled by looking at the standard mode shapes of the structure. However, there are some errors: mode 2 and mode 3 are swapped and mode 1 is negative instead of positive. This means that this scaling method can still result in the wrong modal matrix. This will however become evident when looking at the anti-resonance peaks of the frequency response function. If anti-resonance peaks show up at frequencies where anti-resonance is not supposed to happen, the plus/minus-signs of entire modes can be swapped and the curve fitting can be retried.

Normalisation

A vector can be normalised to remove any data about its length. This leaves a vector with a certain shape and length equal to one ($norm = 1$) [19]. Removing the length data makes it easier to scale the shape of the modal matrix. Normalisation of a vector is achieved by dividing the vector with its length:

$$\widehat{\boldsymbol{V}} = \frac{\boldsymbol{V}}{|\boldsymbol{V}|}$$

8.2 Basic model curve fitting

The basic model curve fitting method makes use of the definition of the standard modal frequency response function:

$$H_{ij}(\omega) = \sum_{k=1}^{N} \frac{A}{(-\omega^2 + {}_kC_m\, i\omega + {}_kK_m)}, \qquad where\ A\ is\ a\ scaling\ constant$$

Because any frequency response function can be written as a summation of single mode contributions, only the amount of single modes will have to be derived. Therefore, the natural frequencies of the system must be derived. After these frequencies are found, the standard frequency response function can be fitted to the measurement data, which is like drawing a line through the data. A curve fitting tool can then be used to fit the model, after which the mass, damping and stiffness of the system can be found.

Figure 8.3 shows the MATLAB interactive curve fitting tool. This tool can be called using the command:

```
cftool
```

The figure shows a fit for the first frequency response function in the matrix of the four-body model, where a three-term standard modal frequency response functions was fitted. This figure shows that the tool is able to find a good fit for the data. While this might seem like a decent fit, its usefulness is limited. The reason for this is that the found coefficients are only the damping and scaling of the system. The frequency response function in system parameters will then have to be regenerated from this modal description. This does however result in erroneous system matrices, which results in large errors when trying to make a prediction for the unmeasured frequency response functions.

Another attempt can be done by directly fitting the frequency response function in system parameter:

$$H_{ij}(s) = \sum_{l=1}^{N} \frac{A}{{}_lm\, s^2 + {}_lc\, s + {}_lk}, \qquad where\ A\ is\ a\ scaling\ constant$$

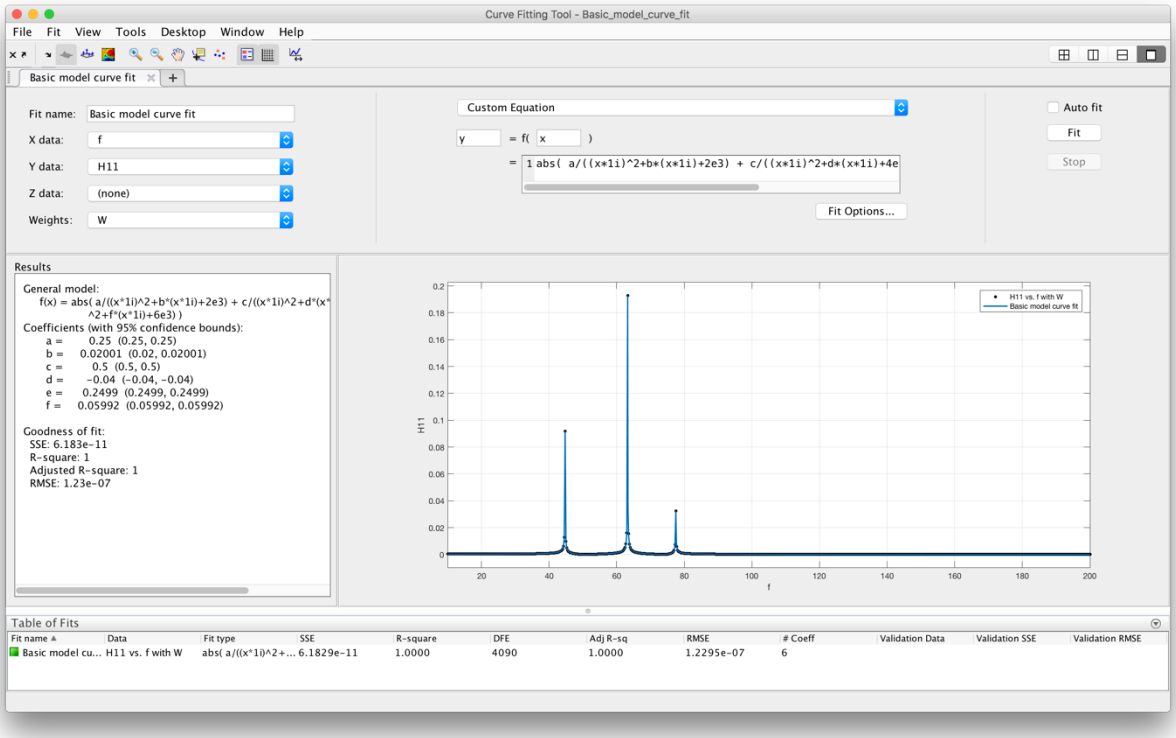


Figure 8.3: MATLAB curve fitting tool (cftool) – Modal frequency response function fit for the first FRF of the four-body model.

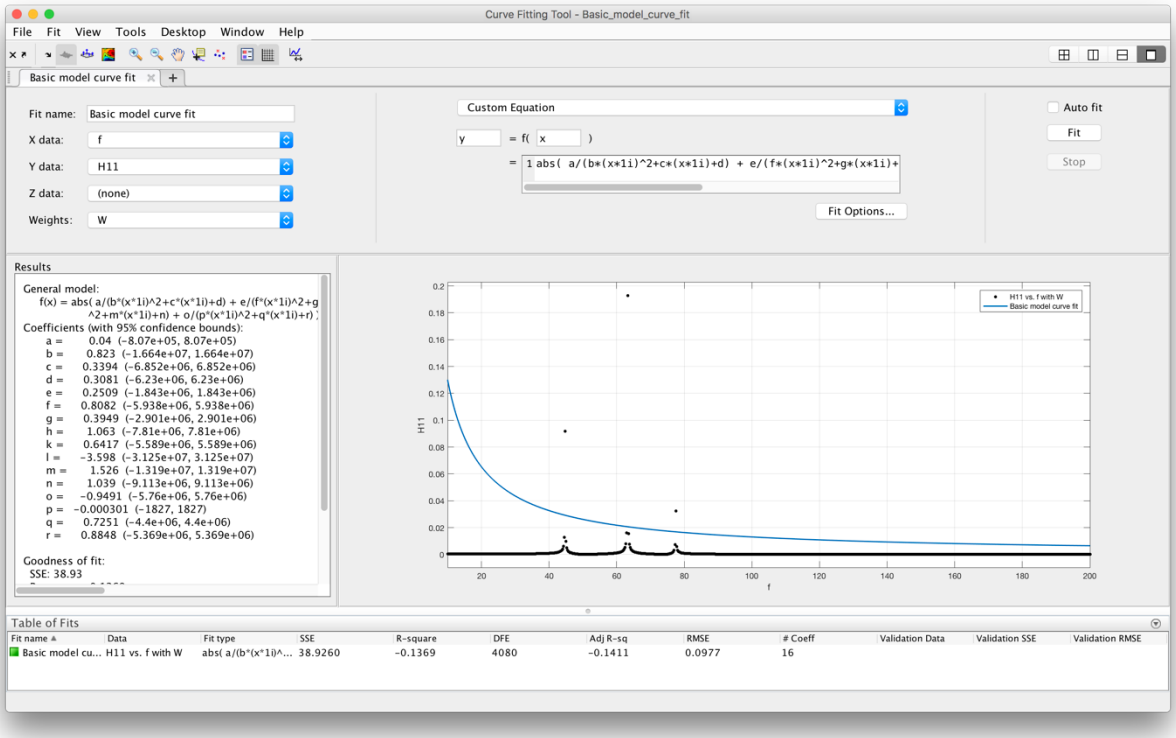


Figure 8.4: MATLAB curve fitting tool (cftool) – System parameter frequency response function fit for the first FRF of the four-body model.

Figure 8.4 shows this type of fit for the four-body. While the curve fitting tool does find a solution for this problem, it takes a long time to complete and does not find a reasonable fit. This is a problem that occurs often during the curve fitting of the measurement data. As the model becomes more complex, the chance of finding an accurate fit for the measurement data decreases rapidly.

8.3 Single mode contribution scaling

Because of the problems encountered during the basic model curve fit another technique will be attempted, where the single mode contributions of the frequency response functions are scaled. Each single mode contribution is scaled individually, after which they are summed to derive the total frequency response functions.

The scaling can be applied to the single mode contributions by multiplication with a scaling constant A:

$$h = \frac{A}{(-\omega^2 + {}_kC_m\, i\omega + {}_kK_m)}$$

The scaling constant can be derived from the difference between the resonance peak of the single mode contribution and the corresponding resonance peak of the measured frequency response function:

$$A = \frac{H_{EoM}(\omega_n)}{h_{nm}(\omega_n)}$$

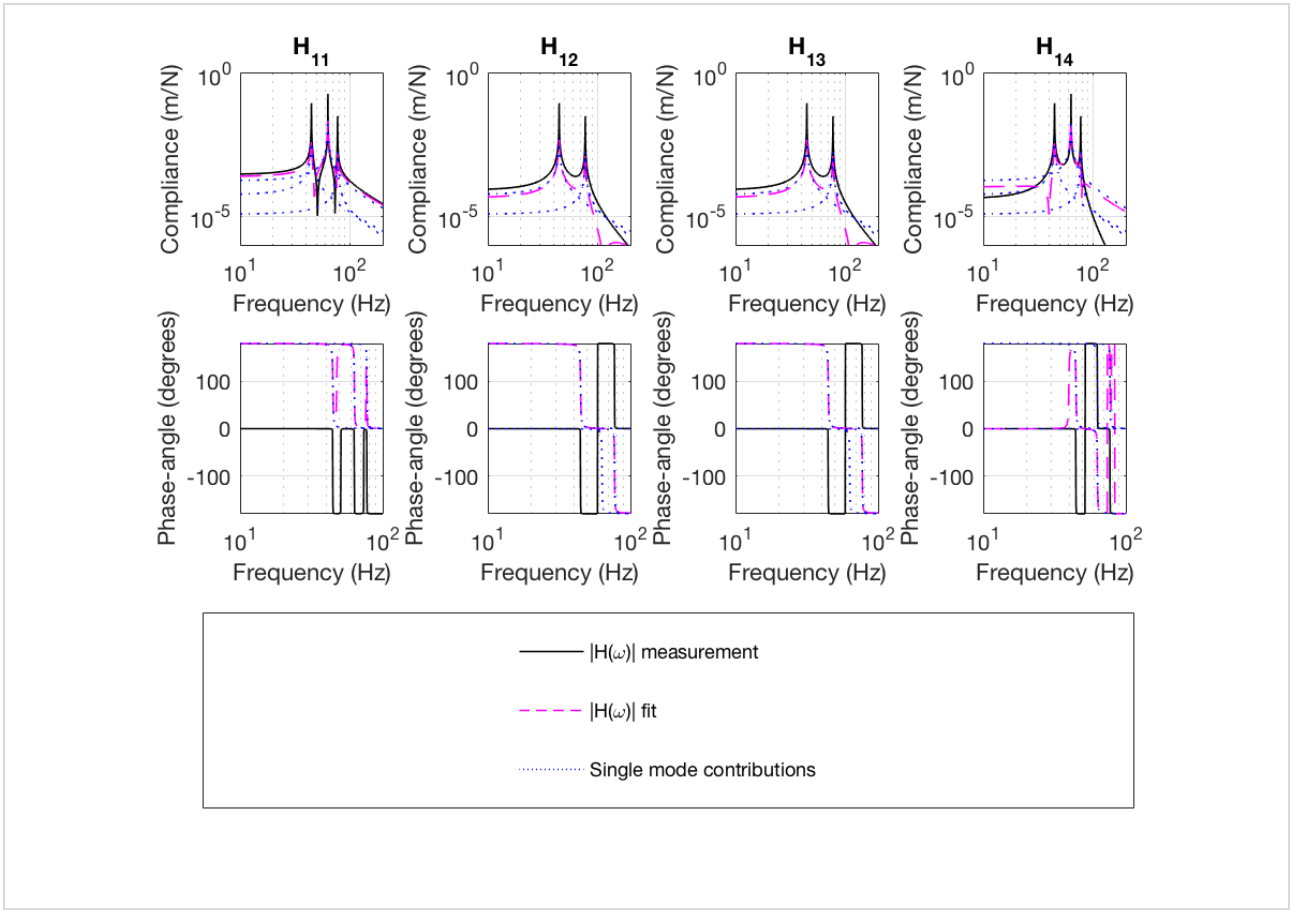


Figure 8.5: Regenerated frequency response functions from the scaled single mode contributions for the first row of the four-body model.

While in theory this method should work, it does not perform well when applied to the four-body model, Figure 8.5. It does result in a better fit than the frequency response functions derived by means of quadrature picking only, but still results in large errors.

8.4 Mean square error curve fitting

The last method is the full error minimisation using optimisation techniques. While this method results in a scaled modal matrix, which can be used to make prediction for the unmeasured frequency response functions, the accuracy of the method greatly depends on the accuracy of the optimisation technique. The accuracy of the optimisation technique in-turn depends on the initial value for the modal matrix and the bounds placed on the optimisation function. Because these bounds are unknown, they must be estimated or guessed, decreasing the accuracy of the fit.

The curve fitting optimisation technique minimises the mean square error between the measurement data and a fit function:

$$Minimise\ f(\boldsymbol{\Phi}) = \sum_{\omega}\sum_{j}\sum_{i} {E_{i,j,\omega}}^{2}\ , \qquad where\ E = H_{measurement} - H_{fit}$$

For the fit-function, the modal description of the system derived during the quadrature picking can be used. The objective function is a function of the modal matrix, this means that a value for the modal matrix must be found, such that the mean square error is minimal.

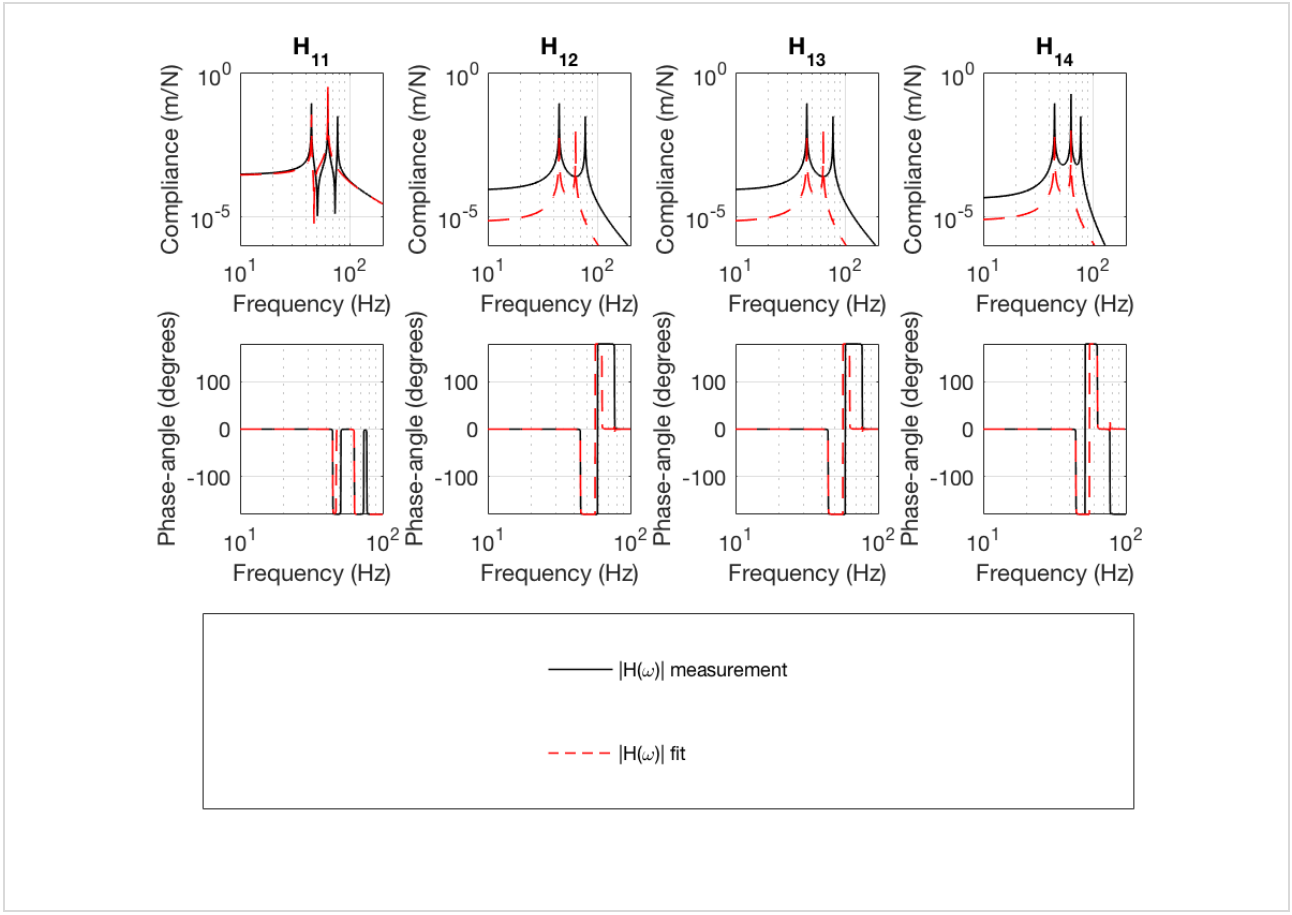


Figure 8.6: Regenerated frequency response functions from the mean square error curve fitting for the first row of the four-body model.

Figure 8.6 shows the mean square error curve fitting applied to the four-body model of the floor. This figure shows that the technique is no guarantee to find a perfect fit. The problem here is the missing resonance peak of the fourth mode. Another problem is the magnitude of the frequency response functions, which will be addressed in the next section. Nevertheless, this method is still preferred when trying to predict the unmeasured frequency response functions, as the mean square error curve fitting technique directly scales the modal matrix. This means that the unmeasured frequency response functions only have to be extracted from the corresponding index in the frequency response functions matrix (7.1).

The problem encountered here, where a resonance peak is completely missing, is not something that will be encountered regularly during normal measurements. The conditions for this to happen are:

- The same masses for all bodies.
- Uniform stiffness of the entire floor.

A slight difference in the stiffness or the masses of the bodies, results in all resonance peaks being visible and a far better fit using the mean square error curve fitting technique, as shown in Figure 8.7. Therefore, the mean square error curve fitting technique will work reasonably well in most cases.

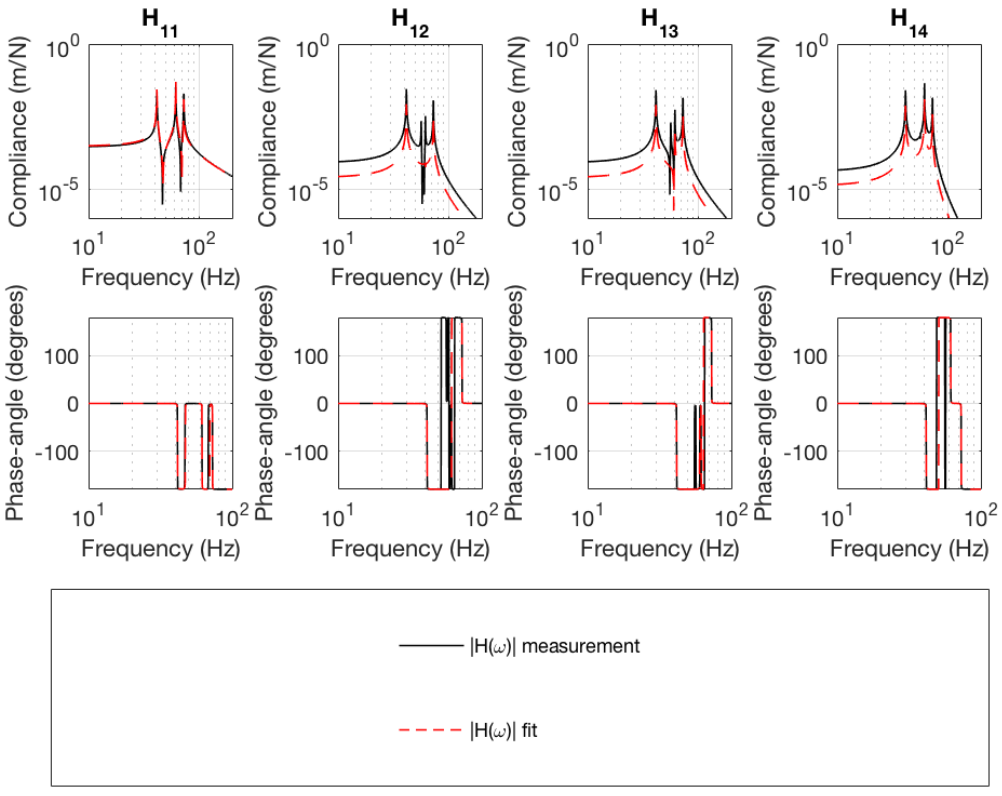


Figure 8.7: Regenerated frequency response functions from the mean square error curve fitting for the first row of the four-body model, with slightly different mass for each body.

8.4.1 Scaling of the magnitude

Because the optimisation technique will likely find a local minimum. This means that it is not able to find the correct magnitude of the modal matrix. This can easily be solved by calculating the difference between the low-frequency behaviour of both frequency response functions and then apply a scaling accordingly.
By doing this for the roving sensor/hammer measurements only and applying the scaling to the complete row or column of the frequency response functions matrix respectively, a descent prediction for all other measurements can be gained.

8.4.2 Residual effects

Residual effects are the resonance peaks that show up in the measurement data because of noise and parasitic modes [2]. These effects become most evident at high frequencies. To take these residual effects into account, they must be added to the derived frequency response functions. This can be done in the same way the single mode contributions are added to the frequency response functions:

$$H_{residue}(s) = \frac{1}{s^2 + c\,s + {\omega_{n,residue}}^2}$$

For each resonance peak as a result of a residual effect, the standard frequency response function can be added to the derived frequency response function.

8.4.3 MATLAB Optimisation

The mean square error optimisation can be performed by MATLAB. The minimisation problem must be formulated in the following way:

```
% Optimisation problem
Options = optimoptions(@fmincon, 'Algorithm', 'interior point');
ObjectiveFunction = @(V) OptimisationFunction(V,AdditionalParameters);
Problem = createOptimProblem('fmincon','x0',VModal,'objective',ObjectiveFunction,'options',Options);

% Minimise the mean square error
MS = MultiStart('Display','iter','UseParallel',true);
GS = GlobalSearch('Display','iter');
[V] = run(MS,Problem,1);        % Choose: Multistart or GlobalSearch
```

The accuracy depends on the fmincon- and interior point-algorithms. Several algorithms where used here to speed up and improve the optimization, which will be explained next.

Performance of fmincon

Figure 8.8 shows the fit of an exponential function:

$$y = a * x + b * x^2 + c * x^3 + d * x^4$$

The MATLAB function fmincon was used to estimate the coefficients of the exponential function. While the fit seems accurate, further comparison of the found coefficients with the original coefficients shows that the fit is actually not that accurate.

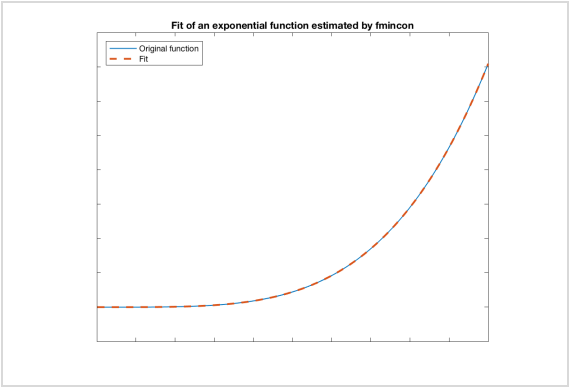


Figure 8.8: Simple Exponential function-fit estimated by fmincon.

	a	b	c	d
Original	89	35	102	709
Estimation	10	10	102	709

From this can be concluded that, while fmincon might find a good fit for the measurement data, the coefficients might not be accurate. Therefore, in case of a modal frequency response function, the estimated modal matrix might not be equal to the actual modal matrix of the system.

MultiStart and GlobalSearch

If an optimiser is run once only, it is for this type of problems almost certain that the result is a local optimum. This means that to find the global optimum of the function, multiple starting points must be tried. MATLAB has two algorithms to find the global optimum of a function: MultiStart and GlobalSearch. The main differences between GlobalSearch and MultiStart are:

- GlobalSearch uses a scatter-search mechanism for generating start points. MultiStart uses uniformly distributed start points within bounds, or user-supplied start points.
- GlobalSearch analyses start points and rejects those points that are unlikely to improve the best local minimum found so far. MultiStart runs all start points (or, optionally, all start points that are feasible with respect to bounds or inequality constraints).
- MultiStart gives a choice of local solver: fmincon, fminunc, lsqcurvefit, or lsqnonlin. The GlobalSearch algorithm uses fmincon.
- MultiStart can run in parallel, distributing start points to multiple processors for local solution.

MathWorks provides the following documentation for both functions [20]:

'MultiStart runs the local solver specified in the problem structure, starting at the points that pass the StartPointsToRun filter. If MultiStart is running in parallel, it sends start points to worker processors one at a time, and the worker processors run the local solver. When the local solver stops, MultiStart stores the results and continues to the next step.'

'GlobalSearch runs the optimiser fmincon from the start point defined in the problem structure. If the run converges, GlobalSearch records the start point and end point for the initial estimate on the radius of a basin of attraction. Furthermore, GlobalSearch records the final objective function value for use in the score function. The score function is the sum of the objective function value at a point and a multiple of the sum of the constraint violations. GlobalSearch updates the multiple during the run.'

Parallel computing

Optimisation problems can become extremely large and computational intensive. It is therefore advised to use the MATLAB Parallel Computing Toolbox. This toolbox is able to solve computationally and data intensive problems using multicore processors, GPU's and computer clusters. It is however not available for all optimisers. The parallel computing toolbox can be called via the optimisation options:

```
Options = optimoptions(@Solver, 'UseParallel', true);
```

8.5 Other methods

Besides curve fitting, many other methods are available for the experimental modal analysis of a system. Most of these methods are far more complicated and only available in advanced FEM-software packages.

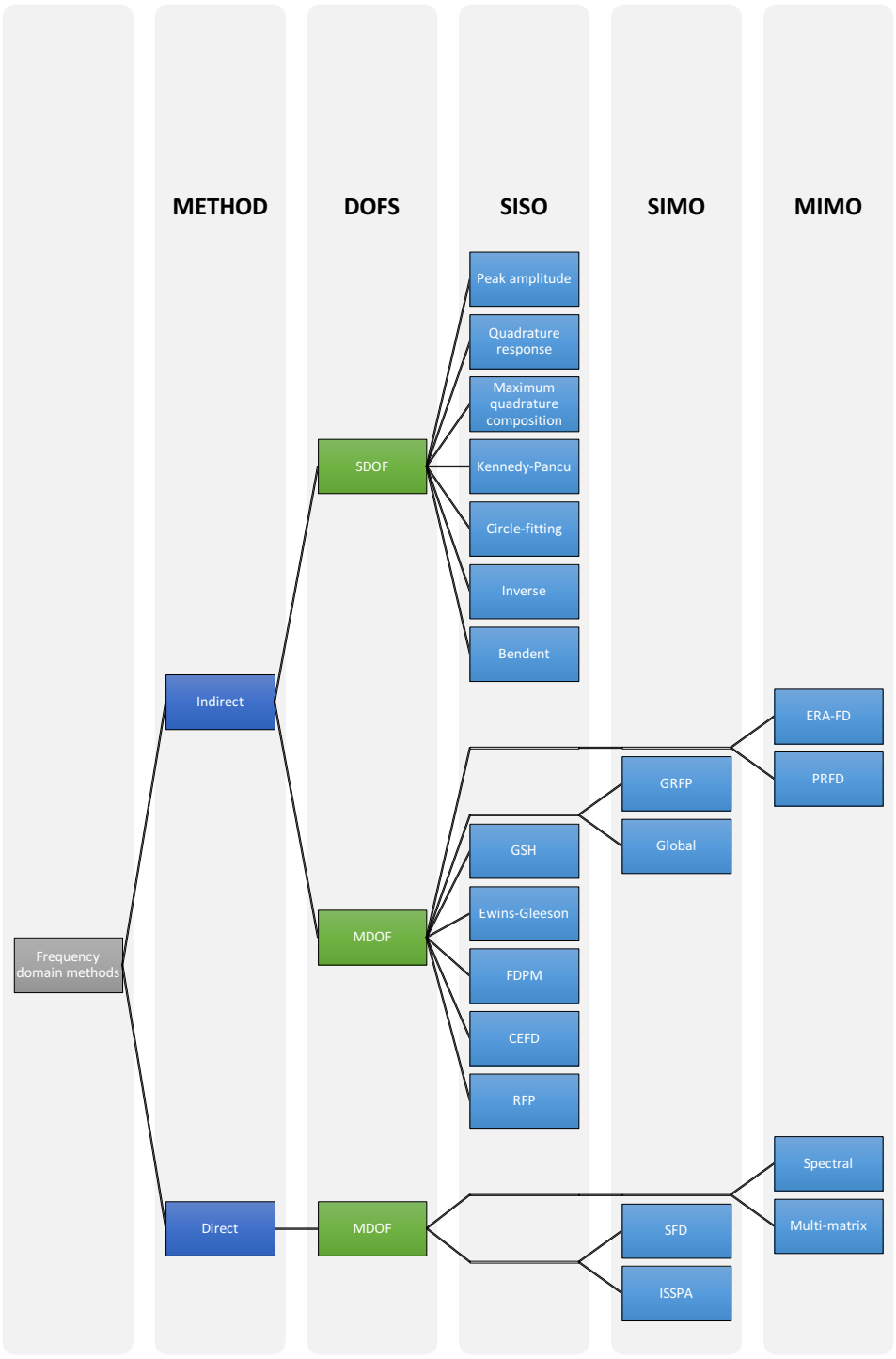


Figure 8.9: Other methods available for the experimental modal analysis of a system [21].

Chapter 9 Error of the modal analysis

The modal parameter estimation techniques discussed in Chapter 8 all showed that it becomes increasingly harder to find a fit for the measurement data, as the complexity of the problem increases. An analysis of the error made can give a clear view on which method works best in a certain situation. The error analysis is done by looking at increasingly difficult problems, starting with the very basic lightly-damped single-body model of Figure 9.1.

Lightly-damped single-body model

Figure 9.1 shows the mean square error curve fit for the lightly-damped single-body model. With a total error of $\mathrm{E_{total}} = 0.0690$ and a mean error of $\mathrm{E_{mean}} = 1.680 \cdot 10^{-5}$, the curve fit is as expected very accurate. A magnitude scaling was not necessary in this case, as the MATLAB optimisation algorithm already found the global minimum of the mean square error.

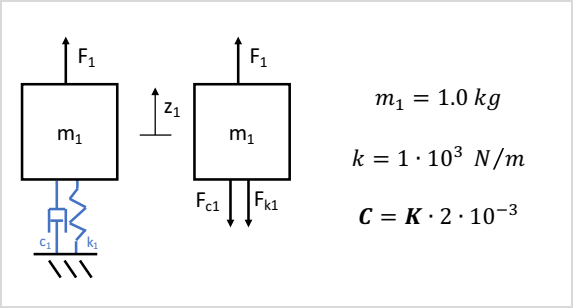


Figure 9.1: lightly-damped single-body model, problem definition.

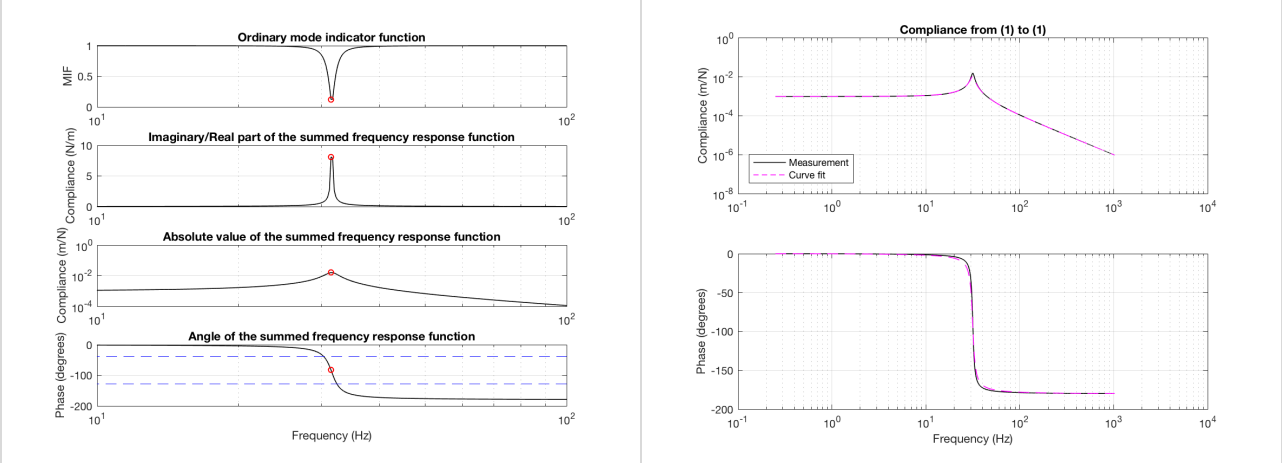


Figure 9.2: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the lightly-damped single-body model.

Extreme-damped single-body model

Next, the extreme heavily-damped single-body model of Figure 9.3 will be analysed. The difference in accuracy between these two single-body models will show the influence of the damping on the accuracy of the curve fitting method.

The result of the curve fit is shown in Figure 9.4. In this case, the ordinary mode indicator function still shows the frequency of the resonance peak.

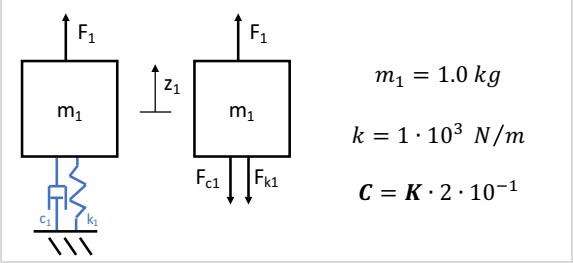


Figure 9.3: Extreme-damped single-body model, problem definition.

It is important that this natural frequency is found, as the curve fitting method would fail otherwise. The total error of the curve fit is $\mathrm{E_{total}} = 0.0137$ and the mean error $\mathrm{E_{mean}} = 3.349 \cdot 10^{-6}$. This shows, for systems with a single resonance peak, that the mean square error curve fitting method provides accurate results, even for systems with very high damping. However, the natural frequency of the system must be found.

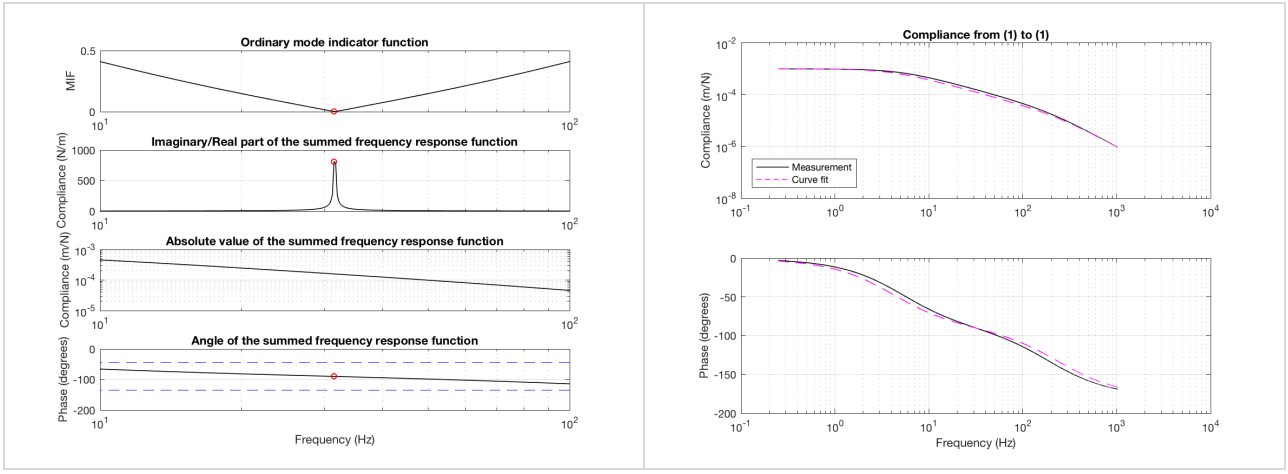


Figure 9.4: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the extreme-damped single-body model.

Lightly-damped two-body model

Next, the two-body models will be analysed. These models will give insight in how the modes influence each-other and what effect the increase in complexity of the system has on the accuracy of the curve fit.
Because the frequency response functions matrix has dimensions 2X2, it is also possible to analyse the predictions made by the method.

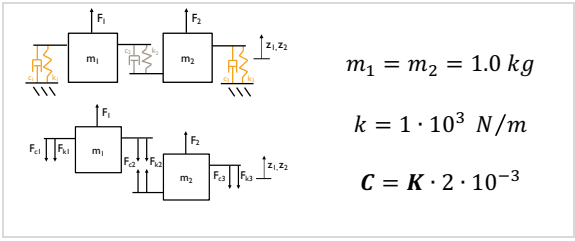


Figure 9.5: lightly-damped two-body model, problem definition.

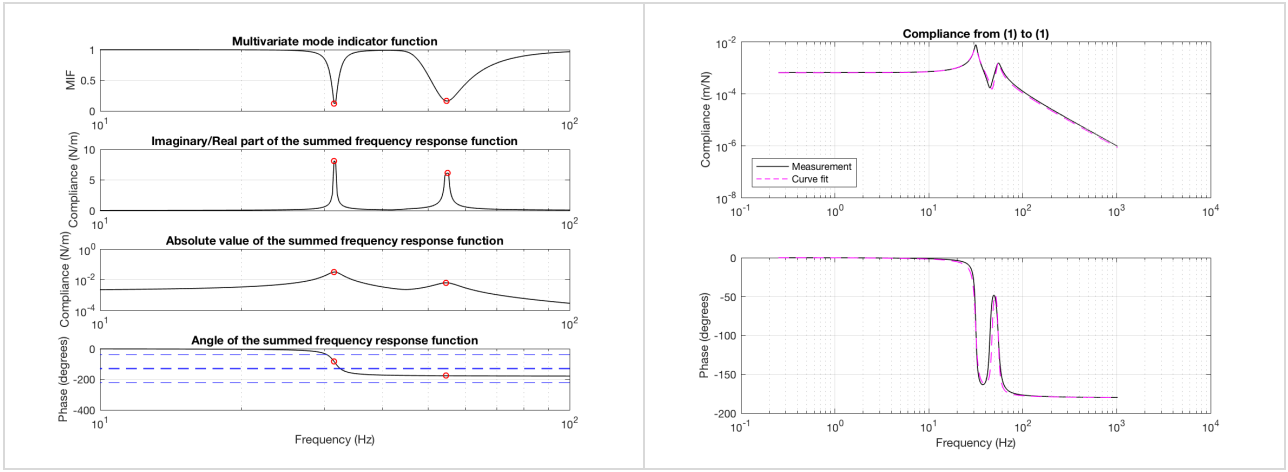


Figure 9.6: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the lightly-damped two-body model.

The total error and mean error for this case are $\mathrm{E_{total}} = 0.0135$ and $\mathrm{E_{mean}} = 8.219 \cdot 10^{-7}$, respectively. This shows that, for the lightly damped case and with decent spacing of the resonance peaks, the curve fit again is very accurate.

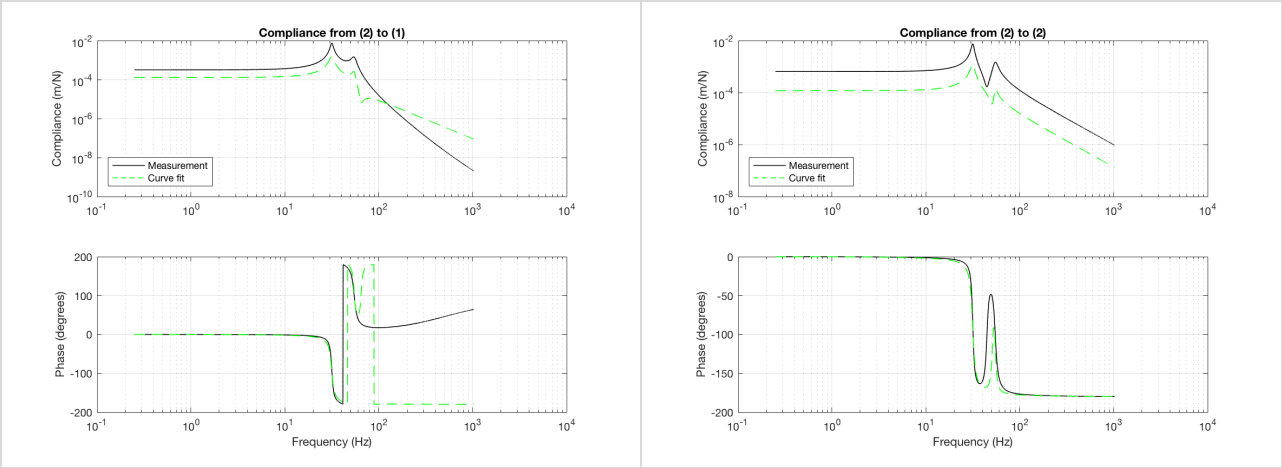


Figure 9.7: Frequency response functions and the corresponding predictions for the lightly-damped two-body model.

Figure 9.7 shows the prediction for an output from the second body. The output from the first body was used here to make these predictions. As can be seen from the figure, the curve fit is already far less accurate, having a total and mean error of $\mathrm{E_{total}} = 0.246$ and $\mathrm{E_{mean}} = 1.504 \cdot 10^{-5}$ for the frequency response function with the first body as input (left figure) and $\mathrm{E_{total}} = 0.334$ and $\mathrm{E_{mean}} = 2.041 \cdot 10^{-5}$ for the frequency response function with the second body as input (right figure). The left figure also shows an additional anti-resonance peak. With some magnitude scaling, the right frequency response function can become a decent fit however.

From this can be concluded that, for a lightly-damped multi-body system where the natural frequencies are visible, the mean square error curve fitting results in a good curve fit for the measurement data.
The predictions made by the method are not that accurate however and must only be used as an indication for the unmeasured frequency response functions.

Heavily-damped two-body model

The heavy-damped two-body model is shown in Figure 9.8. The curve fit has a total error of $\mathrm{E_{total}} = 0.0434$ and a mean error of $\mathrm{E_{mean}} = 2.647 \cdot 10^{-6}$. Surprisingly, the while the error for the curve fit is larger than the lightly-damped model, the error of the prediction is lower: $\mathrm{E_{total}} = 0.211$ and $\mathrm{E_{mean}} = 1.287 \cdot 10^{-5}$. The phase-plot however, like the lightly-damped case, is still off.

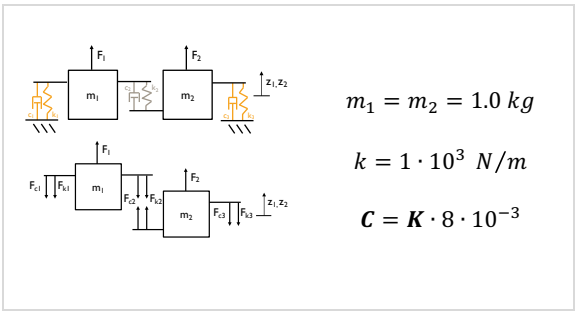


Figure 9.8: Heavily-damped two-body model, problem definition.

From this case can be concluded, that the system with intermediate damping and clearly visible resonance peaks is best when trying to predict the unmeasured frequency response functions, by means of mean square error curve fitting.

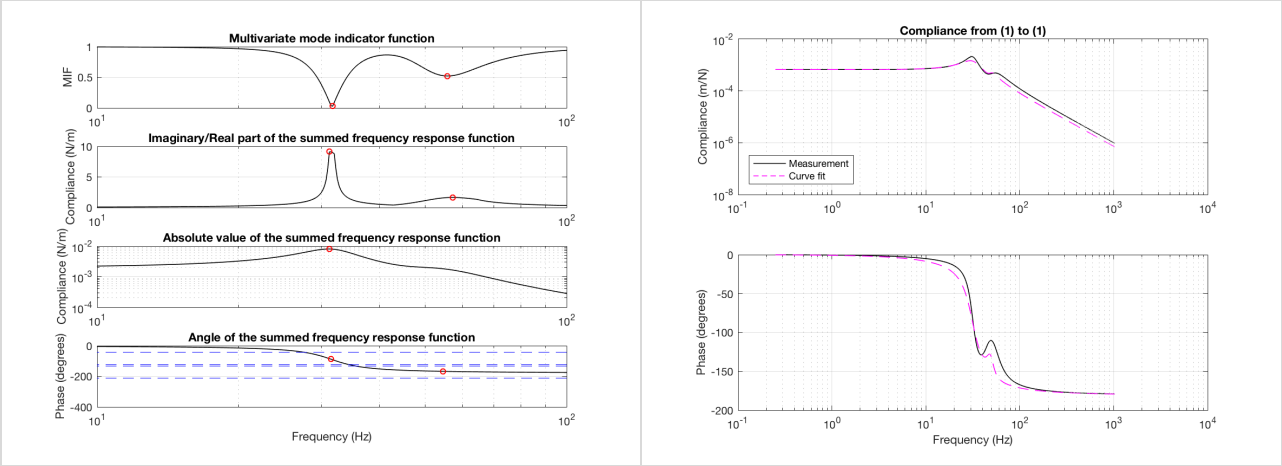


Figure 9.9: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the heavily-damped two-body model.

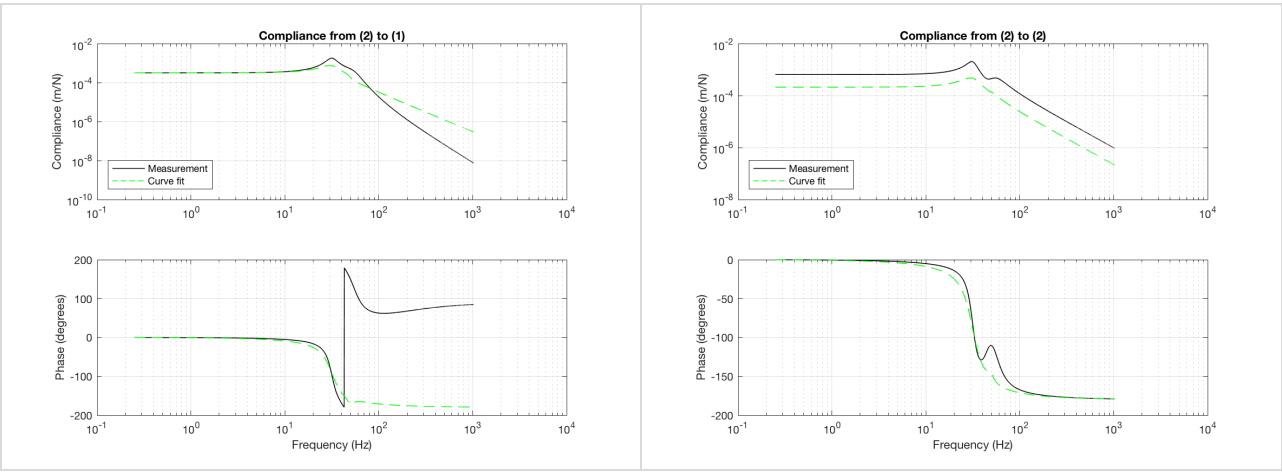


Figure 9.10: Frequency response functions and the corresponding predictions for the heavily-damped two-body model.

Lightly-damped four-body model

To investigate to what extent the curve fitting is viable for measured data, the four-body model will again be analysed, Figure 9.11. The damping coefficient was chosen at the critical point. This means that a slightly higher damping coefficient will result in erroneous curve fits, as will be shown next.

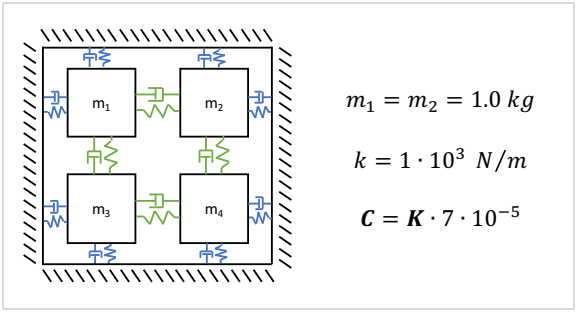


Figure 9.11: Lightly-damped four-body model, problem definition.

The total error of the curve fit is $\mathrm{E_{total}} = 0.2750$ and the mean error $\mathrm{E_{mean}} = 4.1964 \cdot 10^{-6}$.

As discussed in chapter 8.4, because of the missing resonance peak, even for the lightly-damped model the curve fit is not very accurate. The predictions seem to show the same errors that were made for the curve fit of the measurement data and an error in the magnitude, Figure 9.13. The total error is $\mathrm{E_{total}} = 0.3596$ and the mean error $\mathrm{E_{mean}} = 5.4867 \cdot 10^{-6}$.

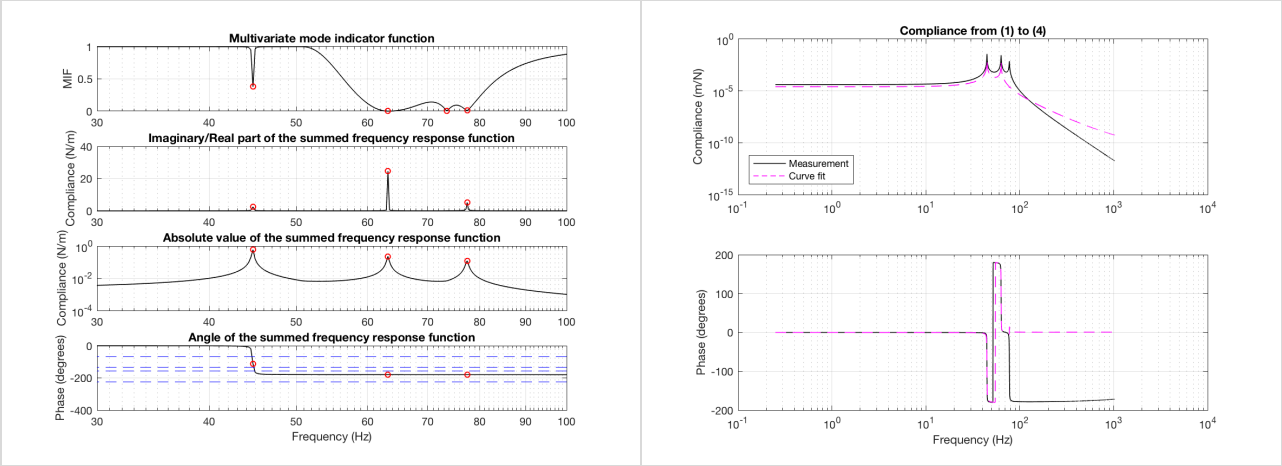


Figure 9.12: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the lightly-damped four-body model.

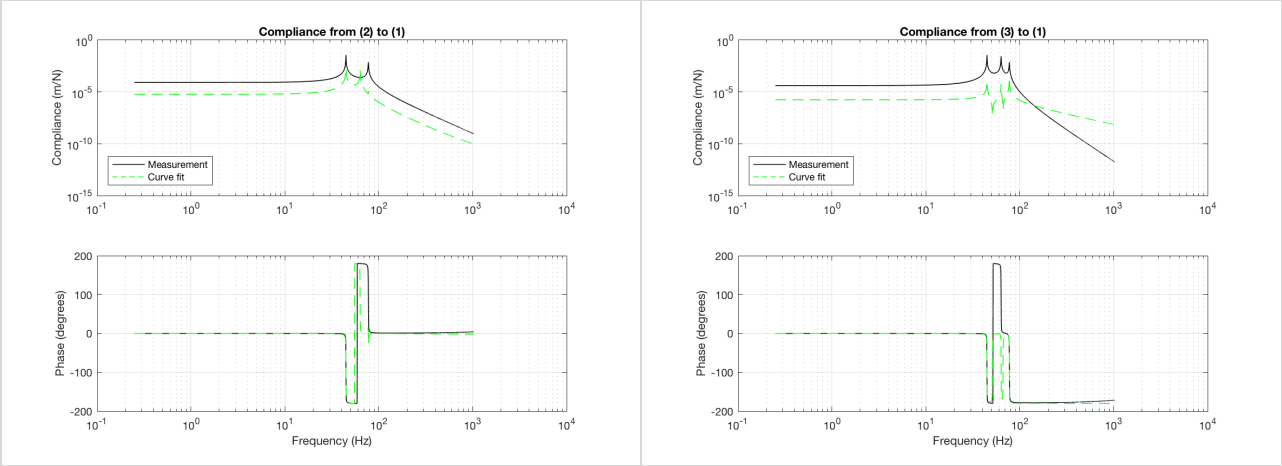


Figure 9.13: Frequency response functions and the corresponding predictions for the lightly-damped four-body model.

Slightly-more-damped four-body model

The last model showed the frequency response functions for the critical point. The model discussed here has a damping that is slightly higher, to show what happens to the curve fitting method when the damping is too high. The model is shown in Figure 9.14.

Figure 9.15 shows the curve fit for the model. This figure shows an extra resonance peak, which was not visible in the curve fit for the critical case.

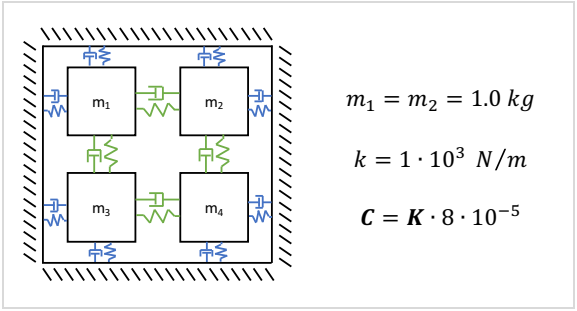


Figure 9.14: Slightly-more-damped four-body model, problem definition.

From this can be concluded that, in case the damping becomes too high for a certain system (strange resonance and anti-resonance peaks start to show up in the curve fit), it is better to apply a basic model curve fit. This does however mean that a prediction cannot be accurately made and more measurements must be carried out.

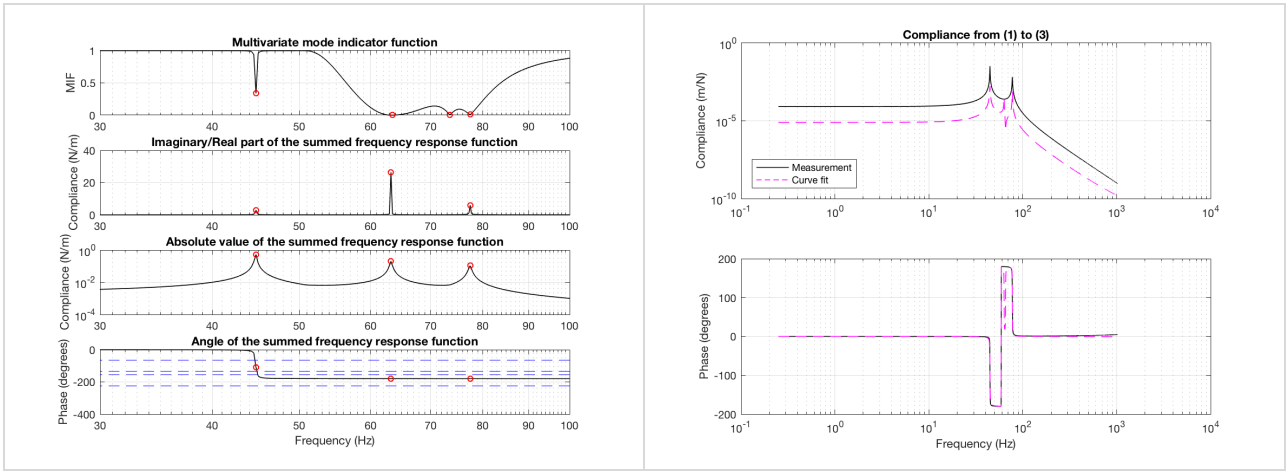


Figure 9.15: Identification of the natural frequencies (left), Frequency response function and the corresponding curve fit (right) for the slightly-more-damped four-body model.

Chapter 10 Discussion and recommendations

This research consisted of the derivation and experimental testing of several modal analysis techniques. The research concluded in these methods being able derive an estimation for the frequency response functions of the measurement data, however these estimations did not seem to be accurate for all situations and no general method was found. The accuracy of the estimation is highly situation-dependent. These conclusions resulted in two recommendations for further research:

- The mode indicator functions should be derived further, especially the complex mode indicator functions. This is a direct result from the loss in accuracy when resonance peaks are not visible in the measurement data and when resonance peaks are closely spaced together. These functions can then be used to identify these peaks.

- If the accuracy is extremely important for the expected results, more complicated methods should be explored for the derivation of the modal description of the structure. These methods can give a better initial estimation, which can then be used in combination with the mean square error curve fitting technique to provide results with a higher accuracy.

Chapter 11 References

[1] S. Bank, Active Vibration Solutions, Eindhoven: Precisiebeurs, 2017.

[2] P. Avitabile, Modal Space - Back to Basics, Experimental Techniques by the Society for Experimental Mechanics, 1998 - 2014.

[3] J. P. Schilder, Dynamics 2, Enschede: University of Twente, 2016.

[4] IDC Technologies Inc., Principles of Active Vibration Control: Basics of active vibration control methods, Milpitas, 2008.

[5] M. H. Richardson, Measurements and analysis of the dynamics of mechanical structures, Detroit: Hewlett-Packard Conference for Automotive and Related Industries, 1978.

[6] J. Lansink, Optimalisatie SpecTest, Enschede: MECAL Pantheon BV, 2015.

[7] National Instruments Corporation, Engineer's Guide to Accurate Sensor Measurements, Austin, 2016.

[8] National Instruments Corporation, Basic Information about AC and DC Coupling, Austin, 1999.

[9] S. Adhikari, Damping Models for Structural Vibration, Cambridge: Cambridge University Engineering Department, 2000.

[10] R. R. Craig Jr. and A. J. Kurdila, Fundamentals of Structural Dynamics, John Wiley & Sons Inc., 2006.

[11] R. Potter and M. Richardson, Mass, Stiffness and damping matrices from measured modal parameters, Santa Clara: ISA 74 International Instrumentation-Automation Conference & Exhibit, 1974.

[12] K. C. Park, Methods for Vibration Analysis, Boulder: University of Colorado, 1998.

[13] H. Schaub, ASEN 5022 - Dynamics of Aerospace Structures, Boulder: University of Colorado, 2005.

[14] T. Irvine, An Introduction to Frequency Response Functions, Madison, 2016.

[15] S. Ziaei-Rad, Modal Parameter Extraction Methods, Isfahan: Department of Mechanical Engineering, Isfahan University of Technology, 2004.

[16] M. Radeş, Performance of Various Mode Indicator Functions, Bucharest: University POLITEHNICA of Bucharest, 2010.

[17] H. Nauta, Means and Methods for Modal Building Analysis, Enschede: MECAL Pantheon BV, 2015.

[18] A. Yew, Curve fitting: least squares methods, Rhode Island: Brown University, 2011.

[19] E. W. Weisstein, Normalized vector, Oxfordshire: MathWorld, 2015.

[20] MathWorks, How GlobalSearch and MultiStart Work, Natick: MathWorks, 2017.

[21] U. E. Carlsson, Presentation: Mode parameter estimation - Circle-fit method, Stockholm: KTH Royal Institute of Technology, 2017.

[22] W. Tong, Z. Lingmi and T. K. Fah, Extraction of real modes and physical matrices from modal testing, Nanjing: Nanjing University of Aeronautics and Astronautics, 2011.

Appendix A MATLAB function EoM

The EoM.m function uses the masses, damping coefficients and spring constants to construct the system matrices.

```
function [K,C,M,F] = EoM(k,c,m,nBodies)
%% [K,C,M] = EoM(k,c,m,nBodies)
%
% Function to construct the system matrices.
%
% Inputs:
%   k = stiffness vector (k1, k2, etc.).
%   c = damping vector (c1, c2, etc.).
%   m = mass vector (m1, m2, etc.).
%   nBodies = the number of bodies (1, 2 or 3).
%
% Outputs:
%   K = stiffness matrix.
%   C = damping matrix.
%   M = mass matrix.
%   F = force matrix.

% Robert Giesen, Internship MECAL, october 2
%%
if (nBodies == 1)
    M = m;
    C = c;
    K = k;
    F = eye(size(M));
else
    M = diag(m(1:nBodies));
    C = diag(-c(2:nBodies),-1) + diag(c(1:nBodies)+c(2:nBodies+1)) + diag(-c(2:nBodies),1);
    K = diag(-k(2:nBodies),-1) + diag(k(1:nBodies)+k(2:nBodies+1)) + diag(-k(2:nBodies),1);
    F = eye(size(M));
end
```

Table 11.1: MATLAB function EoM.m.

Appendix B MATLAB function FRF

The FRF.m function uses the system matrices, modal matrix and eigenvalues to calculate the frequency response functions.

```
function [HEoM,HModal] = FRF(K,C,M,F,V,Kg,f,damping)
%% [HEoM,HModal] = FRF(K,C,M,F,V,Kg,f,damping)
%
% Function to calculate the frequency response functions matrix.
%
% Inputs:
%   K  = Stiffness matrix
%   C  = Damping matrix
%   M  = Mass matrix
%   V  = Undamped modal matrix
%   Kg = Undamped eigenvalues
%   f  = frequency range
%
% Outputs:
%   HEoM    = Frequency response functions matrix matrix calculated from the equations of motion
%   HModal  = Frequency repsonse functions matrix calculated from the modal description

% Robert Giesen, Internship MECAL, october 2
%% Initialisation
s = f * 1i;                                         % Laplace parameter
I = eye(size(M));                                   % Identity matrix

% Preallocate Memory
HEoM(1:size(V,1),1:size(V,2))   = {zeros(1,length(s))};
HModal(1:size(V,1),1:size(V,2)) = {zeros(1,length(s))};

%%
for k = 1 : length(s)
    if (strcmp(damping,'Undamped') == 1)
        % Undamped frequency response function from equations of motion
        D = M * s(k)^2 + K;                         % Dynamic matrix D
        Z = D\F;                                    % (-w^2 * M + K)*Z = F => Z = inv(-w^2 * M + K)*F

        % Undamped frequency response function from modal description
        DModal = I * s(k)^2 + Kg;                   % Modal dynamic matrix
        Eta    = DModal\(V' * F);                   % DModal * Eta = (V' * F) => Eta = DModal\(V' * F)
        Zr     = V * Eta;                           % Zr = recalculated z

    elseif (strcmp(damping,'Damped') == 1)
        % Damped frequency response function from equations of motion
        D = M * s(k)^2 + C * s(k) + K;              % Dynamic matrix D
        Z = D\F;                                    % (-w^2 * M + K)*Z = F => Z = inv(-w^2 * M + K)*F

        % Damped frequency response function from modal description
        DModal = I * s(k)^2 + C * s(k) + Kg;        % Modal dynamic matrix
        Eta = DModal\(V' * F);                      % DModal * Eta = (V' * F) => Eta = DModal\(V' * F)
        Zr = V * Eta;                               % Zr = recalculated z
    end

    % Store data in cell-array
    for i = 1 : size(Z,1)
        for j = 1 : size(Z,2)
            HEoM{i,j}(k)   = Z(i,j);     % Undamped frequency response function from equations of motion
            HModal{i,j}(k) = Zr(i,j);    % Undamped frequency response function from modal description
        end
    end
end
```

Table 11.2: MATLAB function FRF.m.

Appendix C MATLAB function ModeShapes

The ModeShapes.m function uses the modal matrix to define points, through which the mode shapes are fitted. The script makes use of the 'Shape-Preserving Piecewise Cubic Interpolation'-algorithm, which defines intervals between the points and fits the shape for each interval individually. This ensures that the mode shapes intersect the points defined by the modal matrix.

```
function [] = ModeShapes(U)
%% [] = Modeshapes(U)
%
% Function used to fit the modeshapes through the points defined by the
% modal vectors.
%
% Inputs:
%   U = modal matrix
%
% Outputs:
%   Returns a plot of the modeshapes

% Robert Giesen, Internship MECAL, october 2
%%
for i = 1 : size(U,2)
    % Fitting Mode shapes (Shape-preserving piecewise cubic Hermite (pchip) interpolation)
    ModeShape{i} = fit([-1:1:size([U(:,1)',0,0],2)]',[0;0;U(:,i);0;0],'pchipinterp');

    % Plotting figure
    figure(1)
    subplot(size(U,2),1,i)
    stem(real(U(:,i)),'color',rand(1,3)); hold on;
    xlim([0 size(U,2)+1])
    plot(ModeShape{i},'k--')

    legend(['Natural mode ' num2str(i)],'Mode shape')
    ylabel('u')
end
```

Table 11.3: MATLAB function ModeShapes.m.

Appendix D MATLAB function GatherData

The GatherData.m function reads the measurement files and stores the data in the alldata structure. It also constructs the frequency response functions matrix, according to the definition of section 7.1.

```
function [alldata,datamatrix] = GatherData(measurement,window)
%% [alldata] = GatherData(measurement)
%
% Function used to gather the data from the measurements.
%
% Inputs:
%   measurement = name of the measurement ('...')
%
% Outputs:
%   Returns a structure with all measurement data and a cell-array which
%   represents the measurement matrix.

% Robert Giesen, Internship MECAL, october 2
%% Read data
file_list = dir([measurement,'*.mat']);

for k = 1 : length(file_list)
    fl(k,:) = file_list(k).name;
end

fl = sortrows(fl,[2,3]);

%% Store data
for k = 1 : size(fl,1)
    data = load(fl(k,:)); alldata(k).meas = data.StiffGlob.meas;

    FixedSensorPosition     = str2num(fl(k,strfind(fl(k,:),'sf(')+3));
    ColocatedSensorPosition = str2num(fl(k,strfind(fl(k,:),'sc(')+3));
    HammerPosition          = str2num(fl(k,strfind(fl(k,:),'ha(')+3));

    % Store all data in structure
    alldata(k).FixedSensorPosition     = FixedSensorPosition;
    alldata(k).ColocatedSensorPosition = ColocatedSensorPosition;
    alldata(k).HammerPosition          = HammerPosition;

    % Calculate the sample rate
    dt= alldata(k).meas(1).t(2) - alldata(k).meas(1).t(1);

    for i = 1 : size(alldata(k).meas,2)
        Hamt(:,i) = alldata(k).meas(i).Hamt;
        zt(:,i)   = alldata(k).meas(i).zt;
        yt(:,i)   = alldata(k).meas(i).yt;
    end

    % FixedSensor (yt)
    [alldata(k).ytStiffness.f, alldata(k).ytStiffness.H,...
        alldata(k).ytStiffness.AbsH, alldata(k).ytStiffness.Phase,...
        alldata(k).ytStiffness.Coh]  = HAvgAndCoh(Hamt, yt, 1/dt, window);

    [alldata(k).ytCompliance.f, alldata(k).ytCompliance.H,...
        alldata(k).ytCompliance.AbsH, alldata(k).ytCompliance.Phase,...
        alldata(k).ytCompliance.Coh] = HAvgAndCoh(yt, Hamt, 1/dt, window);

    % Imaginary and real parts of the stiffness
    alldata(k).ytStiffness.ImagH = imag( alldata(k).ytStiffness.H );
    alldata(k).ytStiffness.RealH = real( alldata(k).ytStiffness.H );

    % Imaginary and real parts of the compliance
    alldata(k).ytCompliance.ImagH = imag( alldata(k).ytCompliance.H );
    alldata(k).ytCompliance.RealH = real( alldata(k).ytCompliance.H );

    % ColocatedSensor (zt)
    [alldata(k).ztStiffness.f, alldata(k).ztStiffness.H,...
        alldata(k).ztStiffness.AbsH, alldata(k).ztStiffness.Phase,...
        alldata(k).ztStiffness.Coh]  = HAvgAndCoh(Hamt, zt, 1/dt, window);

    [alldata(k).ztCompliance.f, alldata(k).ztCompliance.H,...
```

```
        alldata(k).ztCompliance.AbsH, alldata(k).ztCompliance.Phase,...
        alldata(k).ztCompliance.Coh] = HAvgAndCoh(zt, Hamt, 1/dt, window);

    % Imaginary and real parts of the stiffness
    alldata(k).ztStiffness.ImagH = imag( alldata(k).ztStiffness.H );
    alldata(k).ztStiffness.RealH = real( alldata(k).ztStiffness.H );

    % Imaginary and real parts of the compliance
    alldata(k).ztCompliance.ImagH = imag( alldata(k).ztCompliance.H );
    alldata(k).ztCompliance.RealH = real( alldata(k).ztCompliance.H );
end

%% Build the frequency response functions matrix
for k = 1 : size(fl,1)
    FixedSensor      = alldata(k);
    FixedSensor      = rmfield(FixedSensor,{'ztStiffness','ztCompliance'});
    FixedSensor.meas = rmfield(FixedSensor.meas,{'t','Senst','xt','zt','Hamf','Sensf','Dynf'});

    ColocatedSensor      = alldata(k);
    ColocatedSensor      = rmfield(ColocatedSensor,{'ytStiffness','ytCompliance'});
    ColocatedSensor.meas = rmfield(ColocatedSensor.meas,{'t','Senst','xt','yt','Hamf','Sensf','Dynf'});

    % Store in correct cell datamatrix
    datamatrix{FixedSensor.FixedSensorPosition,FixedSensor.HammerPosition}             = FixedSensor;
    datamatrix{ColocatedSensor.ColocatedSensorPosition,ColocatedSensor.HammerPosition} =
ColocatedSensor;
end
```

Table 11.4: MATLAB function GatherData.m.

Appendix E MATLAB function HAvgAndCoh

The HAvgAndCoh.m function uses the measurement data to calculate the frequency response functions (stiffness/compliance) and the coherence. This function file was written by Servaas Bank.

```
%% function [f H AbsH Phase Coh] = HAvgAndCoh(hammer_data, sensor_data, unit, fs, window)

% calculates the average frequency response function Y(f)/X(f) and coherence
%
% Inputs:
%   y_data(n,4) = array with data
%   x_data(n,4) = array with measured sensor data
%   fs = samplerate
%   window = FFT window (0 = none, 1 = hanning)
%
% Outputs:
%   f(n) = frequency vector (f> 0)
%   AvgH(n) = average compliance in [m/N] vector
%   Coh(n) = coherence vector
%
% sban, sept 2007
% aangepast voor spectest zodat m middelingen ipv 4 toegestaan zijn
% unit toegevoegd
%

% sban this one more general voor overdracht. march 2010

function [f H AbsH Phase Coh] = HAvgAndCoh(y_data, x_data, fs,window)

%% check some things

%check data range
[n m] = size(y_data);
% if ~(m ==4)
%       disp('fft_spectrum > data is not of format[n,4]');
% end

% also check on power of 2

%% calculate individual fft's & average
ft_y = zeros (n/2,m);
ft_x = zeros (n/2,m);
for i = 1:m
    [f ft_y(:,i)] = fft_cspectrum(y_data(:,i), fs, window);
    [f ft_x(:,i)] = fft_cspectrum(x_data(:,i), fs, window);
    %ft_sens(:,i) = ft_sens(:,i).*((2*pi*f).^unit);
end
f(1);
AVG = ( sum(ft_y,2)./ sum(ft_x,2))  ;
H = AVG;
AbsH = abs(AVG);
Phase = angle(AVG)*180/pi;

%% calculate coherence

part1 = sum(ft_x.*conj(ft_y) ,2);
part2 = sum( ft_y.*conj(ft_y) , 2);
part3 = sum( ft_y.*conj(ft_x) , 2);
part4 = sum(ft_x.*conj(ft_x) ,2);

nom = part1 .* part3;
denom = part2 .* part4;
Coh = nom ./ denom;

%% End of function
```

Table 11.5: MATLAB function HAvgAndCoh.m.

Appendix F MATLAB function fft_cspectrum

The fft_cspectrum.m function calculates the complex Fourier spectrum. This function file was written by Servaas Bank.

```
%% function [f cspect] = fft_cspectrum(data, fs, window)
% calculates the complex fourierspectrum of data for
% freq's > 0
% for amplitude spectrum see fft_spectrum.m
% sban sept 2007
% sban added correction for offset and linear drift

function [f cspect] = fft_cspectrum(data, fs,window)

%% check some things

%check data range
[n m] = size(data);
if ~(m ==1)
      disp('fft_spectrum > data is not of format [n,1]');
end

% also check on power of 2
b = log(n)/log(2);
if round(b) ~= b
    disp('fft_spectrum > N is not a power of 2');
end

%% correction for offset and linear drift only when window is applied
if window ==1
    x = 1:n;x = x';
    p = polyfit(x,data,1);  % fit to function y =p(1)*x+p(2)
    data = data-polyval(p,x);
end

%% make the freq vector
nyfreq = fs/2;
df = nyfreq /(n/2);
f = [df:df:nyfreq]';     % do not use freq = 0


%% apply window to the data
if window ==1                          % hanning window
     t = [0: 2*pi/(n-1):2*pi]';
    data = data .* (0.5-0.5*cos(t));
    % mark the following !
    % for corect amplitude's apply factor *2 on amplitudes
    % for correct power estimation (eg psd conversion)
    % apply *sqrt(8/3) on amplitude (or 8/3 on psd)
    % this is left to the user!
end

%% Now the fft
fft_raw = fft(data);
cspect = fft_raw(2:n/2+1)*2/n;

% % SBAN jan 2008
```

Table 11.6: MATLAB function fft_cspectrum.m.

Appendix G MATLAB function ModalFRF

The ModalFRF.m function calculates the frequency response functions from the modal description of the measured system.

```
function [alldata] = ModalFRF(alldata,ModesRange,PlotFigures)
%% [alldata] = ModalFRF(alldata,ModesRange,PlotFigures)
%
% Function to derive the frequency response functions from the modal matrix
%
% Inputs:
%   alldata     = structure with measurement data (use Gather_data m-function).
%   ModesRange  = Range of modes for which the modeshapes have to be determined.
%   PlotFigures = 1 if figures have to be plotted = 0 if no figures have to be plotted
%
% Outputs:
%   Returns the alldata structure with the calculated frequency response functions.
%

% Robert Giesen, Internship MECAL, october 2
%% Calculate the frequency response functions from the modal matrix
f(1,:) = alldata(1).ytCompliance.f;
fn     = alldata(1).ytOmega;
V      = alldata(1).V * 1i;

% Calculation of the damping coefficient Xi (changed during later calculations)
for i = 1 : length(ModesRange)
    for k = 1 : size(alldata,2)
        Omega(i,i) = alldata(1).ytOmega(ModesRange(i));
        Omega1    = Omega(i,i) - 1/2 * alldata(k).ytPeaksWidth(ModesRange(i));
        Omega2    = Omega(i,i) + 1/2 * alldata(k).ytPeaksWidth(ModesRange(i));

        % Take the mean value of the damping coefficient from all measurements
        Xi(k,i) = sum((Omega2 - Omega1)/(2 * Omega(i,i)),1)./6;
    end
end

f = alldata(1).ytCompliance.f;      % Frequency range
V = alldata(1).V;                   % Derived modal matrix
S = f * 1i;                         % Laplace parameter
I = eye(size(V));                   % Identity matrix
F = speye(size(V));                 % Force matrix

for n = 1 : size(alldata,2)
    for k = 1 : length(s)
        DModal = I * s(k)^2 + (2 * Xi * Omega) * s(k) + Omega.^2;       % Modal dynamic matrix
        Eta = DModal\(V' * F);          % DModal * Eta = (V' * F) => Eta = DModal\(V' * F)
        Zr = V * Eta;                   % Zr = recalculated z

        % Store data in cell-array
        HModal(k,:)  = Zr(2,:);         % For 2nd row of measurement matrix
    end
end

%% Plot figures?
if (PlotFigures == 1)
%% Plotting frequency response functions
range = 1 : 200;    % Frequency range

for k = 1 : size(alldata,2)
    f     = alldata(k).ytCompliance.f(range);
    AbsH  = alldata(k).ytCompliance.AbsH(range);
    Omega = alldata(1).ytOmega;
    Omegaplot(1,:) = Omega(ModesRange);
    Omegaplot(2,:) = Omega(ModesRange);

    figure(110)
    pl(k) = subplot(3,3,k);
    loglog(f,abs(HModal(range,k)),'r','LineWidth',0.8); hold on; grid on;
    loglog(f,AbsH,'b','LineWidth',0.8);
    plot(Omegaplot,[1e-20,1e0],'--k');
    xlabel('Frequency (Hz)'); ylabel('Compliance (m/N)');
```

```
    xlim([1 200]); linkaxes(pl,'x');
    title(['Measured and derived frequency response functions (point ',num2str(k),')'])

    % Create legend on the bottom row
    hSub = subplot(3,3,7:9); plot(1,1,'b',1,1,'r','LineWidth',0.8);
    hLegend = legend('Measured frequency response function','Modal frequency response function',...
                                                                    'Location','northwest');
    set(hLegend, 'position', get(hSub, 'position')); set(hSub,'Visible','off')
end
end
```

Table 11.7: MATLAB function ModalFRF.m.

Appendix H MATLAB function QuadraturePicking

The QuadraturePicking.m function applies the quadrature picking method and plots the mode shapes in 1D and the waterfall plot in 3D.

```
function [alldata] = QuadraturePicking(alldata,ModesRange,PlotFigures)
%% [alldata] = QuadraturePicking(alldata,ModesRange,PlotFigures)
%
% Function used to calculate the natural frequencies and mode shapes by means of the quadrature picking
% method.
%
% Inputs:
%   alldata     = structure with measurement data (use Gather_data m-function).
%   ModesRange  = Range of modes for which the modeshapes have to be determined.
%   PlotFigures = 1 if figures have to be plotted = 0 if no figures have to be plotted
%
% Outputs:
%   Returns the alldata structure with the calculated natural frequencies and mode shapes.
%

% Robert Giesen, Internship MECAL, october 2
%% Find the local maxima
for k = 1 : size(alldata,2)
    ImagH(k,:) = alldata(k).ytCompliance.ImagH;
    RealH(k,:) = alldata(k).ytCompliance.RealH;
    AbsH(k,:)  = alldata(k).ytCompliance.AbsH;
    Coh(k,:)   = alldata(k).ytCompliance.Coh;
    f(k,:)     = alldata(k).ytCompliance.f;
end

sumImagReal = (sum(abs(ImagH.*Coh),1)./sum(abs(RealH.*Coh),1));
sumAbs      = sum(AbsH,1);

MIF = sum(RealH.*AbsH,1) ./ sum(AbsH.^2,1);

[sumiPeaksMag, sumiPeaksLoc]     = findpeaks(sumImagReal,'MinPeakProminence',1);
[sumAbsPeaksMag, sumAbsPeaksLoc] = findpeaks(sumAbs,'MinPeakProminence',0.5e-7);

% Find peaks of the absolute frequency response functions
for k = 1 : size(alldata,2)
    [alldata(k).ytPeaksMag, alldata(k).ytPeaksLoc, alldata(k).ytPeaksWidth] = findpeaks(AbsH(k,:));
end

%% Natural frequencies
alldata(1).ytOmega = f(1,sumAbsPeaksLoc);

%% Natural modes
for i = 1 : length(ModesRange)
    for k = 1 : size(alldata,2)
        alldata(1).V(k,i)    = alldata(k).ytCompliance.ImagH(sumAbsPeaksLoc(ModesRange(i)));
        alldata(1).AbsV(k,i) = alldata(k).ytCompliance.AbsH(sumAbsPeaksLoc(ModesRange(i)));
    end
end

%% Fitting the mode shapes
for i = 1 : length(ModesRange)
    % Fitting Mode shapes (Shape-preserving piecewise cubic Hermite (pchip) interpolation)
    ModeShape{i}    = fit([-1:1:size([alldata(1).V(:,1)',0,0],2)]',[0;0;alldata(1).V(:,i);0;0],...
                                                                                    'pchipinterp');

    AbsModeShape{i} = fit([-1:1:size([alldata(1).AbsV(:,1)',0,0],2)]',[0;0;alldata(1).AbsV(:,i);0;0],...
                                                                                    'pchipinterp');

    % Evaluate the mode shape
    zAbsModeShape{i} = feval(AbsModeShape{i},[-1:0.01:size([alldata(1).AbsV(1,:),0,0],2)]);
end

%% Plot figures?
```

```
if (PlotFigures == 1)

%% Plotting mode indicator function
range = 1 : 200;    % Frequency range
figure(200)
semilogy(f(1,range),abs(MIF(range)),'k','Linewidth',0.8); hold on; grid on;
loglog(f(1,[13, 25, 35, 62, 70, 79, 107, 115, 139]),abs(MIF([13, 25, 35, 62, 70, 79, 107, 115,
139])),'or','MarkerSize',5)
xlabel('Frequency (Hz)'); ylabel('MIF')
xlim([1 50])
title('Ordinary mode indicator function')

%% Plotting local maxima
figure(1)
pl1 = subplot(2,1,1);
semilogx(f(1,range),sumImagReal(range),'k','Linewidth',0.8); hold on; grid on;
semilogx(f(1,sumiPeaksLoc),sumiPeaksMag,'or','MarkerSize',5)
xlabel('Frequency (Hz)'); ylabel('Compliance (m/N)')
xlim([1 50])
title('Imaginary/Real part of the summed frequency response function')

pl2 = subplot(2,1,2);
loglog(f(1,range),sumAbs(range),'k','Linewidth',0.8); hold on; grid on;
loglog(f(1,sumAbsPeaksLoc),sumAbsPeaksMag,'or','MarkerSize',5)
xlabel('Frequency (Hz)'); ylabel('Compliance (m/N)')
xlim([1 50]); linkaxes([pl1, pl2],'x')
title('Absolute value of the summed frequency response function')

%% Plotting natural modes (2D)
for i = 1 : length(ModesRange)
    for k = 1 : size(alldata,2)
        f        = alldata(k).ytCompliance.f(range);
        fn       = alldata(1).ytOmega(ModesRange(i));
        AbsH     = alldata(k).ytCompliance.AbsH(range);
        Modes    = alldata(1).V(k,i);
        AbsModes = alldata(1).AbsV(k,i);

        figure(2+i)
        subplot(3,size(alldata,2),k)
        loglog(f,AbsH,'b','LineWidth',0.8); hold on; grid on;
        stem(fn,AbsModes,'r');
        xlim([1 100]); ylim([1e-9 1e-5])
        if (k == 2) | (k == 5)
            xlabel('Frequency (Hz)')
        end
        if (k == 1)
            ylabel('Compliance (m/N)')
        end
        title('|H(f)|')

        subplot(3,size(alldata,2),size(alldata,2)+1:2*size(alldata,2))
        stem(k,AbsModes,'r'); hold on; grid on;
        xlim([0 size(alldata,2)+1])

        subplot(3,size(alldata,2),2*size(alldata,2)+1:3*size(alldata,2))
        stem(k,Modes,'r'); hold on; grid on;
        xlim([0 size(alldata,2)+1])
    end
    subplot(3,size(alldata,2),size(alldata,2)+1:2*size(alldata,2))
    plot(AbsModeShape{i},'--k')
    ylabel('Compliance (m/N)')
    title(['|H(f)| mode shapes (Omega = ',num2str(fn),')'])

    subplot(3,size(alldata,2),2*size(alldata,2)+1:3*size(alldata,2))
    plot(ModeShape{i},'--k')
    xlabel('Measurement location'); ylabel('Compliance (m/N)')
    title(['imag(H(f)) mode shapes (Omega = ',num2str(fn),')'])
end

%% Plotting natural modes (3D)
for i = 1 : length(ModesRange)
    for k = 1 : size(alldata,2)
        f                                = alldata(k).ytCompliance.f(sumAbsPeaksLoc(ModesRange(1))...
                                                                     -10:sumAbsPeaksLoc(ModesRange(end))+10);
        fn(1:length(ModesRange))         = alldata(1).ytOmega(ModesRange);
        AbsH                             = alldata(k).ytCompliance.AbsH(sumAbsPeaksLoc(ModesRange(1))...
                                                                     -10:sumAbsPeaksLoc(ModesRange(end))+10);
        AbsModes(:,1:length(ModesRange)) = alldata(1).AbsV(k,i);
```

```
        figure(100)
        plot3(k*ones(length(f)+2,1),[f(1);f;f(end)],[1e-8;AbsH;1e-8],'LineWidth',1); hold on; grid on;
        fill3(k*ones(length(f)*2,1),[f;flip(f)],[AbsH;1e-8*ones(length(AbsH),1)],'w','EdgeColor',...
                                                                                'none','FaceAlpha',0.9);
    end
    xModeShape = -1:0.01:size([alldata(1).V(1,:),0,0],2);
    figure(100)
    plot3(xModeShape,fn(i)*ones(size(xModeShape,2)),zAbsModeShape{i},'k','LineWidth',1);
    fill3(xModeShape,fn(i)*ones(size(xModeShape,2)),zAbsModeShape{i},'w','EdgeColor','none',...
                                                                        'FaceAlpha',0.9);
    xlim([0 7]); ylim([f(1),f(end)]);
    xlabel('Measurement position'); ylabel('Frequency (Hz)'); zlabel('Compliance (m/N)')
    title('|H(f)| mode shapes')
end
end
```

Table 11.8: MATLAB function QuadraturePicking.m.

Appendix I MATLAB function QuadraturePicking2D

The QuadraturePicking2D.m function applies the quadrature picking method on the measurements of a plane and plots the mode shapes in 2D.

```
function [alldata] = QuadraturePicking2D(alldata,x,y,PickPeaks,PlotFigures)
%% [alldata] = QuadraturePicking2D(alldata,x,y,PickPeaks,PlotFigures)
%
% Script used to calculate the natural frequencies and mode shapes by means of the quadrature picking
% method.
%
% Inputs:
%   alldata     = Structure with measurement data (use Gather_data m-function).
%   PickPeaks   = Vector with numbers of the natural frequency peaks.
%   PlotFigures = 1 if figures have to be plotted = 0 if no figures have to be plotted.
%
% Outputs:
%   Returns the alldata structure with the calculated natural frequencies and mode shapes.
%

% Robert Giesen, Internship MECAL, october
%% Preallocate Memory
ImagH = zeros(size(alldata,2),length(alldata(1).meas(1).f));
RealH = zeros(size(alldata,2),length(alldata(1).meas(1).f));
AbsH  = zeros(size(alldata,2),length(alldata(1).meas(1).f));
Coh   = zeros(size(alldata,2),length(alldata(1).meas(1).f));
f     = zeros(size(alldata,2),length(alldata(1).meas(1).f));

V     = zeros(size(alldata,2),length(PickPeaks));
AbsV  = zeros(size(alldata,2),length(PickPeaks));

X     = zeros(size(alldata,2),length(PickPeaks));
Y     = zeros(size(alldata,2),length(PickPeaks));
Z     = zeros(size(alldata,2),length(PickPeaks));
AbsZ  = zeros(size(alldata,2),length(PickPeaks));

Zmesh(1:length(PickPeaks))          = {zeros(length(y),length(x))};
AbsZmesh(1:length(PickPeaks))       = {zeros(length(y),length(x))};
ModeShape(1:length(PickPeaks),1)    = {[]};
AbsModeShape(1:length(PickPeaks),1) = {[]};

alldata(1).ztC = zeros(size(alldata,2),size(alldata,2));

%% Find the local maxima
for k = 1 : size(alldata,2)
    ImagH(k,:) = alldata(k).ztCompliance.ImagH;
    RealH(k,:) = alldata(k).ztCompliance.RealH;
    AbsH(k,:)  = alldata(k).ztCompliance.AbsH;
    Coh(k,:)   = alldata(k).ztCompliance.Coh;
    f(k,:)     = alldata(k).ztCompliance.f;
end

% Global curve fitting
SumImagReal = (sum(abs(ImagH.*Coh),1)./sum(abs(RealH.*Coh),1));
SumAbs      = sum(AbsH,1);

% Find the peaks of the summed frequency response function
[~, SumAbsPeaksLoc, SumAbsPeaksWidth] = findpeaks(SumAbs,'WidthReference','halfheight',...
                                                                    'MinPeakProminence',1.0);

% Find the peaks of the absolute frequency response functions for each measurement
for k = 1 : size(alldata,2)
    [alldata(k).ztPeaksMag, alldata(k).ztPeaksLoc, alldata(k).ztPeaksWidth] = findpeaks(AbsH(k,:));
end

%% Natural frequencies
% Natural frequencies Omega
alldata(1).ztOmega  = f(1,SumAbsPeaksLoc);
```

```
% Eigenvalue matrix Lambda
alldata(1).ztLambda = diag(f(1,SumAbsPeaksLoc).^2);

%% Natural modes
% Calculate the natural modes for the defined natural frequencies
for i = 1 : size(alldata,2)
    for k = 1 : size(alldata,2)
        V(k,i)    = alldata(k).ztCompliance.ImagH(SumAbsPeaksLoc(i));
        AbsV(k,i) = alldata(k).ztCompliance.AbsH(SumAbsPeaksLoc(i));
    end
end

% Store data
alldata(1).V    = V;
alldata(1).AbsV = AbsV;

%% Fitting the mode shapes
% Define mesh for surface fits
[Xmesh,Ymesh] = meshgrid(x,y);

for i = 1 : length(PickPeaks)
    % Define coordinates
    for k = 1 : size(alldata,2)
        xpos = alldata(k).HammerPosition(1);
        ypos = alldata(k).HammerPosition(2);

        X(k,i)    = x(xpos);
        Y(k,i)    = y(ypos);
        Z(k,i)    = alldata(1).V(k,PickPeaks(i));
        AbsZ(k,i) = alldata(1).AbsV(k,PickPeaks(i));

        Zmesh{i}(ypos,xpos)    = Z(k,i);
        AbsZmesh{i}(ypos,xpos) = AbsZ(k,i);
    end

    % Fitting Mode shapes (5th degree polynomial)
    ModeShape{i}    = fit([X(:,i),Y(:,i)],Z(:,i),'poly55');
    AbsModeShape{i} = fit([X(:,i),Y(:,i)],AbsZ(:,i),'poly55');
end

%% Damping coefficient
for i = 1 : size(alldata,2)
    % Calculate the damping from the intersections with the PeaksMag/sqrt(2) lines
    % PeaksWidth = 50% height, sqrt(2) = 70% height, approximation -> *(2/sqrt(2))
    Omega1 = alldata(1).ztOmega(i) - 1/2 * SumAbsPeaksWidth(i) * (2/sqrt(2));
    Omega2 = alldata(1).ztOmega(i) + 1/2 * SumAbsPeaksWidth(i) * (2/sqrt(2));
    Xi     = (Omega2 - Omega1) / (2 * alldata(1).ztOmega(i));

    % Modal damping matrix
    alldata(1).ztC(i,i) = alldata(1).ztOmega(i)*Xi;     % sigma = xi * omega_n
end

%% Plot figures?
if (PlotFigures == 1)
%% Plotting local maxima
figure(1)
pl1 = subplot(2,1,1);
findpeaks(SumImagReal,f(1,:),'WidthReference','halfheight','MinPeakProminence',0.2,'Annotate','extents')
xlabel('Frequency (Hz)'); ylabel('Compliance (m/N)')
title('Weighted Imaginary/Real part of the summed frequency response function')

pl2 = subplot(2,1,2);
findpeaks(SumAbs,f(1,:),'WidthReference','halfheight','MinPeakProminence',1.0,'Annotate','peaks');
xlabel('Frequency (Hz)'); ylabel('Compliance (m/N)')
set(pl2,'YScale','log'); linkaxes([pl1,pl2],'x'); xlim([1 1e3])
title('Absolute value of the summed frequency response function')

%% Plotting natural modes (2D)
for i = 1 : length(PickPeaks)
    fn = alldata(1).ztOmega(PickPeaks(i));

    figure(2+i)
    sp1 = subplot(2,3,1);
    surf(Xmesh,Ymesh,Zmesh{i}); hold on;
    stem3(X(:,i),Y(:,i),Z(:,i),'k');
    xlabel('x (m)')
    ylabel('y (m)')
    zlabel('Compliance (m/N)')
    title(['imag(H(f)) mode shape ', num2str(i),' (Omega = ',num2str(fn),')'])
```

```
    sp2 = subplot(2,3,2);

    cp1 = plot(ModeShape{i}); view(2)
    xlabel('x (m)'); ylabel('y (m)')
    title(['imag(H(f)) mode shape ', num2str(i),' (Omega = ',num2str(fn),')'])

    sp3 = subplot(2,3,3);
    plot(ModeShape{i}); hold on;
    stem3(X(:,i),Y(:,i),Z(:,i),'k');
    xlabel('x (m)'); ylabel('y (m)'); zlabel('Compliance (m/N)')
    title(['Fitting of imag(H(f)) mode shape ', num2str(i),' (Omega = ',num2str(fn),')'])

    sp4 = subplot(2,3,4);
    surf(Xmesh,Ymesh,AbsZmesh{i}); hold on;
    stem3(X(:,i),Y(:,i),AbsZ(:,i),'k');
    xlabel('x (m)'); ylabel('y (m)'); zlabel('Compliance (m/N)')
    title(['|H(f)| mode shape ', num2str(i),' (Omega = ',num2str(fn),')'])

    sp5 = subplot(2,3,5);
    cp2 = plot(AbsModeShape{i}); view(2)
    xlabel('x (m)'); ylabel('y (m)')
    title(['imag(H(f)) mode shape ', num2str(i),' (Omega = ',num2str(fn),')'])

    sp6 = subplot(2,3,6);
    plot(AbsModeShape{i}); hold on;
    stem3(X(:,i),Y(:,i),AbsZ(:,i),'k');
    xlabel('x (m)'); ylabel('y (m)'); zlabel('Compliance (m/N)')
    title(['Fitting of |H(f)| mode shape ', num2str(i),' (Omega = ',num2str(fn),')'])

    daspect(sp1, [1,1,1]);
    set([cp1,cp2],'linestyle','none'); set(sp1,'XLim',[min(x),max(x)],'YLim',[min(y),max(y)])
    linkprop([sp1,sp2,sp3,sp4,sp5,sp6],'DataAspectRatio');
    linkprop([sp1,sp2,sp3,sp4,sp5,sp6],'XLim');
    linkprop([sp1,sp2,sp3,sp4,sp5,sp6],'YLim');
end
end
```

Table 11.9: MATLAB function QuadraturePicking2D.m.

Appendix J MATLAB script Curve fitting – Single mode contribution scaling

The single mode contribution scaling script is used as a way of curve fitting, as explained in section 8.3.

```
%% Single mode contribution scaling

% Calculate the multiplication constants to place the single mode contributions on the resonance peaks
for counter = 1 : 4;
    A(counter) = abs(HEoM(1,SumAbsPeaksLoc(counter))) / abs(Zr{1,counter}(SumAbsPeaksLoc(counter)));
end

for k = 1 : length(s)
    for i = 1 : size(M,1)
        for j = 1 : size(VPicking,1)
            % Modal dynamic matrix
            DModal   = I(i,i) * s(k)^2 + CModal(i,i) * s(k) + Lambda(i,i);

            % Apply scaling
            VTranspose = VPicking';
            VTranspose(1,2) = VTranspose(1,2) * A(1);
            VTranspose(2,2) = VTranspose(2,2) * A(2);
            VTranspose(3,2) = VTranspose(3,2) * A(3);
            VTranspose(4,2) = VTranspose(4,2) * A(4);

            % DModal * Eta = (V' * F) => Eta = DModal\(V' * F)
            Eta(i,k) = DModal\(VTranspose(i,2));

            % Zr = V11 * Eta1 + V12 * Eta2 + V13 * Eta1 + V13 * Eta1
            Zr{j,i}(k)  = VPicking(j,i) * Eta(i,k);
        end
    end
end

for k = 1 : length(s)
    for i = 1 : size(Zr,1)
        HPickingStep1(i,k) = Zr{i,1}(k) + Zr{i,2}(k) + Zr{i,3}(k) + Zr{i,4}(k);
    end
end

%% Shifting the frequency response function
VPicking(1,:) = VPicking(1,:) * (HEoM(1,1)/HPickingStep1(1,1));
VPicking(2,:) = VPicking(2,:) * (HEoM(2,1)/HPickingStep1(2,1));
VPicking(3,:) = VPicking(3,:) * (HEoM(3,1)/HPickingStep1(3,1));
VPicking(4,:) = VPicking(4,:) * (HEoM(4,1)/HPickingStep1(4,1));

for k = 1 : length(s)
    for i = 1 : size(M,1)
        for j = 1 : size(VPicking,1)
            % Modal dynamic matrix
            DModal   = I(i,i) * s(k)^2 + CModal(i,i) * s(k) + Lambda(i,i);

            % DModal * Eta = (V' * F) => Eta = DModal\(V' * F)
            Eta(i,k) = DModal\(VTranspose(i,2));

            % Zr = V11 * Eta1 + V12 * Eta2 + V13 * Eta1 + V13 * Eta1
            Zr{j,i}(k)  = VPicking(j,i) * Eta(i,k);
        end
    end
end

for k = 1 : length(s)
    for i = 1 : size(Zr,1)
        HPickingStep2(i,k) = Zr{i,1}(k) + Zr{i,2}(k) + Zr{i,3}(k) + Zr{i,4}(k);
    end
end
```

Table 11.10: MATLAB script Curve fitting – Single mode contribution scaling.

Appendix K MATLAB script Curve fitting – mean square error optimisation

The mean square error optimisation script is used for curve fitting, to minimise the mean square error, as explained in section 8.4.

```
%% Curve fitting - Mean square error

% Normalisation
V = V/norm(V);

% Weight factor
W(1,1:length(f)) = 1;
W(1,10:50)       = 10;
W(1,50:end)      = 5;

% Optimisation problem
Options = optimoptions(@fmincon, 'Algorithm', 'interior-
point','MaxIterations',5500,'MaxFunctionEvaluations',5500);
ObjectiveFunction = @(V) OptimisationFunction2(SensorLoc,f,C,Lambda,H,W,V);
Problem = createOptimProblem('fmincon','x0',V,'objective',ObjectiveFunction,'options',Options);

% Minimise the mean square error
MS  = MultiStart('Display','iter','UseParallel',true);
GS  = GlobalSearch('Display','iter');
[V] = run(MS,Problem,1);        % Choose: Multistart or GlobalSearch

% Normalisation
V = V/norm(V);
VModal = V;

% Regenerate the frequency response function
s = f * 1i;                     % Laplace parameter
F = eye(36);                    % Force matrix
I = eye(36);                    % Identity matrix

HModal(size(I,1),size(I,2),length(f)) = 0;

for k = 1 : length(s)
    DModal = I * s(k)^2 + C * s(k) + Lambda;    % Modal dynamic matrix
    Eta    = DModal\(V' * F);                   % DModal * Eta = (V' * F) => Eta = DModal\(V' * F)

    % Calculate the modal frequency response functions
    HModal(:,:,k) = V * Eta;
end

%% Shifting the frequency response function
for l = 1 : length(x)*length(y)
    A(1:36,l) = mean(abs(H(SensorLoc,l,2:20)) ./ abs(HModal(SensorLoc,l,2:20)));
end

for i = 1 : size(HModal,1)
    for j = 1 : size(HModal,2)
        HModal(i,j,:) = HModal(i,j,:).*A(i,j);
    end
end
```

Table 11.11: MATLAB script Curve fitting – mean square error optimisation.

```
%% Optimisation function
function ObjectiveFunction = OptimisationFunction3(SensorLoc,f,C,Lambda,H,W,V)
% Normalisation
V = V/norm(V);

s = f * 1i;          % Laplace parameter
F = eye(36);         % Force matrix
I = eye(36);         % Identity matrix

HModal(size(I,1),size(I,2),length(f)) = 0;

% Calculate the modal frequency response functions with scaled damping
for k = 1 : length(s)
    DModal      = (I * s(k)^2 + C * s(k) + Lambda);    % Modal dynamic matrix
    Eta         = DModal\(V' * F);                     % DModal * Eta = (V' * F) => Eta = DModal\(V' * F)

    % Calculate the modal frequency response functions
    % Changing the model by adding an extra zero to the frequency response functions
    HModal(:,:,k) = V * Eta * s(k);
end

% Calculate the error
for c = 1
    Error(c,:) = reshape(abs(H(SensorLoc,c,:)),1,length(s),[])...
        - reshape(abs(HModal(SensorLoc,c,:)),1,length(s),[]);
end

% Minimise the error
ObjectiveFunction = sum(sum(sum( W .* Error.^2 )));
end
```

Table 11.12: MATLAB script Curve fitting – Optimisation function

Appendix L MATLAB script Compensation residues

The residue compensation script is used to improve the curve fit by addition of extra residues, as explained in section 8.4.2.

```
%% Compensation residues
ResiduesPeaks = 2*Peaks(end):10:150;
C(2:length(ResiduesPeaks)+1,2:length(ResiduesPeaks)+1) = diag(20*C(1,1)*ones(1,length(ResiduesPeaks)));

for l = 1 : length(x)*length(y)
    for k = 1 : length(ResiduesPeaks)
        B(1:36,l,k) = ( abs(mean(H(SensorLoc,l,ResiduesPeaks(k)-2:ResiduesPeaks(k)+2)))...
            - abs(mean(HModal(SensorLoc,l,ResiduesPeaks(k)-2:ResiduesPeaks(k)+2))) )...
            * abs( s(ResiduesPeaks(k))^2 + C(k+1,k+1) * s(ResiduesPeaks(k)) + f(ResiduesPeaks(k))^2 );
    end
end

HModalResidues = HModal;

for i = 1 : size(HModal,1)
    for j = 1 : size(HModal,2)
        for k = 1 : length(ResiduesPeaks)
            HModalResidues(i,j,Peaks(end):end) = HModalResidues(i,j,Peaks(end):end)...
                + reshape( B(i,j,k)./( s(Peaks(end):end).^2 + C(k+1,k+1) * s(Peaks(end):end)...
                + f(ResiduesPeaks(k))^2 ), 1,1,length(s(Peaks(end):end)));
        end
    end
end
```

Table 11.13: MATLAB script Compensation residues.

Appendix M MATLAB GUI - QuadraturePicking

Figure 11.1 shows a preview of the quadrature picking GUI. This program can be used to interactively select the resonance peaks of a measurement, after which the program calculates the mode shapes corresponding with the selected natural frequencies.

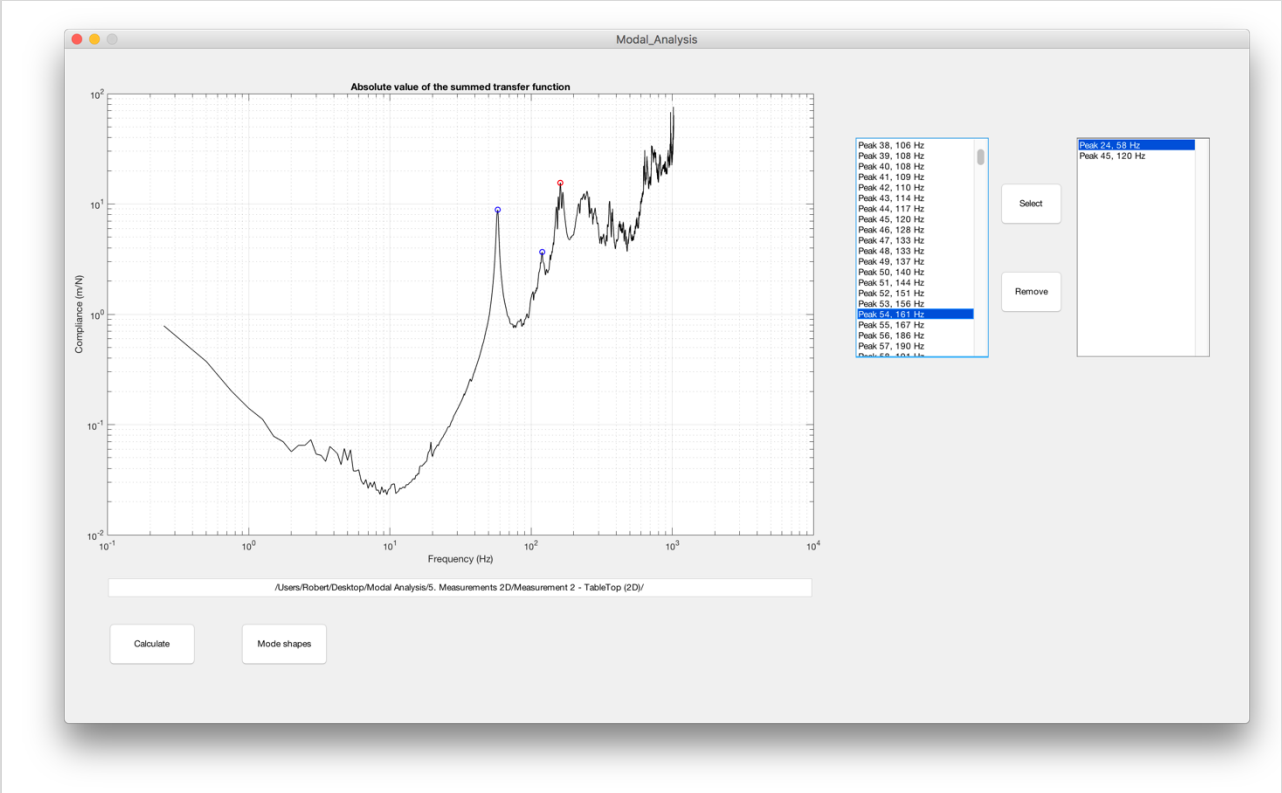


Figure 11.1: Preview of the quadrature picking GUI.

Instructions

- First enter the path where the measurement data is located in the textbox. The name of the measurement data should have the following format:
  - 'MeasurementName_sf(x,y)_ha(x,y).mat'

  Where sf(x,y) is the position of the fixed sensor and ha(x,y) the excitation position. As an example, the measurement data used here has the following name:
  - 'Measurement2_sf(9,6)_ha(1,1).mat'
- Next press calculate. The program will generate a plot of the averaged absolute frequency response function.
- Select all resonance peaks for which mode shapes should be calculated. This is done by selecting the peak from the list and pressing select. The peak frequency is moved to the list on the right. To remove peaks, select the peak from the right list and press remove.
- Press Mode shapes to generate the mode shape plots. These plots consist of the mode shapes derived from the modal matrix, a 5^{th} degree polynomial fit of the mode shapes and a top view contour plot.

```
%% Robert Giesen, Internship MECAL
function varargout = QuadraturePicking(varargin)
% QUADRATUREPICKING MATLAB code for QuadraturePicking.fig
%      QUADRATUREPICKING, by itself, creates a new QUADRATUREPICKING or raises the existing
%      singleton*.
%
%      H = QUADRATUREPICKING returns the handle to a new QUADRATUREPICKING or the handle to
%      the existing singleton*.
%
%      QUADRATUREPICKING('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in QUADRATUREPICKING.M with the given input arguments.
%
%      QUADRATUREPICKING('Property','Value',...) creates a new QUADRATUREPICKING or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before QuadraturePicking_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to QuadraturePicking_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 QuadraturePicking

% Last Modified by GUIDE v2.5 15-Nov-2017 13:28:27

%% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @QuadraturePicking_OpeningFcn, ...
    'gui_OutputFcn',  @QuadraturePicking_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 QuadraturePicking is made visible.
function QuadraturePicking_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 QuadraturePicking (see VARARGIN)

% Choose default command line output for QuadraturePicking
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes QuadraturePicking wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%% --- Outputs from this function are returned to the command line.
function varargout = QuadraturePicking_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;

function DataPath_Callback(hObject, eventdata, handles)
% hObject    handle to DataPath (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 DataPath as text
%        str2double(get(hObject,'String')) returns contents of DataPath as a double
```

```
%% --- Executes during object creation, after setting all properties.
function DataPath_CreateFcn(hObject, eventdata, handles)
% hObject    handle to DataPath (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%% --- Executes during object creation, after setting all properties.
function PeakList_CreateFcn(hObject, eventdata, handles)
% hObject    handle to PeakList (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%% --- Executes during object creation, after setting all properties.
function SelectedPeaks_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SelectedPeaks (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%% --- Executes on button press in Calculate (Calculate).
function Calculate_Callback(hObject, eventdata, handles)
% hObject    handle to Calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% Clear previous calculations
clc; cla(handles.Peaks); set(handles.PeakList,'String',[]); %set(handles.SelectedPeaks,'String',[]);

%% Initialisation
MeasurementPath = get(handles.DataPath,'String');

x = [0.15 0.3 0.45 0.6 0.75 0.9 1.05 1.2 1.35];
y = [0.1 0.2 0.25 0.35 0.4 0.55 0.6];

window = 0;

% Define the points and range which have to be averaged
SuperPoints = [2 2; 2 6; 5 2; 5 6; 8 2; 8 6];

%% 1. Gathering data
handles = GatherData_Callback(hObject, eventdata, handles, MeasurementPath, window);

%% 2. Peak picking
handles = PeakPicking_Callback(hObject, eventdata, handles, SuperPoints);

%% --- Executes on button press in ModeShapes.
function ModeShapes_Callback(hObject, eventdata, handles)
% hObject    handle to ModeShapes (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% Clear previous calculations
clc;

%% Initialisation
x = [0.15 0.3 0.45 0.6 0.75 0.9 1.05 1.2 1.35];
y = [0.1 0.2 0.25 0.35 0.4 0.55 0.6];

% Define the points and range which have to be averaged
SuperPoints = [2 2; 2 6; 5 2; 5 6; 8 2; 8 6];

% Resulting peaks from the peak picking
AllSelectedPeaks = get(handles.SelectedPeaks,'string');
SumAbsPeaksLoc   = handles.alldata(1).ztCompliance.SumAbsPeaksLoc;
```

```
for i = 1 : size(AllSelectedPeaks,1)
    PickPeaks(i,:) =
SumAbsPeaksLoc(str2double(AllSelectedPeaks{i,:}((strfind(AllSelectedPeaks{i,:},'Peak
')+5):(strfind(AllSelectedPeaks{i,:},',')-1))));
end

% Sorting the found peaks
PickPeaks = sort(PickPeaks);

%% 2. Quadrature picking
handles = QuadraturePicking_Callback(hObject, eventdata, handles, SuperPoints, PickPeaks, x, y);

%% --- Executes on selection change in PeakList.
function PeakList_Callback(hObject, eventdata, handles)
% hObject    handle to PeakList (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns PeakList contents as cell array
%        contents{get(hObject,'Value')} returns selected item from PeakList
%% Initialisation
% Get selected peak
PeakList = get(hObject,'string');
CurrentPeak = PeakList{get(hObject,'value')};
CurrentPeakNumber = str2double(CurrentPeak((strfind(CurrentPeak,'Peak ')+5):(strfind(CurrentPeak,',')...
                                                                                -1)));

%% Set visibility of the selected peak
set(handles.PeakPickingPlot,'Xdata',handles.PeakPickingPlotx,'Ydata',handles.PeakPickingPlotx,'visible',
'off')
set(handles.PeakPickingPlot,'Xdata',handles.PeakPickingPlotx(CurrentPeakNumber),'Ydata',...
                                          handles.PeakPickingPloty(CurrentPeakNumber),'visible','on')

%% --- Executes on selection change in SelectedPeaks.
function SelectedPeaks_Callback(hObject, eventdata, handles)
% hObject    handle to SelectedPeaks (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns SelectedPeaks contents as cell array
%        contents{get(hObject,'Value')} returns selected item from SelectedPeaks

%% --- Executes on button press in Select.
function Select_Callback(hObject, eventdata, handles)
% hObject    handle to Select (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% Initialisation
handles = guidata(hObject);

f = handles.alldata(1).ztCompliance.f;
SumAbsPeaksMag = handles.alldata(1).ztCompliance.SumAbsPeaksMag;
SumAbsPeaksLoc = handles.alldata(1).ztCompliance.SumAbsPeaksLoc;

% Get selected peak
PeakList = get(handles.PeakList,'string');
CurrentPeak = PeakList{get(handles.PeakList,'value')};

%% Add to list of selected peaks
AllSelectedPeaks = get(handles.SelectedPeaks,'string');

if (isempty(AllSelectedPeaks) == 1)
    set(handles.SelectedPeaks,'String',{CurrentPeak})
else
    set(handles.SelectedPeaks,'String',{AllSelectedPeaks{:},CurrentPeak})
end

%% Plot selected peak
AllSelectedPeaks = get(handles.SelectedPeaks,'string');

% Define the peaks to plot
for i = 1 : length(AllSelectedPeaks)
    PlotSelectedPeaks(i) = str2double(AllSelectedPeaks{i}((strfind(AllSelectedPeaks{i},...
                                                    'Peak ')+5):(strfind(AllSelectedPeaks{i},',')-1)));
end

axes(handles.Peaks)

handles.PeaksPlot = loglog(f(SumAbsPeaksLoc(PlotSelectedPeaks)),SumAbsPeaksMag(PlotSelectedPeaks),...
                                                                        'ob','MarkerSize',6);
```

```
guidata(hObject,handles)

%% --- Executes on button press in Remove.
function Remove_Callback(hObject, eventdata, handles)
% hObject    handle to Remove (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%% Initilisation
AllSelectedPeaks = get(handles.SelectedPeaks,'string');

%% Remove selected peak from list
for i = 1 : get(handles.SelectedPeaks,'value')-1
    NewAllSelectedPeaks{i} = AllSelectedPeaks{i};
end
for i = get(handles.SelectedPeaks,'value')+1 : size(AllSelectedPeaks,1)
    NewAllSelectedPeaks{i-1} = AllSelectedPeaks{i};
end

set(handles.SelectedPeaks,'String',NewAllSelectedPeaks);

%% --- GatherData
function handles = GatherData_Callback(hObject, eventdata, handles, MeasurementPath, window)
%% Initialisation
handles = guidata(hObject);

%% Read data
file_list = dir([MeasurementPath,'*.mat']);

for k = 1 : length(file_list)
    fl(k,:) = file_list(k).name;
end

fl = sortrows(fl,[2,3]);

%% Store data
for k = 1 : size(fl,1)
    data = load([MeasurementPath fl(k,:)]); handles.alldata(k).meas = data.StiffGlob.meas;

    FixedSensorPosition = [str2num(fl(k,strfind(fl(k,:),'sf(')+3)),
str2num(fl(k,strfind(fl(k,:),'sf(')+5))];
    HammerPosition      = [str2num(fl(k,strfind(fl(k,:),'ha(')+3)),
str2num(fl(k,strfind(fl(k,:),'ha(')+5))];

    % Store all data in structure
    handles.alldata(k).FixedSensorPosition = FixedSensorPosition;
    handles.alldata(k).HammerPosition      = HammerPosition;

    % Calculate the sample rate
    dt = handles.alldata(k).meas(1).t(2) - handles.alldata(k).meas(1).t(1);

    for i = 1 : size(handles.alldata(k).meas,2)
        Hamt(:,i) = handles.alldata(k).meas(i).Hamt;
        zt(:,i)   = handles.alldata(k).meas(i).zt;
        yt(:,i)   = handles.alldata(k).meas(i).yt;
    end

    % FixedSensor (yt)
    [handles.alldata(k).ytStiffness.f, handles.alldata(k).ytStiffness.H,
handles.alldata(k).ytStiffness.AbsH, handles.alldata(k).ytStiffness.Phase,...
        handles.alldata(k).ytStiffness.Coh] = HAvgAndCoh(Hamt, yt, 1/dt, window);

    [handles.alldata(k).ytCompliance.f, handles.alldata(k).ytCompliance.H,

handles.alldata(k).ytCompliance.AbsH, handles.alldata(k).ytCompliance.Phase,...
        handles.alldata(k).ytCompliance.Coh] = HAvgAndCoh(yt, Hamt, 1/dt, window);

    % Imaginary and real parts of the stiffness
    handles.alldata(k).ytStiffness.ImagH = imag( handles.alldata(k).ytStiffness.H );

handles.alldata(k).ytStiffness.RealH = real( handles.alldata(k).ytStiffness.H );

    % Imaginary and real parts of the compliance
    handles.alldata(k).ytCompliance.ImagH = imag( handles.alldata(k).ytCompliance.H );
    handles.alldata(k).ytCompliance.RealH = real( handles.alldata(k).ytCompliance.H );

    % ColocatedSensor (zt)
    [handles.alldata(k).ztStiffness.f, handles.alldata(k).ztStiffness.H,
handles.alldata(k).ztStiffness.AbsH, handles.alldata(k).ztStiffness.Phase,...
        handles.alldata(k).ztStiffness.Coh]  = HAvgAndCoh(Hamt, zt, 1/dt, window);
```

```
    [handles.alldata(k).ztCompliance.f, handles.alldata(k).ztCompliance.H,
handles.alldata(k).ztCompliance.AbsH, handles.alldata(k).ztCompliance.Phase,...
        handles.alldata(k).ztCompliance.Coh] = HAvgAndCoh(zt, Hamt, 1/dt, window);

    % Imaginary and real parts of the stiffness
    handles.alldata(k).ztStiffness.ImagH = imag( handles.alldata(k).ztStiffness.H );
    handles.alldata(k).ztStiffness.RealH = real( handles.alldata(k).ztStiffness.H );

    % Imaginary and real parts of the compliance
    handles.alldata(k).ztCompliance.ImagH = imag( handles.alldata(k).ztCompliance.H );
    handles.alldata(k).ztCompliance.RealH = real( handles.alldata(k).ztCompliance.H );
end

guidata(hObject,handles)

%% --- PeakPicking
function handles = PeakPicking_Callback(hObject, eventdata, handles, SuperPoints)
%% Initialisation
handles = guidata(hObject);

DataSize = size(handles.alldata,2);

for k = 1 : DataSize
    ImagH(k,:)          = handles.alldata(k).ztCompliance.ImagH;
    RealH(k,:)          = handles.alldata(k).ztCompliance.RealH;
    H(k,:)              = handles.alldata(k).ztCompliance.H;
    AbsH(k,:)           = handles.alldata(k).ztCompliance.AbsH;
    HammerPosition(k,:) = handles.alldata(k).HammerPosition;
end

f = handles.alldata(1).ztCompliance.f;

% Preallocate Memory
AveragingSet(1:size(SuperPoints,1),1) = {[]};
SuperPointsIndex = [];

%% Calculate average frequency response function for all measurements
% Global curve fitting
SumImagReal = (sum(abs(ImagH),1)./sum(abs(RealH),1));
SumAbs      = sum(AbsH,1);

%% Find the local maxima
[SumAbsPeaksMag, SumAbsPeaksLoc, SumAbsPeaksWidth] = findpeaks(SumAbs,'WidthReference','halfheight');

%% Return data to figure
axes(handles.Peaks)
loglog(f,SumAbs,'k','Linewidth',0.8); hold on; grid on;
PeakPickingPlot = loglog(f(SumAbsPeaksLoc),SumAbsPeaksMag,'or','MarkerSize',6,'visible','off');
xlabel('Frequency (Hz)'); ylabel('Compliance (m/N)')
title('Absolute value of the summed frequency response function')

%% Return data to listbox
for i = 1 : length(SumAbsPeaksLoc)
    Frequentie = num2str(round(f(SumAbsPeaksLoc(i))));
    PeakList(i,1) = strcat({'Peak '},num2str(i),{', '},Frequentie,{' Hz'});
end
set(handles.PeakList,'String',PeakList)

%% Store data
handles.alldata(1).ztCompliance.SumAbs           = SumAbs;
handles.alldata(1).ztCompliance.SumImagReal      = SumImagReal;
handles.alldata(1).ztCompliance.SumAbsPeaksMag   = SumAbsPeaksMag;
handles.alldata(1).ztCompliance.SumAbsPeaksLoc   = SumAbsPeaksLoc;
handles.alldata(1).ztCompliance.SumAbsPeaksWidth = SumAbsPeaksWidth;
handles.SuperPointsIndex = SuperPointsIndex;
handles.PeakPickingPlot  = PeakPickingPlot;
handles.PeakPickingPlotx = get(PeakPickingPlot, 'Xdata');
handles.PeakPickingPloty = get(PeakPickingPlot, 'Ydata');

guidata(hObject,handles)

%% --- QuadraturePicking
function handles = QuadraturePicking_Callback(hObject, eventdata, handles, SuperPoints, PickPeaks, x, y)
%% Initialisation
handles = guidata(hObject);

DataSize = size(handles.alldata,2);

for k = 1 : DataSize
    ImagH(k,:)          = handles.alldata(k).ztCompliance.ImagH;
```

```
    AbsH(k,:)           = handles.alldata(k).ztCompliance.AbsH;
    HammerPosition(k,:) = handles.alldata(k).HammerPosition;
end

f                = handles.alldata(1).ztCompliance.f;
SumAbsPeaksLoc   = handles.alldata(1).ztCompliance.SumAbsPeaksLoc;
SumAbsPeaksWidth = handles.alldata(1).ztCompliance.SumAbsPeaksWidth;
SuperPointsIndex = handles.SuperPointsIndex;

%% Natural frequencies
Omega  = f(SumAbsPeaksLoc);                  % Natural frequencies Omega
Lambda = diag(f(SumAbsPeaksLoc).^2);         % Eigenvalue matrix Lambda

%% Natural modes
for i = 1 : length(PickPeaks)
    for k = 1 : DataSize
        V(k,i)    = ImagH(k,PickPeaks(i));
        AbsV(k,i) = AbsH(k,PickPeaks(i));
    end
end

%% Fitting the mode shapes
% Define mesh for surface fits
[Xmesh,Ymesh] = meshgrid(x,y);

for i = 1 : length(PickPeaks)

    % Define coordinates
    for k = 1 : DataSize
        xpos = HammerPosition(k,1);
        ypos = HammerPosition(k,2);

        X(k,i)    = x(xpos);
        Y(k,i)    = y(ypos);
        Z(k,i)    = V(k,i);
        AbsZ(k,i) = AbsV(k,i);
        Zmesh{i}(ypos,xpos)    = V(k,i);
        AbsZmesh{i}(ypos,xpos) = AbsV(k,i);
    end

    % Fitting Mode shapes (5th degree polynomial)
    ModeShape{i}    = fit([X(:,i),Y(:,i)],Z(:,i),'poly55');
    AbsModeShape{i} = fit([X(:,i),Y(:,i)],AbsZ(:,i),'poly55');
end

%% Damping coefficient
for i = 1 : DataSize
    % Calculate the damping from the intersections with the PeaksMag/sqrt(2) lines
    % PeaksWidth = 50% height, sqrt(2) = 70% height, approximation -> *(2/sqrt(2))
    Omega1 = Omega(i) - 1/2 * SumAbsPeaksWidth(i) * (2/sqrt(2));
    Omega2 = Omega(i) + 1/2 * SumAbsPeaksWidth(i) * (2/sqrt(2));
    Xi     = (Omega2 - Omega1) / (2 * Omega(i));

    % Modal damping matrix
    C(i,i) = Omega(i) * Xi;     % sigma = xi * omega_n
end

%% Plotting the natural modes
for i = 1 : length(PickPeaks)
    figure(2+i)
    sp1 = subplot(2,3,1);
    surf(Xmesh,Ymesh,Zmesh{i}); hold on;
    stem3(X(:,i),Y(:,i),Z(:,i),'k');
    xlabel('x (m)'); ylabel('y (m)'); zlabel('Compliance (m/N)')
    title(['imag(H(f)) mode shape ', num2str(i),' (Omega = ',num2str(f(PickPeaks(i))),')'])

    sp2 = subplot(2,3,2);
    cp1 = plot(ModeShape{i},'Style','Contour'); view(2)
    xlabel('x (m)'); ylabel('y (m)')
    title(['imag(H(f)) mode shape ', num2str(i),' (Omega = ',num2str(f(PickPeaks(i))),')'])

    sp3 = subplot(2,3,3);
    plot(ModeShape{i}); hold on;
    stem3(X(:,i),Y(:,i),Z(:,i),'k');
    xlabel('x (m)'); ylabel('y (m)'); zlabel('Compliance (m/N)')
    title(['Fitting of imag(H(f)) mode shape ', num2str(i),' (Omega = ',num2str(f(PickPeaks(i))),')'])

    sp4 = subplot(2,3,4);
    surf(Xmesh,Ymesh,AbsZmesh{i}); hold on;
    stem3(X(:,i),Y(:,i),AbsZ(:,i),'k');
```

```
    xlabel('x (m)'); ylabel('y (m)'); zlabel('Compliance (m/N)')
    title(['|H(f)| mode shape ', num2str(i),' (Omega = ',num2str(f(PickPeaks(i))),')'])

    sp5 = subplot(2,3,5);
    cp2 = plot(AbsModeShape{i},'Style','Contour'); view(2)
    xlabel('x (m)'); ylabel('y (m)')
    title(['imag(H(f)) mode shape ', num2str(i),' (Omega = ',num2str(f(PickPeaks(i))),')'])

    sp6 = subplot(2,3,6);
    plot(AbsModeShape{i}); hold on;
    stem3(X(:,i),Y(:,i),AbsZ(:,i),'k');
    xlabel('x (m)'); ylabel('y (m)'); zlabel('Compliance (m/N)')
    title(['Fitting of |H(f)| mode shape ', num2str(i),' (Omega = ',num2str(f(PickPeaks(i))),')'])

    daspect(sp1, [1,1,1]);
    set([cp1,cp2],'linestyle','none'); set(sp1,'XLim',[min(x),max(x)],'YLim',[min(y),max(y)])
    linkprop([sp1,sp2,sp3,sp4,sp5,sp6],'DataAspectRatio');
    linkprop([sp1,sp2,sp3,sp4,sp5,sp6],'XLim');
    linkprop([sp1,sp2,sp3,sp4,sp5,sp6],'YLim');
end

%% Store data
handles.alldata(1).V      = V;
handles.alldata(1).AbsV   = AbsV;
handles.alldata(1).ztC    = C;
handles.alldata(1).Lambda = Lambda;

guidata(hObject,handles)
```

Table 11.14: MATLAB GUI – QuadraturePicking.