Investigation of the Structural Dynamics in an Automotive A/C System

Master's thesis in Applied Mechanics

MÅRTEN AGNESSON
NIKLAS KARLSSON

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis in Applied Mechanics

Investigation of the Structural Dynamics in an Automotive A/C System

MÅRTEN AGNESSON
NIKLAS KARLSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Applied Mechanics
Division of Dynamics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Investigation of the Structural Dynamics in an Automotive A/C System
MÅRTEN AGNESSON, NIKLAS KARLSSON

Supervisor: PhD Reza Okhovat, Department of Analytics, Alten AB
Examiner: Prof. Thomas Abrahamsson, Department of Applied Mechanics, Chalmers University of Technology

Master's Thesis 2017:03
Department of Applied Mechanics
Division of Dynamics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover:
Picture of the investigated A/C system.

Printed by Department of Applied Mechanics
Gothenburg, Sweden 2017

Investigation of the Structural Dynamics in an Automotive A/C System
Master's thesis in Applied Mechanics
MÅRTEN AGNESSON, NIKLAS KARLSSON
Department of Applied Mechanics
Division of Dynamics
Chalmers University of Technology

Abstract

In the range of detectable sounds inside a car coupe interior noise and vibration induced by the A/C system is present. As engines become more efficient and hybrid cars more common this source gets more prominent. The current study has investigated unwanted resonance in the A/C-system. Previous studies has shown that the noise inside the coupe correlates to measured vibrations at the end of the A/C system. Based on these studies a section for the vibration the transfer path, between the compressor and the thermal expansion valve, was considered during this project. The objective of the study was to numerically simulate the structural vibration behavior in the structure. To do so a FE model was built, validated and calibrated to experimental data obtained from modal analysis on the physical system. The procedure of testing with methods of both pre-test planning and post analysis of the measurement is presented. FE modelling was done in ANSA and a normal modes analysis was conducted using Nastran. Post processing of test results was done in Matlab. Validation of results obtained from numerical simulation and experimental modal analysis was performed. The percental deviation between the corresponding eigenfrequencies were calculated and the eigenmodes were evaluated using the modal assurance criterion. A parametrization of the FE model was performed for material- and part properties. The model was then calibrated, using FEMcali, to fit better to the experimental test data. The results showed that the numerical model could predict the eigenmodes with better accuracy after the calibration. Numerical results had a mean deviation to the test of less than ten percent for eigenmodes below 500 Hz.

Finally, an investigation was conducted on how two different added mounting points to the car chassis could influence the resonance in the system. The methodology of correlating numerical results to physical testing introduces sources of error during the validation process and the signification of performing it in an iterative manner is discussed.

Keywords: Nastran, Normal Modes Analysis, ANSA, FE modelling, LMS, Experimental modal analysis, Modal Assurance Criterion, FEMcali.

En undersökning av strukturdynamiken i ett fordons A/C-system
Examensarbete inom Tillämpad Mekanik
MÅRTEN AGNESSON, NIKLAS KARLSSON
Institutionen för Tillämpad Mekanik
Avdelningen för Dynamik
Chalmers Tekniska Högskola

Sammanfatnning

Till de detekterbara ljud och vibrationer i ett förarutrymme hör de som inducerats av A/C-systemet. Allt eftersom motorer blir mer effektiva och hybridmotorer allt vanligare leder det till att denna källa är mer framträdande. Den aktuella studien kommer att undersöka en oönskad ressonans med ursprung från A/C:n som detekterats i coupén. Tidigare studier har visat att detta oljud är korrelerat till uppmätta vibrationer vid änden av A/C-systemet. Baserat på dessa studier har en del av systemet mellan kompressor och TXV undersökts i detta projekt. Syftet med studien var att numeriskt simulera det strukturella vibrationsbeteendet i systemet. För att göra detta byggdes en FE modell som validerades och kalibrerades mot resultat från en fysisk modalanalys på systemet. Proceduren för testningen med de metoder som använts i både testplanering och efteranalys av mätningen presenteras. FE-modelleringen gjordes i ANSA och en egenmodsanalys genomfördes i Nastran. Bearbetning av testresultat från testning gjordes i Matlab. Validering av resultat från numerisk simulering och experimentell modalanalys utfördes och den procentuella avvikelsen mellan motsvarande egenfrekvenser beräknades. En parametrisering av FE modellen gjordes där material- och komponentegenskaper parametriserades. Modellen kalibrerades med hjälp av FEMcali i Matlab för att bättre korrelera med de experimentella testdata. Resultaten visade att den numeriska modellen kunde förutse egenmoder med bättre noggrannhet efter kalibreringen. Numeriska resultat hade en genomsnittlig avvikelse mot test på mindre än tio procent, för egenmoder under 500 Hz.

Slutligen genomfördes en undersökning av hur två olika sätt att ytterligare fästa strukturen i chassit hade en inverkan på egenmoderna. Metodiken att korrelera numeriska resultat med fysiska tester introducerar felkällor under valideringsprocessen och innebörden av att utföra arbetet i en iterativ process dras som slutsatser.

Nyckelord: Nastran, Egenmodsanalys, ANSA, FE modellering, LMS, Experimentell modal analys, Modal Assurance Criterion, FEMcali.

Contents

Preface

This master thesis project has been conducted during September 2016 to February 2017 at Alten AB and Volvo Cars. It has been supervised by

- Prof. Thomas Abrahamsson, Department of Applied Mechanics, Chalmers University of Technology
- PhD Reza Okhovat, Department of Analytics, Alten AB

First of all, we want to express our gratitude to Prof. Thomas Abrahamsson for all guidance and knowledge he has contributed with during this project.

We want to thank all the staff at Volvo Cars for their support and encouragement during the project.

Specifically we would like to thank Parinaz Sedarati who has supported and helped guiding us during the testing procedure.

Also, thanks to Patrik Johansson who has provided a good insight in the NVH area and helped us understand the background of the problem.

A special thanks to Milton Milton Peña at BETA CAE Nordic who acted as support for the FE modelling and always was eagar to answer ourquestions.

Thank you also Sören Eriksson, for giving us the chance to perform this project and sorting out all practicalities.

Finally, we would like to thank our families for the immense support you have given us during this project and in our lives.

Mårten Agnesson and Niklas Karlsson

Gothenburg February 2017

List of Figures

List of Tables

1

Introduction

1.1 Background

In the modern automotive industry car engines become more and more efficient at the same time as hybrid and fully electrical cars become common. Therefore a change in car dynamics is a fact. Large and loud engines are not any longer desired in today's cars. Less noise and vibrations are emitted from the engine room which makes the driver comfort, in terms of coupe environment, more sensitive to other sources of noise and vibration. This is an important aspect to cope with in the competitive market among premium range car developers.

Volvo Cars Group (VCG) is a Swedish car manufacturer with its headquarter in Gothenburg. Since 1927 VCG has designed and manufactured automobiles with high standards. With a new line of cars VCG has taken a step into the premium segment and need to seriously address the problem. Safety and comfort have been key factors behind their success during these decades which have set high demands on the product. Large resources within research and development are spent on passenger comfort.

Due to less sources and lower overall noise inside the coupe other sounds/noise become more prominent. One of these sources that transmits vibrations into the coupe is the air condition (A/C) system. From past investigations consisting of experimental testings a tonal dissonance is detected in the interior sound spectra origin from the A/C system. From subjective ratings of the interior sound environment this dissonance is interfering with the engine induced sounds in an unfavorable manner. Serious work of physical testing and experiments has been conducted in order to find the origin of the dissonance.

However, the solution has not been transferable between different car models. The complexity of the system in terms of interaction of structural and fluid dynamics make the problem hard to analyze by experiments. Also the fact that sounds/noise is of subjective character adds to the request to find reliable and repeatable data when analyzing the problem. Not yet has any numerical model of the system been created and used to numerically simulate the case.

1.2 Structural Vibrations in an Automotive A/C System

An automotive A/C system operates to change the condition of the air inside the coupe by regulating the temperature, humidity and the air flow. The refrigeration cycle contains different stages where several thermodynamic processes take place. Both fluid-, thermal- and structural dynamics is to be considered during the process. The A/C that is mounted into a Volvo XC90 can be seen in Figure 1.1. The compressor is belt-driven by the engine and compresses the refrigerant fluid. The fluid is transferred trough, at first, a braided rubber hose (2) and further through one of the two aluminum pipes. At the middle of the aluminum pipe system (4) the two pipes interact with each other in purpose of heat exchange. Refrigerant gas flowing from the compressor to the thermal expansion valve (TXV) (5) is flowing in the pipe covering the one that is bringing cold liquid back to the cooling unit (not present in Figure 1.1) through the low pressure hose (7). A correlation between measured vibrations at the TXV and experienced noise inside the coupe have been found during pre-studies.

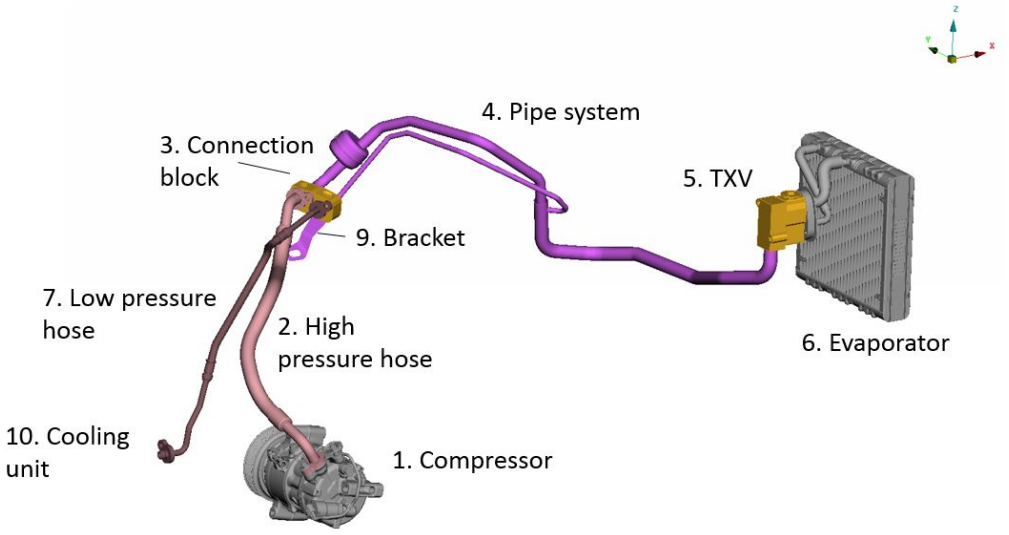


Figure 1.1: The complete A/C system.

1.2.1 Noise

The origin of the problem itself can be described as a undesired sound inside the coupe which is caused by vibrations inside the A/C system. Results from previous work and experiments a tonal dissonance origin from the A/C system is perceived as a noise inside the driver compartment. When discussing noise it is important to notice that it is a subjective measure. The sound transmitted from the A/C system is by automotive standards not to a desired sound in the coupe. The measurements in the coupe have shown an unwanted concurrence between the engine and compressor induced vibrations. Engine induced vibrations are to some extent defined as wanted and the compressor induced are defined as undesired noise.

1.2.2 Compressor orders

Tests and experiments of interior sound levels has shown a relation between compressor orders and high resonances in the coupe. A tonal noise follows the orders of vibration created by the compressor. These orders are related to the mechanism of the compressor and its corresponding forces and moments. Unbalanced forces corresponding to the movements of the pistons and the geometrical arrangement of them relates to the vibrating frequency the compressor emits [1]. In this case where a six cylinder compressor is used the result is 6 strokes per revolution, thus of order 6. Every order is related to the compressor's shaft rate of revolution. Also, from theory[1] the frequency at which rate of revolution the order six is occurring can be calculated by Equation (1.1).

$$\frac{RPM\ rev}{minute} \times \frac{1\ minute}{60\ seconds} \times 1.54 \times 6 = 0.154\,[Hz] \tag{1.1}$$

Equation 1.1 calculates the frequency from rotating the compressor at different revs per minute (RPM) on the engine. A gearing factor of 1.54 is introduced in the equation and is multiplied to the engine rate of revolution in order to get the rotating speed of the compressor. Different rotating frequencies at different RPM's is presented in Table 1.1.

Engine RPM	**Order 6 (Hz)**
1000	154
1500	231
2000	308
2500	385
3000	462

Table 1.1: Engine RPM and corresponding frequency order.

1.2.3 Transfer path

Given that the compressor and evaporator are components in the A/C system not to be modified the parts and structures adjustable are the parts connecting them two together. These are the focus of this study. As mentioned before the compressor should be considered as the input and the evaporator as the output of a transfer path of vibrations. Also, to conclude from the previously mentioned discussion of sounds/noise the problem is that vibrations created by the compressor are transferred through the system into the coupe. An undesirable interaction between the compressor and the pipes and hoses occurs which results in resonance in the system. An ideal case would be a system that does not let any vibrations through to the evaporator, hence no sound/noise from the compressor would be present in the coupe. However, not all frequencies nor magnitudes of vibrations are perceived which permits some vibrations through. At first only frequencies laying in the hearing range need to be considered and secondly only sound of magnitude great enough to be detected by a human. From the experiments where the sound levels in the coupe has been measured it can also be seen that vibrations at certain frequencies

have much higher amplitude than others. Conclusions that have been drawn are that the vibration transfer path either have a positive or negative impact on the vibrations. Hence, resonance occurs in the system. Either the system damps a certain frequency or it forces the system to oscillate with greater amplitude. The ultimate goal is to investigate in what manner the transfer path influences frequencies that origin from the compressor and transfers into the coupe. Resonances created in the pipes needs to be detected and a work method of how to investigate this phenomena will be conducted in this project. Modifying the A/C system and performing a full scale interior sound test is a time consuming and complex work, thus being able to numerically test and analyze the system would be beneficial in many ways. Having a source of validated numerical results will permit a higher complexity of analysis performed on the system. The cost of performing analysis on the system is much lower when performing numerical simulations compared to physical experiments. Also the time required to conduct such a test is far less by numerical analysis.

1.3 Strategy to Investigate a Structural Vibration Behavior

The work of studying a structural dynamics behavior in a complex system is a wide definition consisting of a large range of aspects. As mentioned in the background previous work has been put into by physical testing trying to collect supporting data to point towards problems in different A/C system configurations. This effort has not yet resulted in a method to locate the origin of the problem in a reliable and repeatable manner. The purpose of this thesis is to study the structural vibrations in a particular A/C pipe through experimental and numerical analyses in order to develop a model-based approach to evaluate the transfer path of vibrations. The final goal is to evaluate how further work to reduce noise and vibration induced by the system should be conducted.

In terms of understanding the specific character of the problem an extended approach is to consider the task as an 'Eigenvalue Problem'. This is due to the fact that the solutions are uniquely defined by a set of system configurations such as resonance and damping. A central part in this project is to find a way to numerically simulate the behavior of the structural system. Thereby, an important part during the project will be to perform 'Modal Testing' on the system in order to experimentally map the dynamic behaviour of the system. This method of characterization of structural vibrations through physical testing and measurements is a way to determine mode shapes (eigenvectors) and eigenvalues of the system. If being able to successfully perform modal testing on a physical system the resulting eigenvalues and eigenvectors should function as target values for numerical simulations.

It is assumed that the system is linear while performing modal testing and analyzing the problem by numerical simulations. Thereby, only the parts between the connection block and the TXV is to be considered in this project, see Figure 1.2. The rubber hose does not show a linear behaviour in the pre-studies performed and is therefore discarded in the following analysis, the pre-studies are presented in Appendix A.

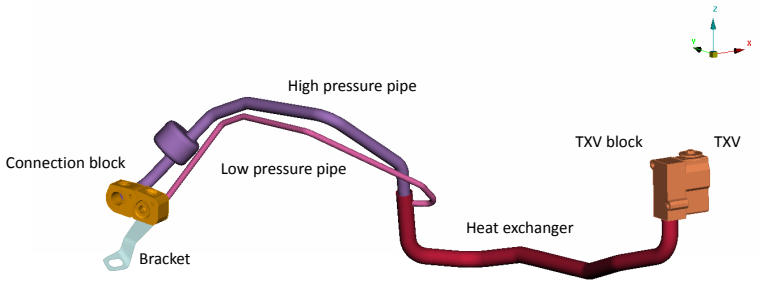
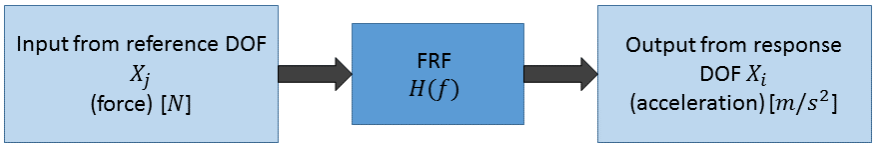


Figure 1.2: Point and section of low pressure pipe attached to chassis.

To ensure that the numerical analysis of the system is reliable and corresponds to the real case a validation of the model should take place. Thereby, the work of modal testing is an essential part in the validation process to ensure that a reliable source of normal frequencies and modes are compared with the numerical results. Hence, a validated FE model will be constructed. Results from these simulations as frequency response functions (FRFs), a quantitative measure of output in response to the defined input, will be analyzed. A schematic visualisation over the FRF concept can be seen in Figure 1.3.



$$H(f) = \frac{X_i}{X_j}$$

Figure 1.3: FRF calculation

The numerical analysis will be performed in MSC Nastran[1] where a Normal Mode Analysis (SOL103) will be performed in order to extract eigenfrequencies and eigenvectors from the FE model. As for the testing the frequency responses will be captured in a vibration test. The response can then be used for calculating resonance frequencies and their corresponding mode shapes. The system will be freely supported meaning that it will be hanging in very flexible strings that give supporting resonance frequencies below 5 Hz.

[1]Nastran is registered trademark from MSC Software

2
Theory

In the following sections governing theory behind analysis methods used in the thesis is introduced with its corresponding equations.

2.1 Normal Modes Analysis

A normal modes analysis is a method to calculate at what frequencies a system can experience resonance. Each eigenfrequency corresponds to a specific eigenmode which describes in what deformation pattern the system will move at that frequency. Eigenfrequencies and eigenmodes are in some structural dynamics theory referred to as "free vibration" because they are calculated without subjected to any load to the structure. The eigenfrequencies and eigenmodes are calculated by obtaining the non-trivial solution for the equation of motion for the system [3].

2.1.1 Equation of motion

The equation of motion can for a damped mechanical system be written as

$$[\mathbf{M}]\{\ddot{\mathbf{q}}(t)\} + [\mathbf{C}]\{\dot{\mathbf{q}}(t)\} + [\mathbf{K}]\{\mathbf{q}(t)\} = \mathbf{F}(t) \tag{2.1}$$

where:

$\mathbf{M} =$ Mass matrix
$\mathbf{C} =$ Damping matrix
$\mathbf{K} =$ Stiffness matrix
$\ddot{\mathbf{q}}(t) =$ Acceleration dependent on time
$\dot{\mathbf{q}}(t) =$ Velocity dependent on time
$\mathbf{q}(t) =$ Displacement dependent on time
$\mathbf{F}(t) =$ Load dependent on time

From the equation of motion the equation for "free vibration" can be written. This is done by setting the load equal to zero.

In MSC Nastran solver 103 (SOL103) solves the equation for performing a normal modes analysis. The analysis is performed only for an undamped system and the equation solved is then

$$[\mathbf{M}]\{\ddot{\mathbf{q}}\} + [\mathbf{K}]\{\mathbf{q}\} = 0 \tag{2.2}$$

As can be seen the equation is not dependent on time since no force is applied, neither the damping of the system. Regarding that the damping is disregarded is important to now since it assumes that the system is completely undamped. In reality all systems have damping but it is commonly assumed that for linear systems is very low.

To calculate the eigenvalues and eigenvectors the assumption of a harmonic solution is introduced [4]:

$$\{x\} = \{\phi\} sin\omega t$$

By inserting this assumption into Equation (2.2) a new equation is obtained as

$$\left([\mathbf{K}] - \omega_i^2 [\mathbf{M}]\right) \{\phi_i\} = 0 \tag{2.3}$$

In the equation the non-trivial solution will give the eigenvalues ω_i^2 and the eigenvectors $\{\phi_i\}$. From the eigenvalues the eigenfrequencies is calculated according to (in Hz)

$$f_i = \frac{\sqrt{\omega_i^2}}{2\pi} \tag{2.4}$$

2.2 State-Space Modelling

A system identification is a method to build mathematical models of a dynamic system from experimentally measured data, called a state-space model. By using an algorithm, N4SID[2], a state space model for measured frequency response can be developed. The algorithm utilizes both input and output data and the objective of the method is to identify A, B, C and D-matrices in Equation 2.5 and 2.6 [2].

A state space model (SSM) is a mathematical representation of a physical system described by a set of first-order differential equations. A general state space representation of a linear system is defined by the following equations:

$$\dot{\mathbf{x}}(t) = [\mathbf{A}]\,\mathbf{x}(t) + [\mathbf{B}]\,\mathbf{u}(t) + \mathbf{w}(t) \tag{2.5}$$

$$\mathbf{y}(t) = [\mathbf{C}]\,\mathbf{x}(t) + [\mathbf{D}]\,\mathbf{u}(t) + \mathbf{v}(t) \tag{2.6}$$

where:

- $t \in \mathbb{R}$
- $\mathbf{x}(t) \in \mathbb{R}^n$ denotes the state vector
- $\mathbf{u}(t) \in \mathbb{R}^m$ is the input or control
- $\mathbf{y}(t) \in \mathbb{R}^p$ is the output
- $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the dynamic matrix
- $\mathbf{B} \in \mathbb{R}^{n \times m}$ is the input matrix

- $\mathbf{C} \in \mathbb{R}^{p \times n}$ is the output or sensor matrix
- $\mathbf{D} \in \mathbb{R}^{p \times m}$ is the feedthrough matrix

The model is written in the most usual case called *Linear Time-Invariant (LTI* where *A, B, C and D*-matrices is not dependent on time. In the model $\boldsymbol{w}(t)$ and $\boldsymbol{v}(t)$ are sources of noise[2].

2.2.1 Modal assurance criterion

When comparing experimental data and numerical results in a validation process later in this project a modal assurance criterion (MAC), as an indicator, is used. This analysis relies on the relation of the angle between two corresponding eigenvectors from experiment and FEA. If the vectors are co-linear, meaning identical similar, the angle between them should be 0° (or 180°). The MAC is calculated according to Equation (2.7).

$$MAC(i,j) = \frac{\left(\left(\boldsymbol{\rho}_i^X\right)^T \left(\boldsymbol{\rho}_j^A\right)\right)}{\left(|\boldsymbol{\rho}_i^X||\boldsymbol{\rho}_j^A|\right)^2} \tag{2.7}$$

In which:

- $\boldsymbol{\rho}_i^X$ is the i-th experimentally obtained mode shape
- $\boldsymbol{\rho}_j^A$ is the j-th predicted mode shape from FEA

The numerical value from Equation (2.7) is a scalar between zero (0) and one (1). If the input vectors are co-linear the value is 1 which equals fully correlated. If they are orthogonal to each other, meaning fully un-correlated, the MAC number is 0.

3
Methods

The overall methodology used in this project can be divided into five sections; *Finite Element Modelling, Experimental Testing, Validation, Calibration* and *Application of Model*. They are performed in the same sequence as they are introduced in this chapter and are highly dependent on each other.

3.1 Finite Element Modelling

The first part of the project was to build and develop a finite element model (FE model) of the system. The purpose of having a FE model is to numerically simulate, using the finite element method (FEM), the system behaviour and in a repeatable and time effective manner elaborate with different system configurations. The modelling work was performed in ANSA[1], a pre-processing software. As described in *Section 2.1.1* the Nastran solver 103 was used to calculate the eigenfrequencies and eigenvectors and the solution was later post-processed in META[2].

When modelling a FE model one has to carefully consider the intention of the. For dynamic simulations some modeling actions to consider in order to reduce modelling errors can be found in MSC Nastran's Dynamic Analysis Manual [4]. These suggestions are:

- Use beams and plates instead of solids.
- Use RBE2 and RBE3 where they will simplify the model.
- Simplify modellings offsets and local modelling details.
- Ignore minor discontinuities such as small holes and fillets.

These are just some suggestions, please refer to the manual for a complete overview.

3.1.1 Geometry

The geometry of the system was provided by VCG. The CAD geometries were given in IGES files (.igs) and the parts included were; connector block, steel bracket, pipe system, TXV block and the TXV, see Figure 1.2. In order to create a FE model where these parts are merged together the geometry was modified. Due to the complexity of the system with small detailed parts and features the geometry was also simplified to some extent. The goal of the geometry modification work was to

[1]ANSA is registered trademark from BETA CAE Systems
[2]META is registered trademark from BETA CAE Systems

achieve a model manageable to apply a mesh to without considerably changing the structure. In order to build an efficient FE model a well defined foundation (the geometry) of the system had to be created. Evaluating the geometry need to be done considering both the real mechanical system and the theory and practice of FEM.

If the CAD geometry provided is a replica of the real system every change will affect the mechanical behavior of the system. More specifically an increase of mass of the system will give lower eigenfrequencies and higher stiffness in the system will increase the eigenfrequencies. Thus, every change need to be carefully considered in order to achieve a FE model that fulfills the objectives to simulate the dynamic behavior. Also important to consider is the purpose of the work, furthermore the simulations that are intended to be conducted. As previously stated the simulation intention in this project is to investigate the structural dynamics of the system and to perform an eigenvalue analysis. The system will not be subjected to any loads were the mechanical response should be evaluated. Hence, the volume and weight of the structure plays a more significant role than a very precise specification of its geometry.

3.1.2 Materials

The structure mainly consists of parts made by aluminum. The pipes, connections blocks and the TXV are all of aluminum. The only part not made by aluminum is the bracket which is made of steel. The material properties used in the FE model can be seen in Table 3.1.

Material	**E [MPa]**	**v [−]**	**Density [ton/mm^3]**
Alu EN AW 3103	69 500	0.3365	2.73E-09
Alu EN AW 6082	70 000	0.3462	2.71E-09
Steel N10130	210 000	0.3	7.85E-09

Table 3.1: Mechanical properties of the materials.

3.1.3 Short evaluation of different FE elements

To see how different FE elements affected the dynamical behavior of a structure by a normal modes analysis a simple study was performed. A simple geometry describing a large plate with thickness 1 mm was created and different types of elements was applied to it. The elements 1st and 2nd order Tria, Quad, Tetra and Hexa elements were evaluated. The model was analyzed through a normal modes analysis in Nastran and the resulting eigenfrequencies was extracted, see Table 3.2. The eigenfrequencies are given in Hertz (Hz) and the first 6 eigenfrequencies were rigid body motion (rbm) and is therefore not presented.

Mode	Quad 1st	Quad 2nd	Tria 1st	Tria 2nd	Tetra 1st	Tetra 2nd
1-6	*rbm*	*rbm*	*rbm*	*rbm*	*rbm*	*rbm*
7	820	820	821	838	1675	846
8	1206	1213	1208	1217	2184	1266
9	1493	1500	1495	1504	2587	1546
10	2096	2107	2105	2105	3858	2229

Table 3.2: Eigenfrequencies for different meshing elements on a simple structure.

In the study it was shown that 1st order tetras produced much higher eigenvalues compared to the other mesh elements which is due to their lack of some degrees-of-freedom. A 1st order solid tetra element has four (4) nodes where a 2nd order tetra element has ten (10) nodes. This results in a too stiff solution for 1st order elements. It was decided not to use 1st order tetra elements but instead only use 2nd order elements.

3.1.4 Meshing of the structure

Also, the geometry needs to be evaluated to the intended analysis before applying a mesh. A fairly good estimation of how the structure will behave is sufficient to obtain prior to applying a mesh of finite elements. Different parts in the geometry have certain characteristics that need to be addressed with specific meshing types and techniques. Regarding what type of elements that should be used where in the geometry the structure can be in a first step divided into two main groups, solid and thin-walled parts. Connection blocks and the TXV are parts defined as solid while the pipes and the bracket are defined as thin parts. Solid elements were applied to the solid parts and shell elements to the thin parts. Dynamic analysis with solids will produce stiffer results for thin structures, especially for 1st order elements. By using 2nd order solids more degrees of freedom are introduced, however the computational time will increase.

The mesh was created using the *batch mesh* tool in ANSA. The parameters used for creating the mesh can be seen in Table 3.3 and 3.4. For holes special requirements were set on number of nodes to obtain smooth surfaces.

Table 3.3: Shell mesh parameters.

Shell elements	Value
Meshing method	*General*
Elements	*Quad*
Order	*1st order*
Target length	2.5 mm
Minimum target length	2 mm
Maximum target length	6 mm
Distortion distance	20 %
Distortion angle	0

Table 3.4: Solid mesh parameters.

Solid elements	Value
Meshing method	*Tetra Rapid*
Order	*2nd order*
Maximum growth rate	1.2
NASTRAN max. apsect ratio	4

3.1.4.1 Shell elements

In the geometry files provided by VCG the pipes were constructed with inner and outer surfaces, forming a volume geometry. Meshing these with solid elements would result in very small elements if the aspect ratio critera should not be violated. Instead, the middle surface between the inner and outer surfaces was created using the MID. SURFACE function in ANSA. By measuring the thickness of the pipe and creating the ”mid surface”, shells could be used instead of solids to represent the pipe.

3.1.4.2 Solid elements

Solid blocks are not well represented with shell elements, mainly due to the importance to obtain the correct mass. For a complex geometry it is difficult and very time consuming to model the system with hexa elements where each section has to be divided into regions to control the creation of the elements. Tetra elements were chosen due to the benefit that they are easily defined. The main drawback is when these are introduced to represent thin surfaces. To reduce the drawbacks of using 1st order elements, 2nd order tetras were used which increases the degrees-of-freedom from four (4) to ten (10), for each element.

When implementing the 2nd order tetra elements at a surface with relatively strong curvature compared to the elements size the elements experienced a ”crack” in the element, due to the extra node. Nastran is not able to account for this and the analysis was unsuccessful when using such elements. Instead the solids had to be created with 1st order tetras and later converted to 2nd order. The result from this can be seen in Figure 3.2. The solids elements with a crack, which aren't supported in Nastran, can be seen in Figure 3.1.

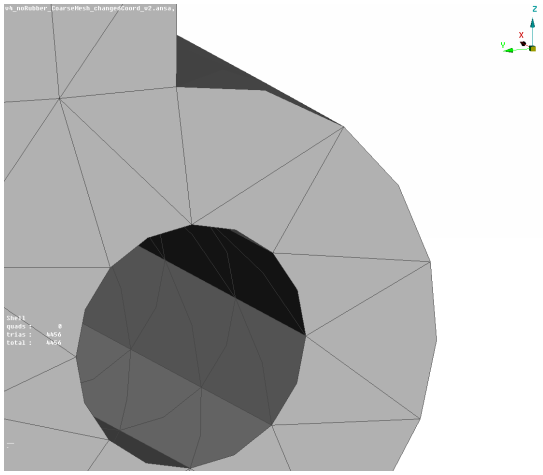


Figure 3.1: 2nd order solid elements with crack.

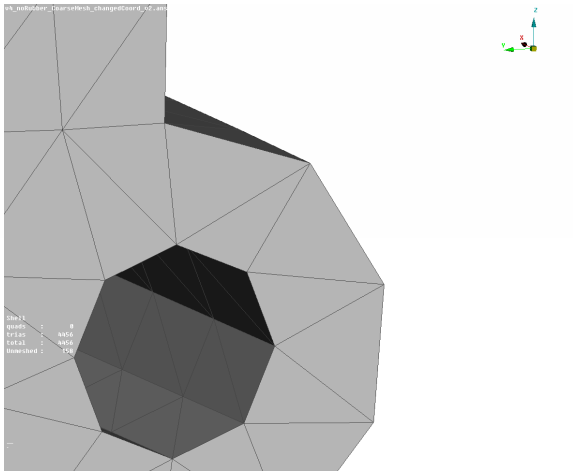


Figure 3.2: 2nd order solid elements without crack.

3.1.4.3 Mesh quality criterion

A mesh criterion was added to evaluate the mesh quality. The criterion was obtained from VCG and was modified with the current settings for element lengths. The

criterion can be seen in Table 3.5.

Table 3.5: Quality criterion for mesh.

Criteria	**Shells**	**Solids**
Aspect ratio	*NASTRAN* 5	*NASTRAN* 8
Jacobian	*NASTRAN* 0.7	*NASTRAN* 0.7
Skewness	-	*ABAQUS* 0.1
Min length	1.5 mm	1.5 mm
Max length	6.5 mm	6.5 mm
Min angle quads	*IDEAS* 45°	-
Max angle quads	*IDEAS* 135°	-
Min angle trias	*IDEAS* 20°	-
Max angle trias	*IDEAS* 120°	-
Min angle quads	*IDEAS* 45°	-
Mid point deviation %	33.3	33.3
Mid point alignment %	33.3	33.3
Negative volume	-	*PARTIAL*
Inclomplete element	-	*Checked*

3.1.5 Connections

When assembling the parts in the FE model their connections and constraints should be applied correctly in order to mimic the mechanical behavior of the system. This was done in ANSA and when applying connections the intended simulation was carefully considered. All modelling features requested have to be supported by the solver in Nastran. In total the system contains seven different parts assembled together by different methods.

3.1.5.1 RBE2

An effective method to use when connecting two parts is the use of rigid body elements. One type of a rigid body element used by Nastran is called RBE2 elements. RBE2 elements are applied between parts and connects nodes to each other. The usage of RBE2 elements allows for loads to impart between the parts connected which will simulate a weld rigid joint connection[6]. These connections are used at several locations in the FE model such as between the bracket, low- and high pressure pipe and the connection block and at the beginning of the heat exchanger. When applying the RBE2 with a tolerance value of the distance between the parts the created elements was set and elements were then created orthogonally to the surfaces.

3.1.5.2 Bonded intersections

Another method of connection used in the FE model is the method of intersecting parts. If two or more parts with the same material are connected together and no relative motion is intended to take place the use of ”intersection” is an easy and

simple method to use. When solving for modal analysis all loads and motion are transmitted from one part to the adjacent one, hence it simulates a ”perfect contact”.

3.1.6 Local coordinate system

To be able to easily obtain the normal displacement to the surface local coordinate system's had to be created for nodes. The displacements could then be referred to the local coordinate system instead of the global system. The local coordinate system's were created in ANSA and extracted in the Nastran-file. However, ANSA do not have any way to define a local coordinate system for a set of nodes and therefore this routine had to be scripted manually, see Appendix E. Scripting in ANSA is performed in the programming language Python and can be implemented in the ANSA file to perform actions using existing functions.

The script creates one point in the candidate node chosen and another two in the nearest node on the surface. Next, the script created a node perpendicular to the x-axis and the y-axis. The nodes obtained could then be used to create a local coordinate system, with the z-axis in normal to the surface. This script was looped for all nodes in a set and in that way created one local coordinate system for every measurement candidate node.

3.2 Experimental Testing

In order to validate a FE model with the purpose of use of simulation a structural behaviour it is important to have measured data from the system to compare with. This is done by experimental testing where the system behaviour is measured during excitation. Tests are performed when the system is set in motion by a shaker and the response is measured by several accelerometers placed around the system. The procedure of the testing is described in this section by in detail present the methods and equipment used.

3.2.1 Pretest planning

To make a meaningful validation of a FE model it is a necessity to have obtained test data of good quality. This is an important aspect in the work of validating a FE model and the test results have to be trusted as a representation of the system's behavior. To maintain the desired quality of the results a pretest procedure was carried out. This procedure had the purpose to evaluate the planned test setup in order to enable that the results could be made trustworthy and useful in further analysis. To make sure the information extracted are reliable it is important to investigate how to measure the response by sensors. In this case accelerometers were used. Since the occurrence of resonance will be present in the system at some frequencies an important aspect is not to place accelerometers close to nodal lines of the corresponding vibrational modes. If that is the case sensors would not measure any response contribution from that particular eigenmode while the surrounding structure is in motion.

3.2.1.1 Method of effective independence

One method that is used for deciding where to place the accelerometers is called Method of Effective Independence (EFI). A Matlab[3] implementation of the *efi*-method can be found in Appendix D.2.

Before executing this method a FE model that can be assumed to fairly well predict the eigenmodes of the system is needed. Since the purpose of the testing is to find the system's normal modes the outcome from the FE model prior to this action is to map the eigenmodes in the system. Successfully mapping the eigenmodes in the system could be utilized when searching for optimal sensor placement using the method of EFI. The EFI algorithm suggests an accelerometer placement setting. The normal modes from a FEA and the number of available sensors are defined as input to the algorithm. The resulting suggestion of placement can be seen in Figure 3.3.

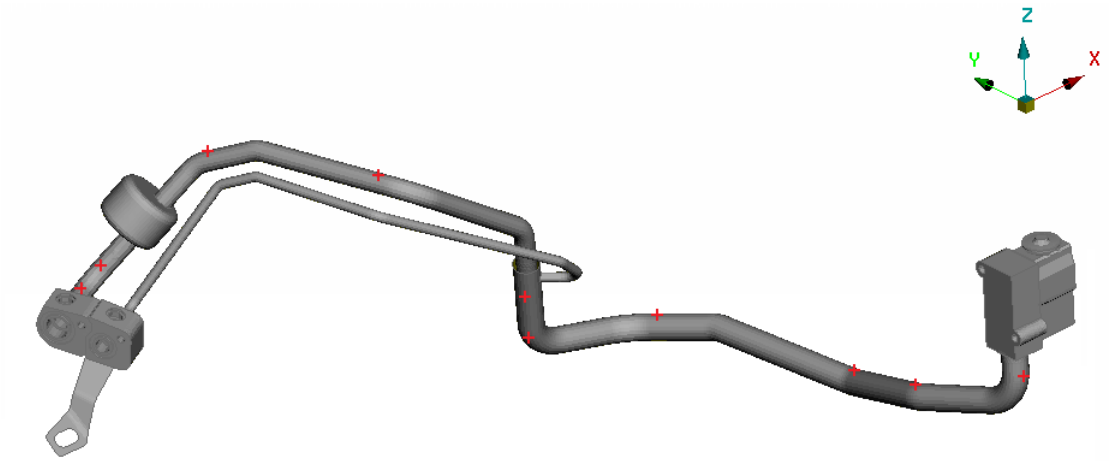


Figure 3.3: Sensors placement marked by red dots.

When using the EFI algorithm candidate nodes for accelerometer placement are chosen. The nodes are taken at the surface of the system and the normal displacement is measured in the nodes. In Equation (3.1) the measured displacement response $\mathbf{y}_s(t)$, or output, in each candidate node is expressed by the partition of the modal matrix $\mathbf{P}_s(t)$. The modal matrix comes from the FE model and is associated with the location and direction of the candidate nodes.

$$\mathbf{y}_s = \mathbf{P}_s\mathbf{z}(t) + \mathbf{v}(t) \tag{3.1}$$

In the equation $\mathbf{v}(t)$ represent the measurement noise and it is assumed that the system do not have any throughput term $\mathbf{Du}$ from the stimuli $\mathbf{u}$ to the output $\mathbf{y}_s$. If the measurement noise is assumed to be stationary, a good sensor placement is obtained when the covariance matrix of the estimation error $\mathbf{R} \equiv E[(\boldsymbol{z}-\hat{\boldsymbol{z}})(\boldsymbol{z}-\hat{\boldsymbol{z}})^T]$ is minimized. Here $\mathbf{z}$ is the system state and the estimate which should be close to $\hat{\mathbf{z}}$ [2].

[3]Matlab is a registered trademark from MathWorks

3.2.2 Test setup

The purpose of the experimental testing as previously described is to obtain the natural modes of the system. The aim was to obtain a test setup as simple as possible. Any types of constrains of the system adds up to sources of errors when trying to capture the physical response. Thereby, the solution used was to have the structure freely suspended in flexible threads. These threads allowed the system to swing in frequencies far below the intended frequency spectrum applied by the shaker. This ensured that the suspension had little influence on the response. The test setup can be seen in Figure 3.4. The shaker was attached to the connection block in the direction of where hoses are attached.

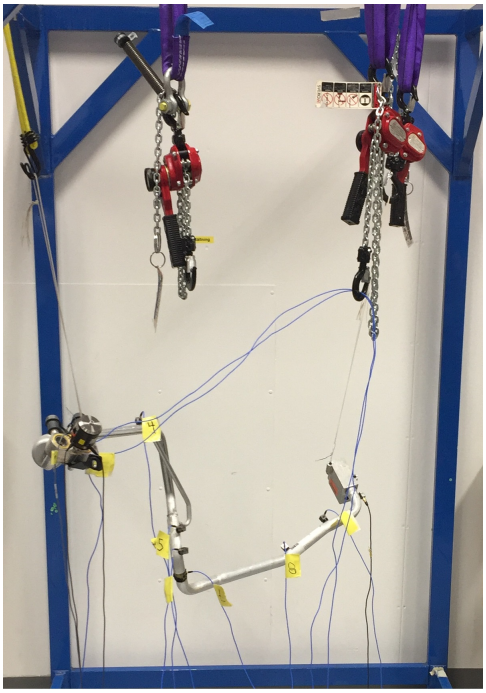


Figure 3.4: Test setup with accelerometers and shaker attached.

3.2.2.1 Equipment

Hardware and software used during the testing were provided by VCG. The fundamental piece of equipment for vibration testing is the data acquisition system (DAQ) which is the device that sample the signals from the sensors. These signals are processed to digital form and can later be further processed in computer software. As for both the DAQ and the associated software used in the testing a LMS[4] system, developed by Siemens, was used. This system is proven to be a suitable choice of testing equipment within the field of structure dynamics and NVH and is widely used at VCG.

The sensor used was a PCB Model 356A32 accelerometer [10], a 3-channel accelerometer that measures vibration in x-, y- and z-axis separately. Ten accelerometers were

[4]LMS is a registered trademark from Siemens

used which thus gave 30 response channels. Their frequency range is 1.0 to 4000 Hz with a sensitivity of 10.2 mV/(m/s²) (±10%).

All accelerometers used had previously been calibrated by the Technical Research Institute of Sweden (SP) and the sensitivity of each accelerometer was noted on the corresponding package. The sensitivity was defined in the test settings in LMS for each accelerometer. However, to make sure an accelerometer was working accurately they were all checked through a calibration process before attachment. This work was done in the LMS Test.lab software and a calibration shaker with a constant frequency (159.2Hz) was used. The sensor was assumed to be in working condition if the measured value from the calibration did not differ more than (±5%) from the calibration sensitivity stated by SP. Hence, the sensitivity specified by SP was used when defining the sensors in the LMS software.

A vibration shaker was used as excitation source in the testing, a LMS Qsources integral shaker from Siemens [9]. The shaker has a frequency range of 20-2000 Hz for a random signal.

3.2.3 Test settings

Several setting options in LMS need to be investigated before performing the tests. It is important to evaluate the settings before the data collection, otherwise severe mistakes can be made and the test data may contain errors. In the coming sections some important settings such as resolution, windowing, averaging and overlapping are described briefly. For a complete overview please refer to the LMS Test.lab manual [13].

3.2.3.1 System excitation

In the experimental testing the system intended to be analyzed is excited by a force in purpose to bring the system in motion. Different kinds of excitation techniques can be used to set a system in motion depending on the structural configuration and intended purpose. In a experimental vibration test procedure the excitation method is a central part. The way a system is set in motion is highly influential on the measured response. The desired motion or response need to be useful and contain the right information needed when analyzing the system. In this project the signal used during the testing was a random signal called 'White Noise'. For that signal a frequency range is set which is then distributed randomly over the signal cycle. The frequency range for the excitation was set to 20-1000 Hz.

Another excitation signal possible to use is called 'Period Chirp' that can be used if the source has difficulty to get energy into the system. When using 'Period Chirp' the signal is continuously ramped up from the lowest to the highest frequency. At low frequencies this creates a very powerful movement on the shaker and because it was only glued on to a flat surface it was concerned that the shaker would fall of by using this method. By using LMS the user also have the possibility to use, closely related to 'White Noise', a signal called 'Pink Noise'. This signal is similar to 'White Noise' but with the difference that each period contains the same amount of energy.

It is important that enough energy is inserted into the system such that it will trigger resonances at all frequencies of interest. The energy is influenced by the voltage of the signal to the shaker. To evaluate that enough energy was inputted into the system three different settings on voltage were evaluated:

- Voltage setting 1: 0.5 V
- Voltage setting 2: 1 V
- Voltage setting 3: 2 V

3.2.3.2 Signal processing

In this section some of the basics within signal processing are explained and the used settings and their affects on the frequency response are discussed. Some settings are coupled such as changing one will affect another, while some settings can be set individually.

When performing a test the user can set a wide range of different settings that will affect the way that the signal is processed. As discussed in *Section. 3.2.3.1* the signal used for exciting the system was a random signal, in the LMS system referred to as 'White noise'. Another setting for the user to determine was the frequency resolution of the response. The resolution will determine for how many frequencies the response will be evaluated. Two different frequency resolutions were tested in order to determine the optimal choices for the test.

- Resolution 1: 0.125 Hz
- Resolution 2: 0.0625 Hz

Continuing this chapter the frequency resolution, f_r, of 0.125 Hz will be used to describe the other settings.

As mentioned the exciting signal was set to 20-1000 Hz and the sampled frequency range for the response was 0-1024 Hz. From these settings the number of frequencies where the response is evaluated, named number of lines, could be calculated as

$$\text{Number of lines} = \frac{f_{Ny}}{f_r} = \frac{1024}{0.125} = 8192 \text{ lines}$$

It is also known that the "maximal useful frequency is $f_{Ny} = f_s/2$"[8] where f_{Ny} represents the Nyquist frequency. Since the maximal frequency sampled is 1024 Hz the sampling frequency can be calculated as

$$f_s = 2 * f_{Ny} = 2 * 1024 = 2048 \text{ Hz}$$

Knowing the sampling frequency,f_s, and the frequency resolution, f_r, the number of samples could be calculated as [8]

$$N = \frac{f_s}{f_r} = \frac{2048}{0.125} = 16384 \text{ samples}$$

. This corresponds well with the theory that at least two points are needed to sample a sinus curve[8]. Since the number of frequency values obtained was 8192 values the number of sample points need to be the double, 16384.

3.2.3.3 Averaging

In summary from *Section. 3.2.3.2* the sampling is performed on the response represented in the time domain producing 16384 samples in total. For these samples a Fast Fourier Transform (FFT) is applied which reveals the frequency content in the signal. Averaging is used to create smooth responses. It divides the time domain signal into blocks where the FFT is performed on each individual block and the frequency content of each block is then averaged together for a total frequency response. The block size, also called window size, is measured in seconds and can be calculated as

$$T = \frac{N}{f_S} = \frac{16384}{2048} = 8\,\mathrm{s}$$

In the project two different settings for averaging was evaluated, the number of averages evaluated was 15 and 30.

The averaging was of the type *Linear averaging* calculated as

$$\overline{A}_{linear} = \frac{1}{N}\sum_{i=1}^{N} a_i \tag{3.2}$$

3.2.3.4 Windowing and overlapping

When averaging is applied another problem occurs which is interfering with the assumption of the FFT. Namely, the FFT assumed that the time domain response is periodic and repeats itself infinitely. This is not the case when using several blocks where the signal has a frequency content distributed randomly across the whole measurement. To cope with this problem a method called windowing is used. The objective of using a window is to get the signal to look the same at the start and end of each block, otherwise something referred to as 'Spectral leakage' will occur. This is where the windowing is used, see Figure 3.5.

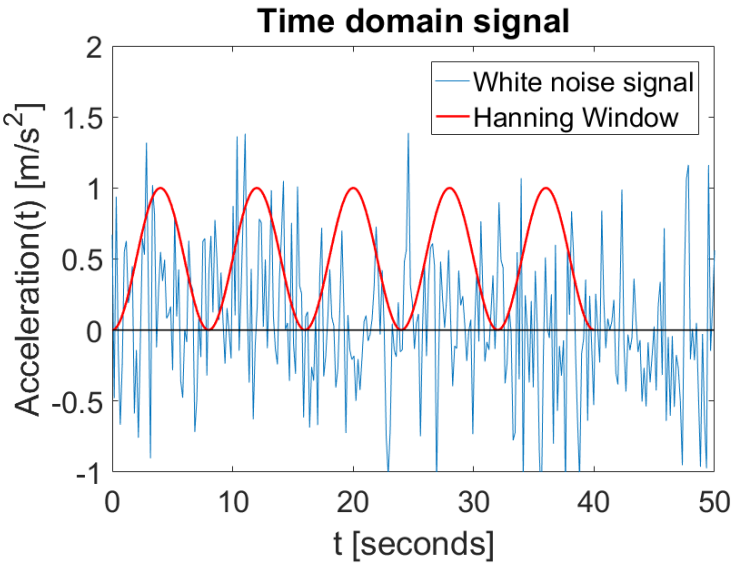


Figure 3.5: Time domain signal with applied window.

In the figure, as it is shown a Hanning window is used in this project which is applied for every measurement block, thus every 8 s. The window is in the form of a function which is zero at the beginning and the end of each block. The function is multiplied with the time domain signal resulting in a zero value at the start and end of the block, hence creating a signal without discontinuities. However, if the block would be of zero value at the start and the end the signal data will be lost at those places. To avoid this a technique called *overlapping* is used. In Figure 3.6 it can be seen that the windows are overlapping each other by 20 %, which was also the case in this project. The overlapping makes sure that the data at the end of one block is captured at the next coming block.

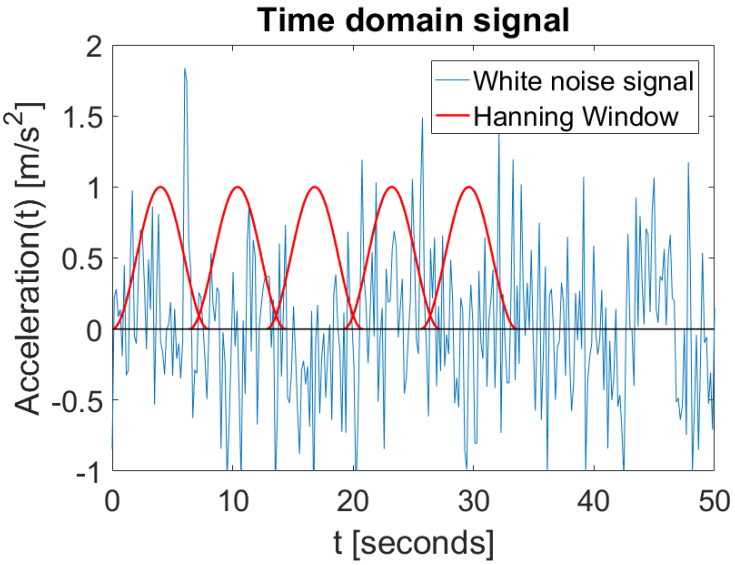


Figure 3.6: Time domain signal with applied window and 20 % overlap.

3.3 Validation

One of the main goal of this project was to build a FE model with the purpose of simulating the dynamic behaviour of the system. As discussed in the previous sections to successfully do so the FEA results need to be trustworthy before trying to simulate the real behaviour. With trustworthy test data a process of creating a mathematical representation of the measurements can be made. That can be used to compare results from the FEA. To characterize the dynamics of the system its frequency response functions (FRF) are utilized. This functions are representation of the relationship between the input and the output measured in the physical tests. Using these functions the behaviour in terms of eigenfrequencies and damping can be extracted in post-processing. The method of how the experimental data was transformed into a mathematical representation is described in this section.

3.3.1 System identification

With the objective to compare experimental data with numerical results a mathematical representation of the experimental data is sought for. The method used in

this project was a system identification method where so called state space models (SSM) of test response data were created. The objective is to identify the A, B, C and D-matrices in Equations 2.5 and 2.6. This work was done using the *ident*-toolbox in Matlab. Furthermore, the system identification method used by ident is the N4SID method. This method uses the experimentally measured FRF data when identifying the SSMs. One SSM representing each FRF corresponding to every sensor was identified. To evaluate the accuracy of the created SSMs each output of the model was plotted and compared over its corresponding measured FRF. A visual inspection to evaluate the fit of the representation was performed on every SSM. In the case of a bad fit the corresponding FRF from the test data was removed and the process was remade. A poor fit is a result of the system identification method not being able to find a good representation of the test data. It could also indicate bad measurements that should be removed. A fundamental quantity of a SSM is the *model order* which can be defined when executing the N4SID method. The model order is related to the number of states of the SSM. The model order is the number of states of the model, hence how many eigenfrequencies and eigenmodes it has.

By introducing:

$$\mathbf{x} = \{\phi\}\mathbf{e}^{(\lambda \mathbf{t})}$$

where:
$\{\phi\}$ = a vector.
λ = a constant.

Equation (2.5) can be re-written into Equation (3.3).

$$[\mathbf{A}]\{\phi\} = \lambda\{\phi\} \tag{3.3}$$

The Equation (3.3) has the non-trivial solution $[\mathbf{A}]\{\phi\}_i - \lambda_i\{\phi\}_i = 0$ and since $[\mathbf{A}]$ is an $n_s \times n_s$ matrix $\{\phi\}_i$ are the eigenvectors and λ_i are the eigenvalues for $(i = 1, 2, ..., n_s)$ for this equation.

3.3.1.1 Creation of state-space model

As previously mentioned the model order corresponding to how many modes a SSM will identify has to be defined prior to the creation of a SSM. An efficient method to use is to divide the frequency spectra into several blocks where each block contains just a few eigenmodes. The advantage of doing so is to be able to define a lower model order based on an anticipation of how many modes that will participate strongly in the frequency block. The problem of having several detected modes in a very small frequency range due to a highly defined model order can thereby be solved. A block was chosen after a visual inspection of the FRFs. After creating a state-space model to each block the blocks were joined by the *parallel*-function in Matlab. As a final step the C and D were re-estimated by a Matlab-function called *ff2cdest.m*, refer to Appendix D.3.

3.3.2 Finite element analysis

To be able to compare numerical results from the finite element analysis (FEA) the same FRFs as given by the test data need to be produced. Thus, output from the

FEA needs to correspond to the ones in the tests. To do this the displacements at the same DOF as the accelerometers measured the response was extracted.

3.3.3 Modal assurance criterion

A modal assurance criterion (MAC) was used when comparing the test results to the FEA results. The MAC was used to determine the similarity of mode shapes captured in the physical testing and from the simulated from the FEA. The resulting mode sets from a normal modes analysis in Nastran are compared to the mode sets from the experimental modal analysis. Each mode is compared to another and the MAC number for every comparison is made, this number is a number of correlation and is between zero (0) and one (1). A MAC number of one means that the two nodes are identical, if a perfect correlation between experimental modes and simulated nodes exist then a matrix with diagonal values of one is obtained.

3.4 Calibration of FE model

The last step in the validation process was to use the experimental results and calibrate the FE model. The purpose of this work is to utilize the results from the system identification method and tune the FE model and gain a better correlation between numerical and experimental results. A Matlab application called FEMcali was used during this process and will be described in this section. The parameters chosen to calibrate are presented in the end of this section.

3.4.1 FEMcali

The FEMcali application uses a created SSM of the experimental frequency responses and a parametrized FE model to match magnitude and phase between the two. The shape of the SSM is the objective and parameterized parameters in the FE model are tuned in order to get a good correlation of the resulting eigenmodes and mode shapes.

The user specifies at what place and in what direction the accelerometers has measured the response to match the correct acceleration in the model to the correct test data. This is done by extracting the degree-of-freedom (DOF) in the x-, y- and z-direction of each grid point where an accelerometer was placed in the FE-model. Also the specific DOF for the actuator is defined in FEMcali. To extract the DOF's from the FE model the Matlab-script *aset2dofno.m* was used, see Appendix D.6. FEMcali is available at Mathworks which includes an user manual that explains the settings in detail [11].

The SSMs used in the calibration were the ones which had the best fit to the experimental FRFs, collected by a visual evaluation. A good fit was decided by how well the SSM can fit to the resonance peaks and anti-resonances in the FRFs.

3.4.2 Parameters to calibrate

The chosen parameters to parametrize in the FE model are given in Table 3.4.2. The parameters are chosen as they are thought to influence the mass and stiffness in the system. Therefore will the calibration process be able to increase and decrease the eigenfrequecnies of the system.

- Young's modulus of Alu EN AW 3103.
- Young's modulus of Alu EN AW 6082.
- Density of Alu EN AW 3103.
- Density of Alu EN AW 6082.
- Low pressure pipe thickness.
- High pressure pipe thickness.
- Heat exchanger pipe thickness.

3.5 Application of FE Model

Finally, a numerical investigation of the system behaviour in its working environment is to be conducted. After the process of validating and calibrating the FE model it will be used to produce numerical results of the dynamic behaviour when mounted inside the engine room. The purpose of this analysis is to collect results of the structural dynamics behaviour of the system and to study different mounting arrangements.

3.5.1 Mounting arrangement

When the system is mounted in the engine room it is attached to the chassis by the steel bracket which is bolted directly to the chassis and by the TXV which is mounted to the HVAC unit. An investigation of how representative this arrangement is and if a better solution should be introduced to change the transfer path of the structural vibrations is done. Specifically a section of the low pressure pipe that is closely aligned by the chassis covering the suspension is evaluated as a possible additional mounting point, see Figure 3.7. There an attachment is possible without interfering with the other structures in the engine room.

Figure 3.7: The system mounted in a test rig.

Two main attachment solutions were modelled in ANSA and analyzed through a normal modes analysis in Nastran. The first solution to analyze is to fix a point of the low pressure pipe to the section of the chassis described previously. The second solution to the test is to constrain a large section of the low pressure pipe to the chassis, see Figure 3.8 and 3.9. All attachments to the chassis are modelled as fully constrained in ANSA, meaning that all DOFs are constrained.

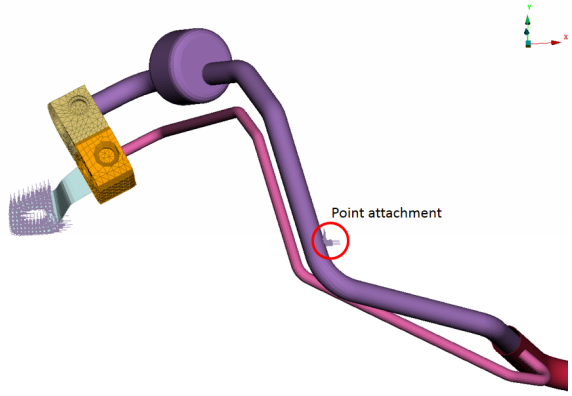


Figure 3.8: Point constraint.

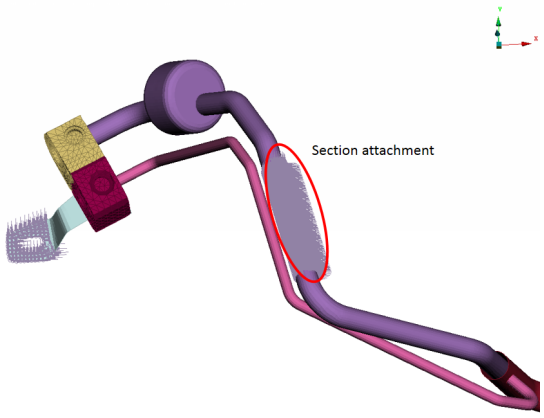


Figure 3.9: Section constraint.

4

Results

4.1 Numerical Results

The resulting FE model has 29,748 shell elements (1 st order Quads) and 14,196 volume elements (2 nd order Tetras). Which elements used where in the model can be seen in Figure 4.1.

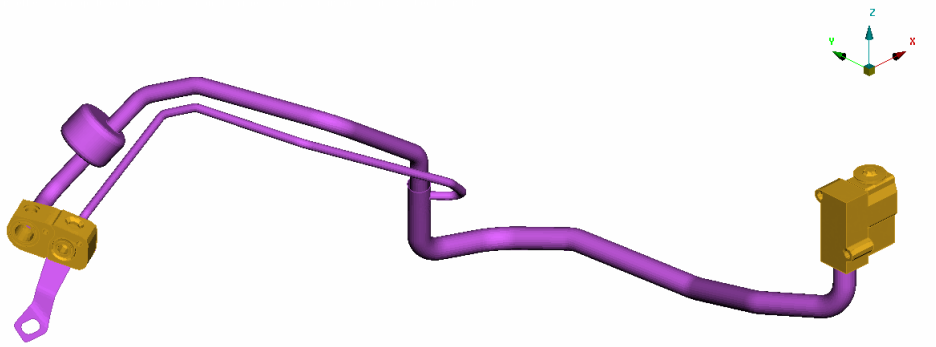


Figure 4.1: FE model, solid elements (yellow) and shell elements (purple).

The resulting eigenfrequencies from a normal modes analysis, SOL103 in Nastran, are presented in this section. An evaluation of the sensor placement is conducted by using the modal assurance criteria (MAC).

4.1.1 Normal modes and eigenfrequencies

From the FE model the non-rigid body eigenfrequencies were extracted, together with the mode shapes. The first 9 eigenfrequencies from the initial model are shown in Table 4.1.

Table 4.1: Values of eigenfrequencies below 200 Hz from the numerical analysis.

Mode	Eigenfrequency [Hz]
1	28.4
2	42.2
3	50.9
4	104.2
5	121.1
6	139.8
7	153.4
8	172.1
9	194.6

4.1.2 Sensor placement check

The sensor placement used in both in FEA and in the test set-up was evaluated by an orthogonality check. To validate whether the number of response collecting points was of good choice a MAC evaluation of the mode shape set from the FE model was evaluated against itself. As seen in Figure 4.2 the results of a MAC evaluation of the mode shapes from FEA shows good correlation, hence only small off-diagonal elements.

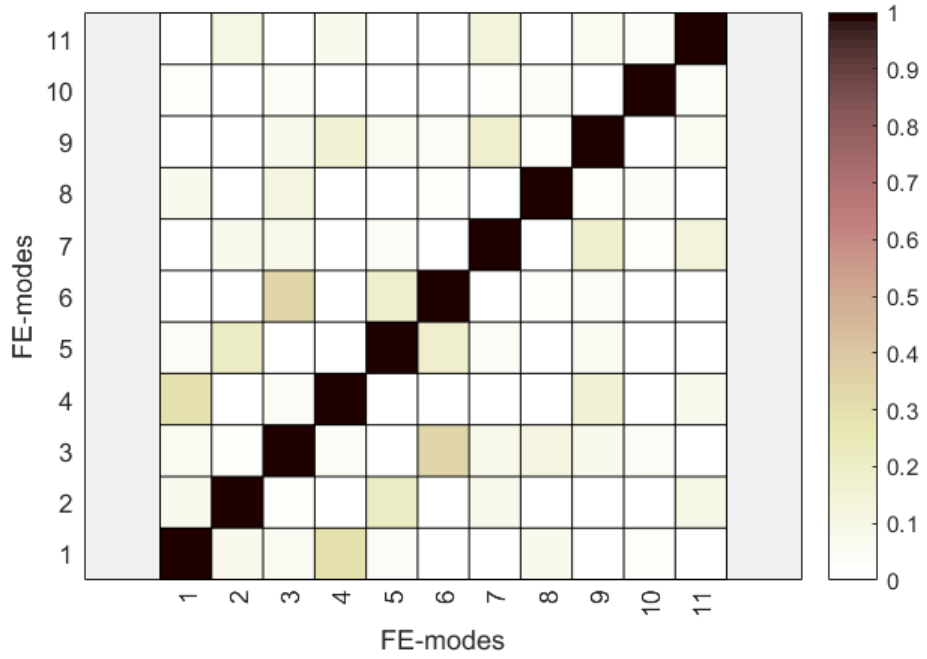


Figure 4.2: Orthogonality check.

The Matlab scripts for creating the MAC plot is found in Appendix D.7 and D.8. To be able to extract the mode shapes from the FE-model the .pch-files (punch files)

are read. To do this a Matlab is script used which can be found in Appendix D.5.

4.2 Experimental Test

From the experimental testing acceleration response as function of frequency for each sensor was extracted. After an individual visual inspection of them a common trend was found, the response in the lower part of the frequency spectra (0-25 Hz) showed an unstable behaviour. In that frequency the response was noisy showing an oscillating behaviour. This was sought to be cause the lowest frequency of excitation was set and limitted to 20 Hz. Since the response showed a non-linear behaviour it was disregarded below 25 Hz.

4.2.1 Linearity

Resulting FRFs from tests executed by altered voltage setting in LMS test lab are used to evaluate the linearity of the system. In Figure 4.3 the sum of all FRFs from a test with the voltage settings of 0.5V, (red), 1 V (green) and 2 V (blue) is plotted together.

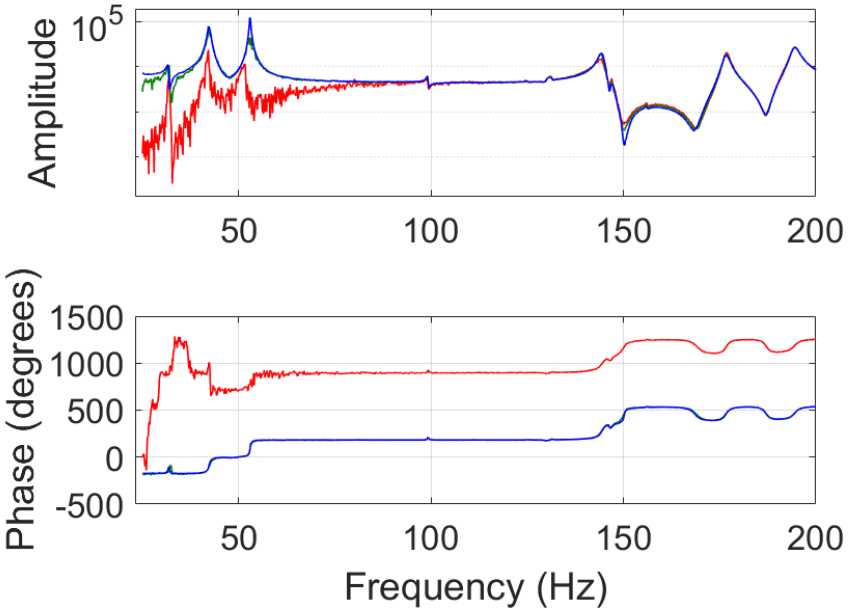


Figure 4.3: Evaluation of linearity.

The result from evaluating different voltage settings showed that below 2 V the response showed a non-linear behaviour. It was decided to use a voltage level of 2 V for the continued testing.

4.2.2 Evaluation of test settings

Different settings was performed for frequency resolution and the number of cycles for which the frequency response was averaged. To evaluate the impact when using different settings the sum of FRFs were evaluated at in a frequency range of 25-200 Hz, see Figure 4.4. In the figure curves represent different settings for averaging and frequency resolution (voltage is 1 V for all responses):

- Blue curve: 15 averages, 0.125 Hz frequency resolution.
- Green curve 30 averages, 0.125 Hz frequency resolution.
- Red curve: 15 averages, 0.0625 Hz frequency resolution.

For visual purposes the magnitude of the response have been multiplied with different scalars to easier evaluate the difference in response.

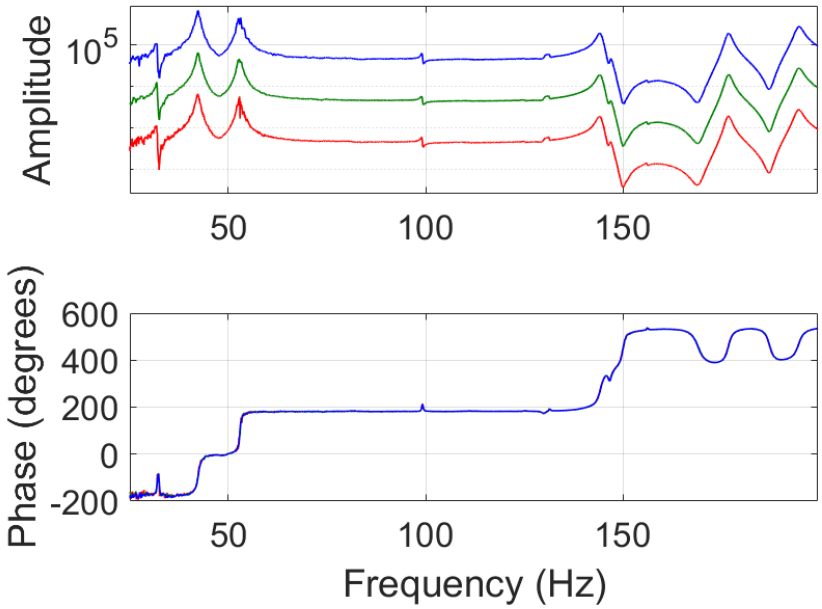


Figure 4.4: Evaluation of number of averages and frequency resolution.

The results from using different averages and frequency resolution showed little difference for the response. The practical difference was that halving the resolution resulted in twice as long response file while doubling the number of averages doubled the measuring time. It was decided to use a frequency resolution of 0.125 Hz to reduce the size of the output file but use 30 averages, resulting in a test time of ≈ 194 s.

4.3 Validation and Calibration

In this section the results from the system identification process where the experimental results were processed are presented. From the system identification the best fit between numerical and experimental results was found for frequencies below 200 Hz, therefore the frequency range considered in this section is in the range 25-200 Hz. These results are compared with the numerical results before and after the calibration process. The resulting calibrated FE model parameters are presented together with a final comparison of numerical and experimental results. The Matlab-script written to do the validation can be found in Appendix D.1.

4.3.1 System identification

The resulting normal modes was first identified by visually examining the sum of all frequency response functions, for all 30 channels (sensors), see Figure 4.5.

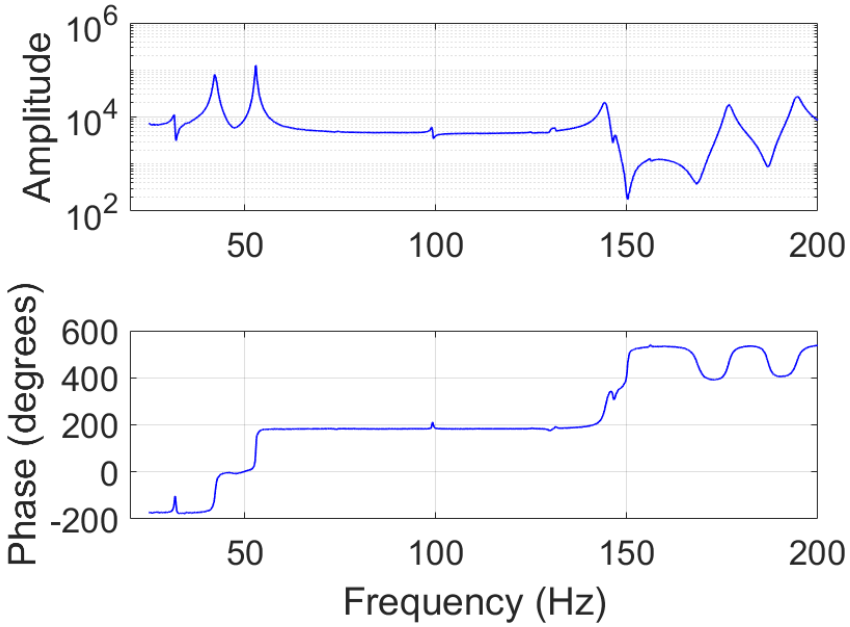


Figure 4.5: Sum of all frequency response functions.

The result from the visual observation was 9 existing eigenfrequencies in the frequency interval of 25-200 Hz. To capture all visually found eigenfrequencies a state-space model was created with the model order of 18, hence capturing 9 eigenfrequencies in the same frequency interval. All frequencies found from the two methods can be found in Table 4.2.

Table 4.2: Eigenfrequencies identified by visual observation and state-space modelling.

Mode	Visual [Hz]	SSM [Hz]
1	∼ 33	31.7
2	∼ 42	42.3
3	∼ 53	53.0
4	∼ 99	99.0
5	∼ 130	130.3
6	∼ 144	144.4
7	∼ 147	146.6
8	∼ 177	176.6
9	∼ 195	194.5

4.3.1.1 State-space models

State-space models (SSM) were created for all channels in the frequency range 25-200 Hz. The 17 models that gave the best fit were saved and would later be used for the calibration. State-space model sensor 1 in the x-direction (u1 -> y1) can be seen in Figure 4.6. The remaining 16 state-space models can be found in Appendix B.

As can be seen in the figure there is a small deviation in the test data at 130 Hz, which also was found when creating the state-space models for all channels (described in *Section. 4.3.1.* In creation of the state-space when using the blockwise method it was set that this peak would be disregarded. This was because the mode shape correlation showed very low correlation for just that mode and other modes because got distorted when they were included. In Figure 4.6 the deviation at 130 Hz between test data and the state-space model is clear. The plot is made using a *Matlab*-script that can be found in Appendix D.4.

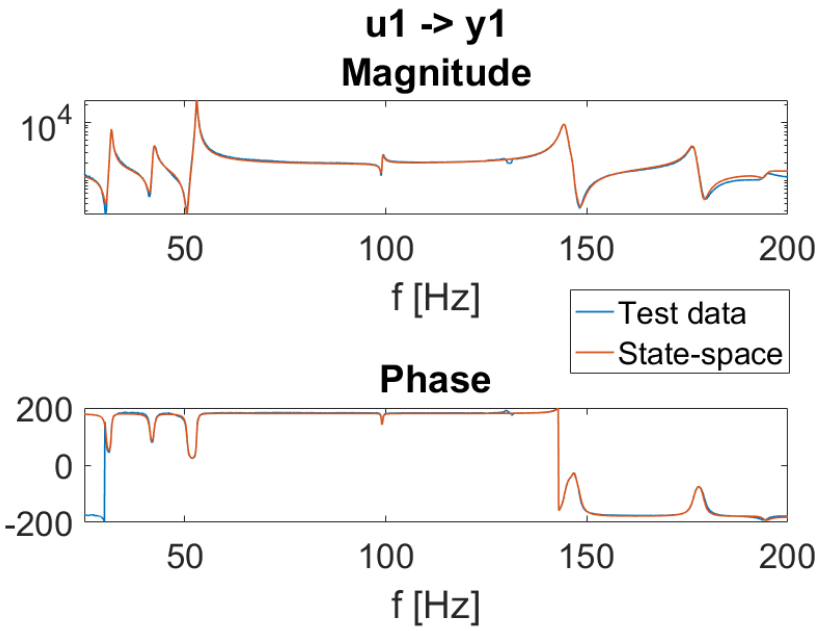


Figure 4.6: Sensor 1 x-direction.

4.3.2 Calibration

Finally, the process of calibrating the FE model to test results was performed. The Matlab application FEMcali was used for this purpose, introduced in *Section. 3.4.*

In the process of tuning FE model parameters the initial nominal parameters presented in *Section. 3.1.2* were used. Six (6) parameterized model parameters together with their corresponding nominal values and upper and lower limits of calibration are given in Table 4.3. All parameters were assumed to have a stochastic spread with an uniform distribution. The frequency range chosen for calibration was set to 25-200 Hz.

Table 4.3: Calibration parameters.

Parameter	**Nominal value**	± %
Young's modulus Alu EN AW 3103	69 500 [MPa]	20
Young's modulus Alu EN AW 6082	70 000 [MPa]	20
Density Alu EN AW 3103	2.73e-9 [ton/mm^3]	20
Density Alu EN AW 6082	2.71e-9 [ton/mm^3]	20
Low pressure pipe thickness	1.25 [mm]	20
High pressure pipe thickness	1.50 [mm]	20
Heat exchanger pipe thickness	1.00 [mm]	20

The other calibration settings used in FEMcali is stated in Table 4.4.

Table 4.4: Calibration settings.

Setting	**Value**
Latin Hybercube sample realization	100
Max number of iterations for each parameter	100
Number of randomized starts for each parameter	5
Equalization damping level	1 %
Number of frequency steps per half-bandwidth	2.0

The resulting eigenfrequencies from the calibration can be seen in Table 4.5. In the table the eigenfrequencies from the state-space model are compared to the numerical results pre- and post calibration, as well as the percental difference.

Table 4.5: Eigenmodes (Hz) of the SSM and FEA pre- and post calibration.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7	Mode 8
SSM	31.7	42.3	52.9	99.1	144.5	146.7	176.6	194.4
PreCal	28.4	42.2	50.9	104.2	121.1	139.8	153.4	172.1
PostCal	31.2	43.2	53.5	106.5	121.8	138.2	158.5	179.9
Diff Pre/SSM	10.48%	0.21%	3.85%	4.89%	16.19%	4.70%	13.14%	11.47%
Diff Post/SSM	1.59%	2.08%	1.11%	6.96%	15.72%	5.83%	10.24%	7.45%

The table shows that the calibrated FE model give a better estimation of all eigenfrequencies except for mode 4 and 6. The sum of deviation percentage points for the first 8 modes has decreased from 65.72 % to 50.98 %.

As described in *Section 3.3.3* the modal assurance criterion was calculated before and after the calibration, see Figure 4.7 and 4.8.

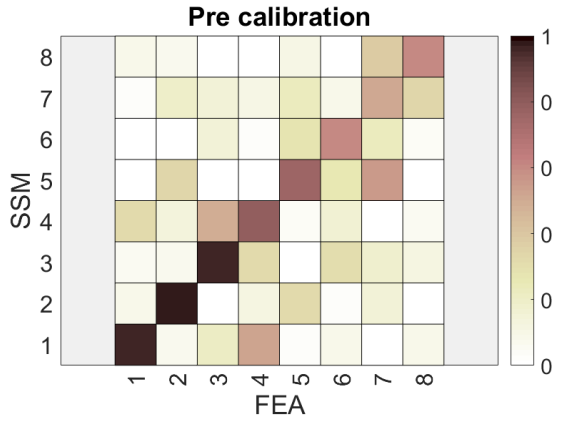


Figure 4.7: MAC before calibration.

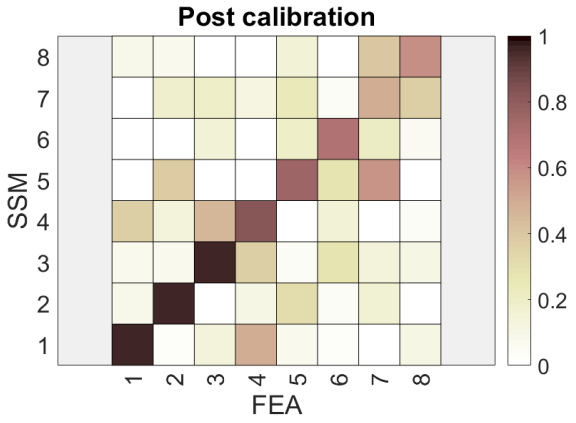


Figure 4.8: MAC after calibration.

Figuratively the MAC-plots shows a slight improvement for the post calibration correlation and as comparison the average of the diagonal elements was calculated. For the first 6 diagonal elements of the MAC the average increased from 0.8380 to 0.8587, before and after the calibraiton.

Finally, in Table 4.6 the calibrated parameters are presented together with percentage difference to the nominal values.

Table 4.6: Nominal and calibrated parameters.

Parameter	**Nominal value**	**Calibrated value**	**Diff** %
1. Young's modulus Alu EN AW 3103	69 500 [MPa]	77 536 [MPa]	11.56 %
2. Young's modulus Alu EN AW 6082	70 000 [MPa]	64 464 [MPa]	-7.91 %
3. Density Alu EN AW 3103	2.73e-9 [ton/mm^3]	3.24e-9 [ton/mm^3]	18.8 %
4. Density Alu EN AW 6082	2.71e-9 [ton/mm^3]	2.46e-9 [ton/mm^3]	-9.23 %
5. Low pressure pipe thickness	1.25 [mm]	1.20 [mm]	-4.00 %
6. High pressure pipe thickness	1.50 [mm]	1.55 [mm]	3.33 %
7. Heat exchanger pipe thickness	1.00 [mm]	1.19 [mm]	19.00 %

The calibration transfer function shows how the response from the nominal and calibrated FE model has tuned to fit the test model, see Figure 4.9 and 4.10. In the figures the frequency range of 25-65 Hz is shown which gave the best correlation between mode shapes.

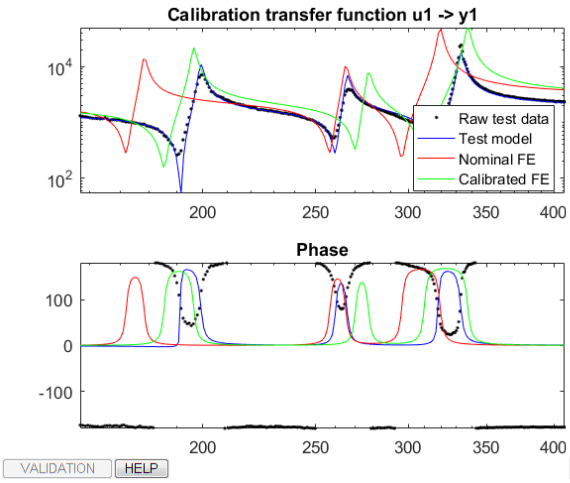


Figure 4.9: Sensor 1 x-direction.

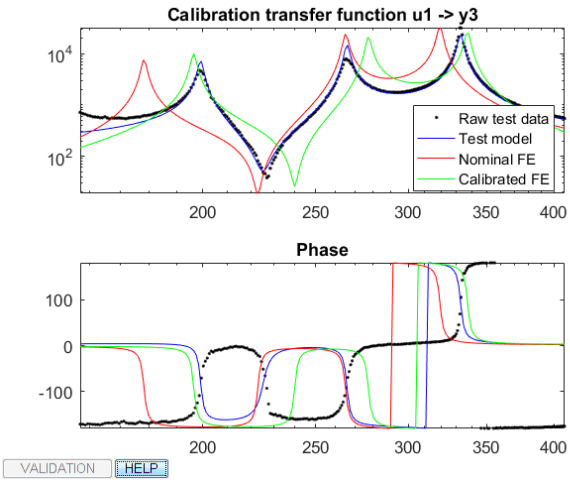


Figure 4.10: Sensor 1 z-direction.

4.3.3 Validation of calibrated model

When comparing the eigenfrequencies with the SSM and both the nominal and calibrated FE model in a wider frequency range (0-500 Hz) it could be noted that the avergae difference (percentual) had decreased from 8.69 % to 7.48 %, see Table 4.7. In the table the 18 (non-rigid body) modes are presented in the frequency interval.

Table 4.7: A comparison in eigenfrequencies (Hz) between nominal and calibrated model.

Mode	SSM	Nominal model	Diff %	Calibrated model	Diff %
1	31.7	28.4	**10.57**	31.2	**1.59**
2	42.3	42.2	**0.26**	43.2	**2.08**
3	52.9	50.9	**3.90**	53.5	**1.11**
4	99.1	104.2	**4.81**	106.5	**6.96**
5	144.5	121.1	**16.21**	121.8	**15.72**
6	146.7	139.8	**4.68**	138.2	**5.83**
7	176.6	153.4	**13.16**	158.5	**10.24**
8	194.4	172.1	**11.52**	179.9	**7.45**
9	221.6	194.6	**12.21**	195.9	**11.58**
10	256.2	217.5	**15.12**	214.4	**16.30**
11	304.5	260.0	**14.60**	265.9	**12.68**
12	308.6	294.0	**4.73**	291.8	**5.44**
13	327.3	298.8	**8.70**	309.4	**5.48**
14	367.5	325.1	**11.54**	332.9	**9.40**
15	392.4	361.6	**7.85**	367.8	**6.26**
16	413.2	378.4	**8.42**	374.3	**9.40**
17	436.8	425.7	**2.54**	435.2	**0.36**
18	443.7	470.2	**5.63**	475.6	**6.71**

4.4 Normal Modes Analysis of Mounted System

Numerical results from simulations of the calibrated and final validated FE model with boundary conditions for the mounted system will be presented in this section. The extracted eigenfrequencies from a normal modes analysis can be seen in Table 4.8. The current mounting arrangement is to have the system attached to the chassis by a bolted connection at the bracket and having the TXV mounted onto the HVAC. The other two mounting configurations are presented in 3.5.1.

Table 4.8: Eigenfrequencies below 500 Hz of system with different mounting arrangements.

Mode	Current [Hz]	Point [Hz]	Section [Hz]
1	32.6	60.7	76.4
2	54.1	93.8	119.2
3	60.4	118.9	141
4	92.9	170.3	212.8
5	118.2	212.3	248.7
6	133.5	251.5	259.3
7	173.1	257	268.3
8	213.1	267.3	298.5
9	263.2	286.7	304.2
10	275.1	315.2	343.5
11	292.9	344.6	372
12	298.9	371.3	418
13	324.1	414.7	463.5
14	371.8	478.6	495.3
15	405.7		
16	452.8		
17	496.1		

5
Conclusion and Discussion

5.1 Conclusions

A validated and calibrated FE model of a part in a Volvo XC90's A/C system have been created. The model simulates the structural dynamics behaviour fairly well in a frequency range of 25-500 Hz. A numerical investigation of different mounting configurations has been conducted by using the model.

The numerical simulations predicts eigenfrequencies up to 500 Hz corresponding to the test results quite accurately. Comparing the mode shapes does not show just as good similarity for frequencies above 100 Hz, however the three first mode shapes from the FEA mimic the test data very well. Poorer numerical results of higher modes indicate poor interface modelling between the parts in the model.

5.1.1 Linearity

From the test results performed with different voltage settings the difference in response amplitude between the three indicates how the input energy correlates to the system response. The lowest voltage setting used was 0.5 V and from Figure 4.3 it is seen that the FRF corresponding to 0.5 V indicates a noisy response for low frequencies. The conclusion from this observation is that not enough energy was inserted into the system. Comparing the graphs corresponding to voltage 1 V and 2 V the response amplitudes are alike, hence the system behaves linearly. At last, 2 V shows the most steady response and was the setting used during the final test.

5.1.2 Evaluation of test settings

The set-up used for the experimental testing can be evaluated in terms of number of averages and resolution. As could be seen in *Section 4.2.2* the behaviour of the frequency response functions did not change substantially.

As for averaging it could be seen in Figure 4.4 that when increasing the number of averages from 15 to 30 the response functions became slightly smoother in the lower frequency regions while at the higher (above 70 Hz) no difference could be seen. It was concluded that using an averaging of 15 was sufficient enough to capture the behaviour at the complete frequency spectra.

The resolution was also evaluated by comparing results of 0.125 Hz steps with 0.0625 Hz steps. It could be seen that at frequency response peaks and at anti-resonances the response was more clearly indicated by having a lower resolution.

However, when it came to evaluation the system's eigenfrequencies, no difference were seen. The conclusion was that a resolution of 0.125 Hz well enough captured the frequency response in the system.

5.1.3 Mounted system

When adding an additional fixation point to the mounting of the system it becomes remarkable more stiff, hence increasing the eigenfrequencies. Comparing the two methods of fixing the system to the chassis it is seen that to use a wider area of fixation the eigenmodes become higher and fewer in a frequency range up 900 Hz.

5.2 Discussion

A preferable starting point of a project as the one presented would be to posses a verified FE model. Modelling and working with a FE model is an iterative process and to achieve a fully functioning model is very time consuming. This was not the case in this project where a model was created from scratch. Due to the time constraints of the project the initial model was used during the pre-test planning and the first validation process. Not having a verified model as initial source of numerical results of the dynamic behaviour could and probably will set the validation and calibration process in the wrong direction.

5.2.1 FE model geometry

During the process of the FE modelling the geometry was simplified to some manner. The main reason to do so was to obtain a geometric description with easy means that was sufficiently accurate to give a proper FE mesh. However the small changes in the geometry has to addressed as a source of errors. The implication of simplifications is not evaluated. One consequence of simplifying the geometry is that the mass of the system could change which would affect the dynamic behaviour.

5.2.2 Sensor placement

A learning outcome from the project is to map the sensor locations in the FE model with caution. When defining and applying local coordinate systems in the FE model the mapping of axis would even with the slightest difference from the accelerometer's have an impact on the resulting mode shapes. This sensitivity has introduced a source of error in the validation process. What should be considered with caution is the difficulty to apply a coordinate system on a circular surface that will correspond to the test setup. If the surface had been flat the sensor location in the FE model compared to test would have smaller impact and the sensitivity should be smaller.

5.2.3 System identification

During the system identification process a visual observation method was utilized to ease the process. The result was that some identified frequencies from the initial state-space model was rejected due to judgement based on plausibility. Hence, no substantial evaluation process was conducted.

5.2.4 Parametrization

The calibration of the model resulted in a change of both eigenfrequencies and mode shapes representing a better fit to the experimental results. However, the calibrated parameters 3 and 7 were changed almost to their limit of ±20% which indicates an insufficient choice of parametrization parameters. An evaluation of the choice of parameters to calibrate the model would be preferable.

5.3 Proposal of Future Work

The work methodology with corresponding results and conclusions presented in this thesis has demonstrated a successful way of obtaining a validated FE model calibrated to experimentally collected data. Future work should be spent on further development and refinements on the FE model with the objective to obtain more correct initial results of the dynamical behaviour. When this work is done the project it self could be extended to also cover the real behaviour during working condition of mounted inside the engine room.

5.3.1 Extension of the system and analysis

As an extension to the problem considered in this project the system could be extended. More components belonging to the whole A/C system mounted in a Volvo XC90 could be conducted in future work. As for example the transfer path between the compressor and TXV should be completed by adding the rubber hose. Also a 'Direct transient response' analysis (solver 109)[5] should be conducted where the time varying compressor load could be defined as input to numerical analysis.

Additionally, to simulate and mimic the real behaviour during working condition the system has to be analyzed with internal pressure and pulsations created by the compressor. Also the acoustics around the pipes that could affect also the structural behaviour should be modelled for.

5.3.1.1 Change of Nastran solver

As stated in *Section 2.1.1* the solution method for extracting natural frequencies (solver 103) do not take damping into account. Nastran offers more alternatives for simulating dynamical behaviour, one is 'Modal Complex Analysis' (solver 111)[5]. In solver 111 it is possible to define damping as a tabular function of natural frequency. However, the natural frequency values will still not consider the damping and the effect is only taken into account in the calculation of the response (displacement, acceleration, velocity). Another possibility is to use a so called 'Direct complex eigenvalues' analysis (solver 107)[5] where structural damping can be used to dampening modes.

5.3.2 Mounting of system

Further analysis and experiments on different mounting solutions of the system should be performed. A survey of existing mounting solutions of other engine components should be performed to study already existing vibration dampening solutions before conducting a wider evaluation.

Bibliography

[1] Taylor, C.F. (1985) *The Internal Combustion Engine in Theory and Practice*. Camebridge: MIT Press.

[2] Abrahamsson, T. (2012) *Calibration and Validation of Structural Dynamics Models*. Göteborg: Chalmers University of Technology.

[3] Craig, R.R., Kurdila, A. (2011) *Fundamentals of Structural Dynamics, 2nd edn.* Hoboken: John Wiley & Sons.

[4] MSC Nastran (2014). MSC Nastran 2014 Dynamics Analysis User's Guide. *http://www.mscsoftware.com/* Jan 21 (2017)

[5] MSC Nastran (2016).MSC Nastran 2016 Quick Reference Guide. *http://www.mscsoftware.com/* Jan 21 (2017)

[6] MSC Nastran (2016) Reference Manual. *http://www.mscsoftware.com/* Jan 21 (2017)

[7] Brandt, A. (2011) *Noise and Vibration Analysis: Signal Analysis and Experimental Procedures*. [Electronic]. Hoboken: John Wiley & Sons.

[8] Heinzel, G., Rüdiger, A., & Schilling, R. (2002). Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new at-top windows. *http://hdl.handle.net/11858/00-001M-0000-0013-557A-5*

[9] Integral Shaker. *http://www.plm.automation.siemens.com/se_se/* (12 Jan. 2017)

[10] Test accelerometers. *http://www.pcb.com* (12 Jan. 2017)

[11] FEMcali application. *https://se.mathworks.com/* (17 Jan. 2017)

[12] LMS Test.Lab Throughput Processing Tips. *http://www.plm.automation.siemens.com/se_se/* (20 Jan. 2017)

[13] LMS Test.Lab Manual. *http://www.plm.automation.siemens.com/se_se/* (20 Jan. 2017)

A

Rubber hose response

Frequency response function for three channels on the rubbe hose subjected to a vibration test. In Figure A.4 a state-space model has been created for one of the frequency responses. The blue curve has model order 36, the green curve has model order 50 and the red curve has model order 58. All curves create a fit lower than 70 %.

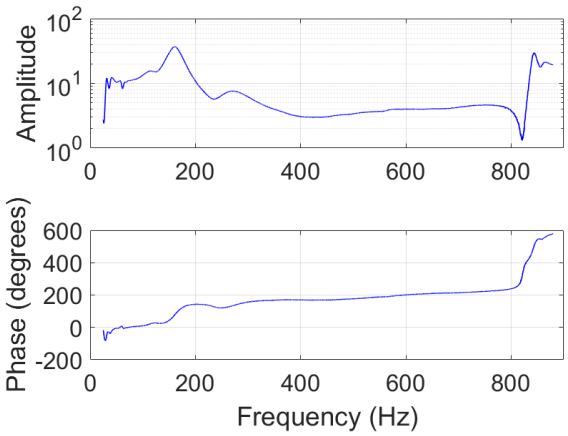


Figure A.1: Sensor 1 x-direction.

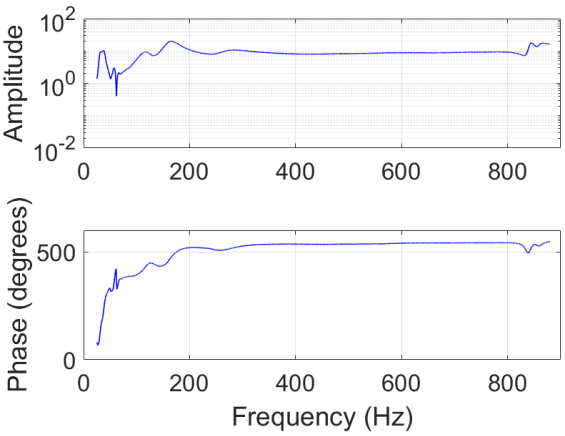


Figure A.2: Sensor 1 y-direction.

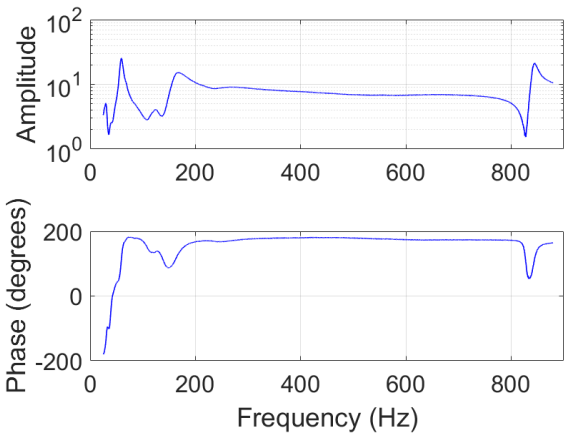


Figure A.3: Sensor 1 z-direction.

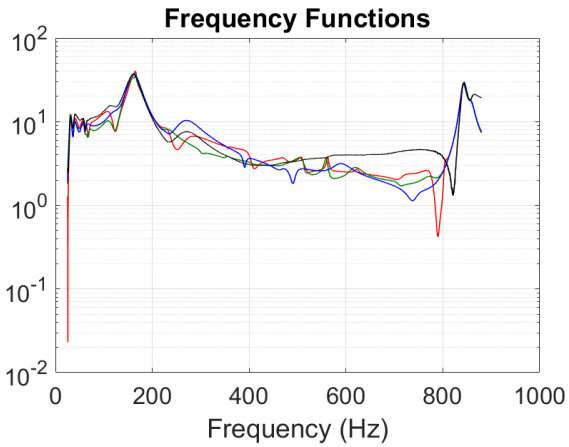


Figure A.4: State-space model for sensor 1 x-direction.

B

State-space models

The 16 best state-space models are shown in this Appendix. They are 17 in total, 1 is given in *Section 3.4*.

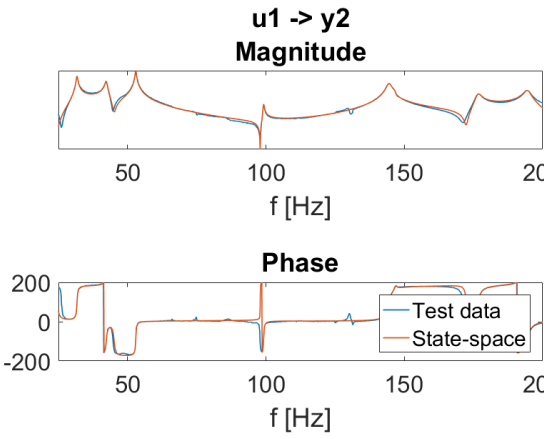


Figure B.1: Sensor 1 y-direction.

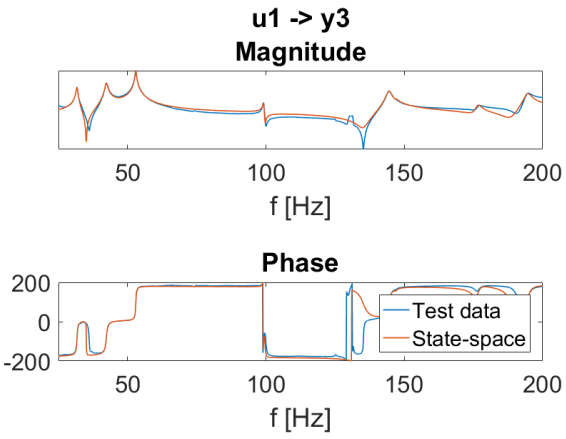


Figure B.2: Sensor 1 z-direction.

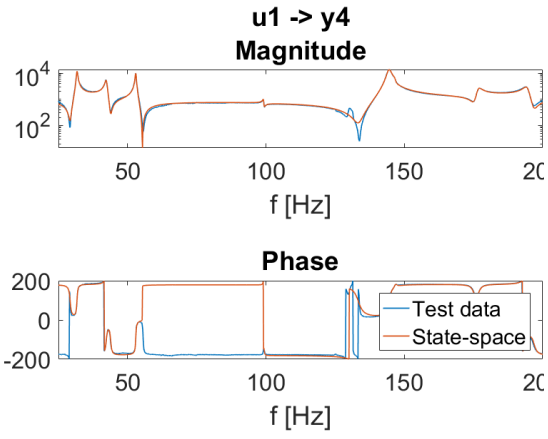


Figure B.3: Sensor 2 y-direction.

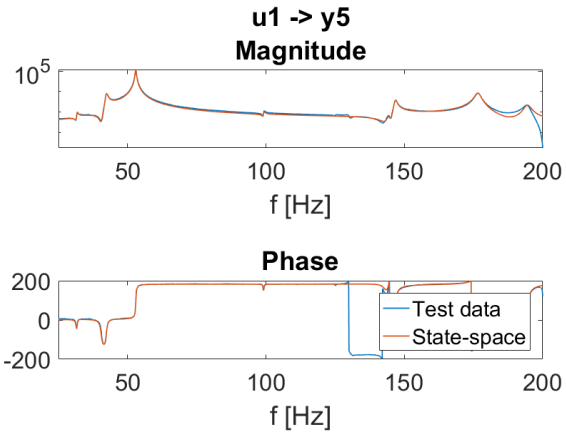


Figure B.4: Sensor 4 x-direction.

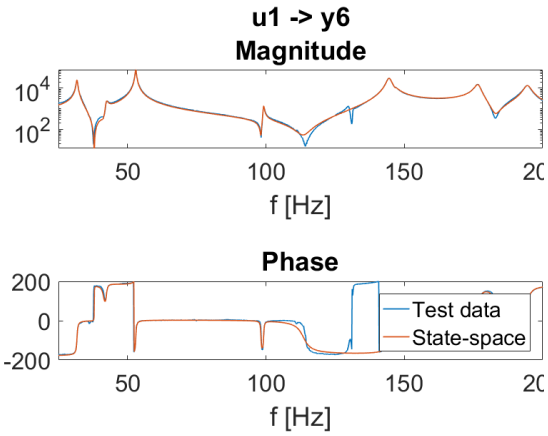


Figure B.5: Sensor 4 y-direction.

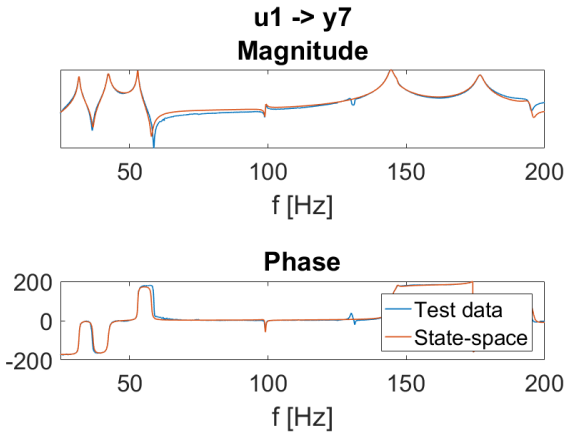


Figure B.6: Sensor 5 x-direction.

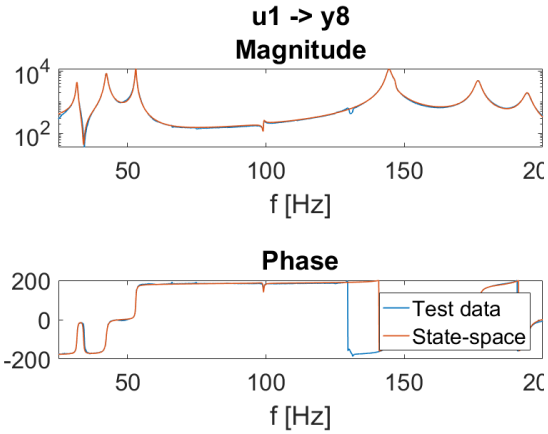


Figure B.7: Sensor 5 y-direction.

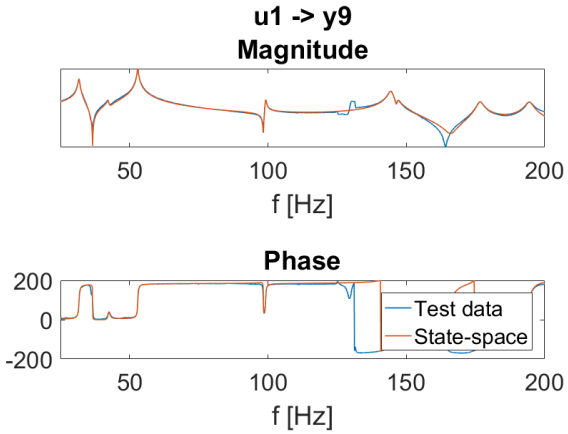


Figure B.8: Sensor 5 z-direction.

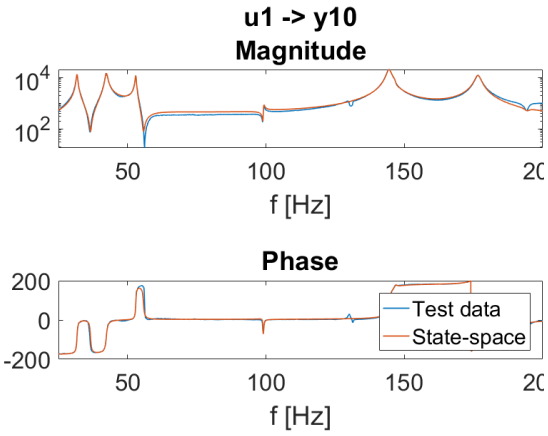


Figure B.9: Sensor 6 x-direction.

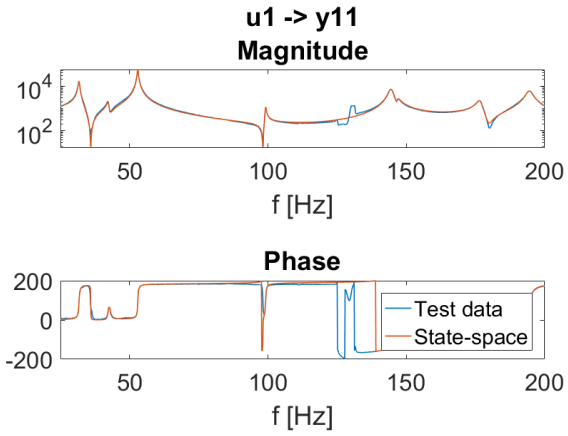


Figure B.10: Sensor 6 z-direction.

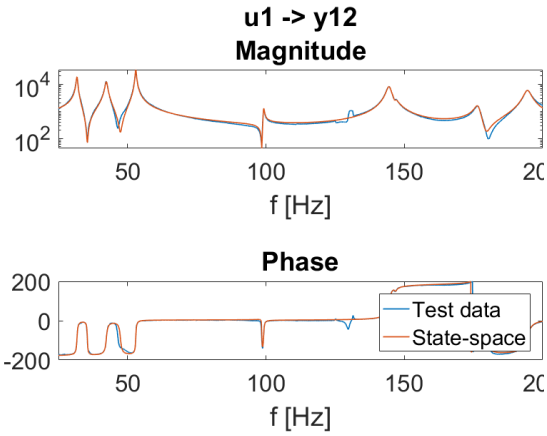


Figure B.11: Sensor 7 x-direction.

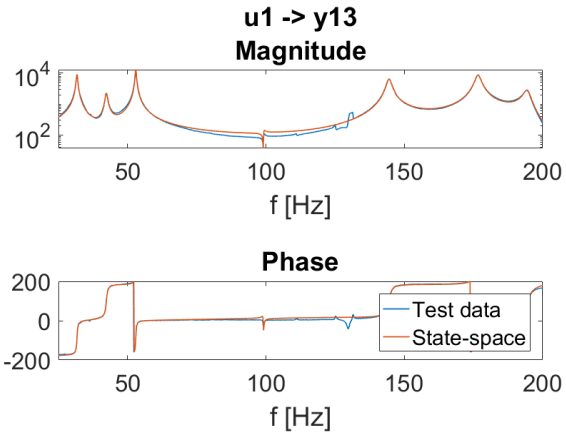


Figure B.12: Sensor 7 y-direction.

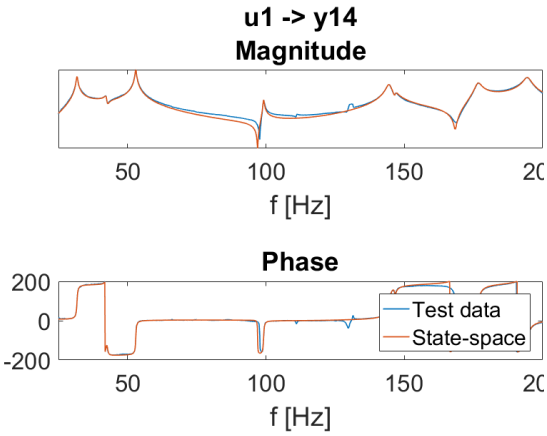


Figure B.13: Sensor 8 x-direction.

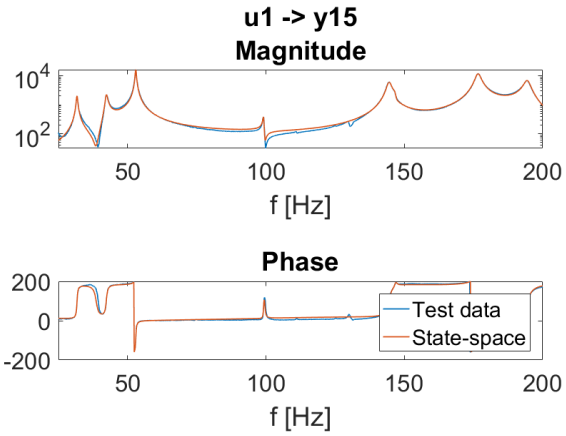


Figure B.14: Sensor 8 z-direction.

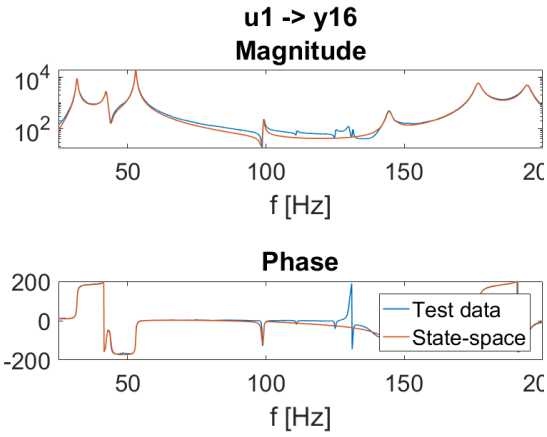


Figure B.15: Sensor 9 x-direction.

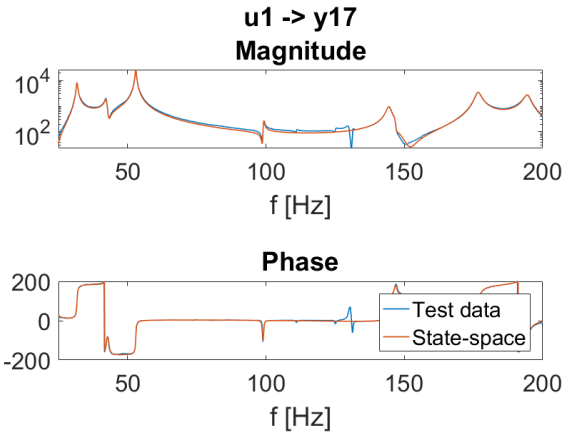


Figure B.16: Sensor 10 z-direction.

C

Transfer functions from calibration

Comparison of transfer functions between nominal and calibrated FE model, state-space model and test data as reference.

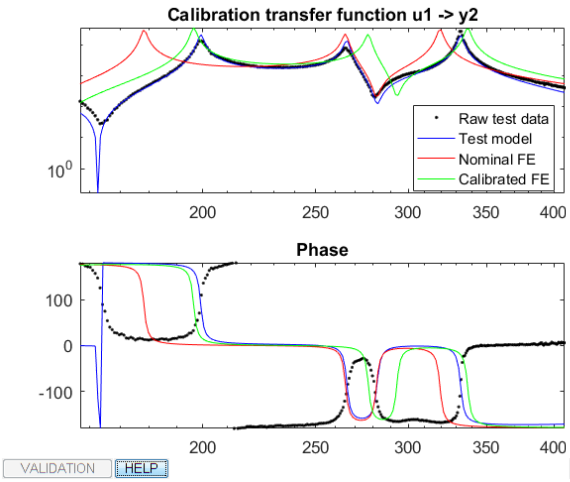


Figure C.1: Sensor 1 y-direction.

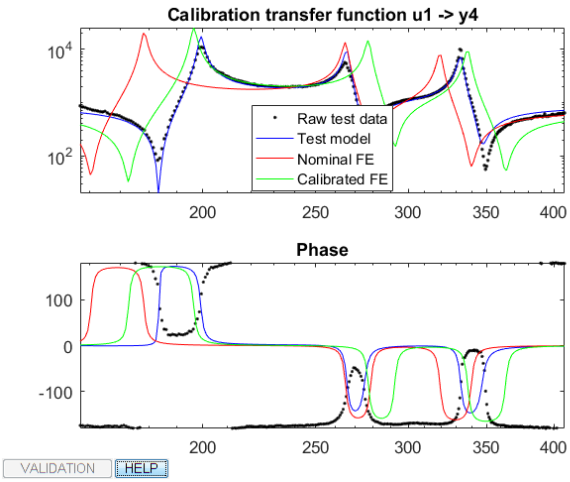


Figure C.2: Sensor 2 y-direction.

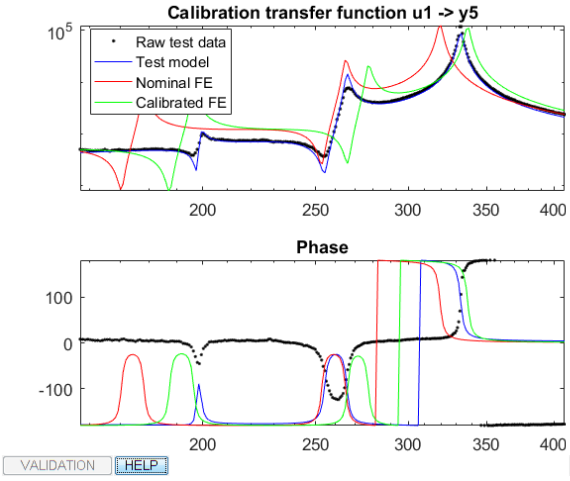


Figure C.3: Sensor 4 x-direction.

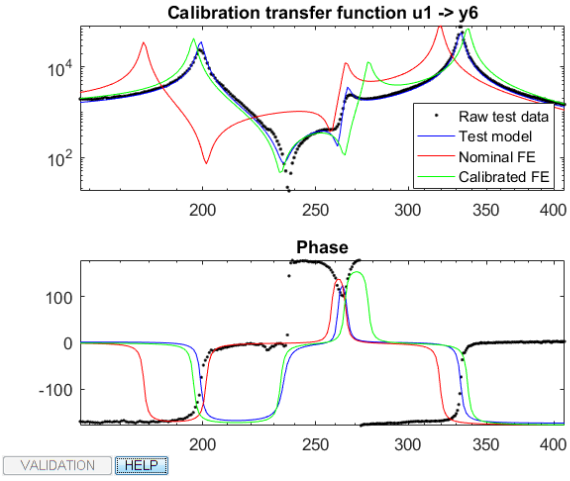


Figure C.4: Sensor 4 y-direction.

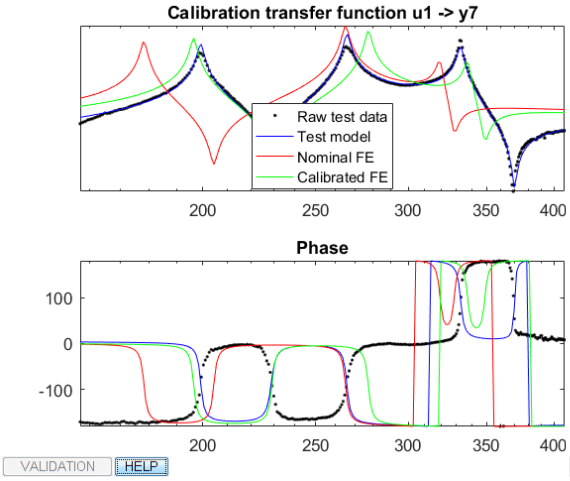


Figure C.5: Sensor 5 x-direction.

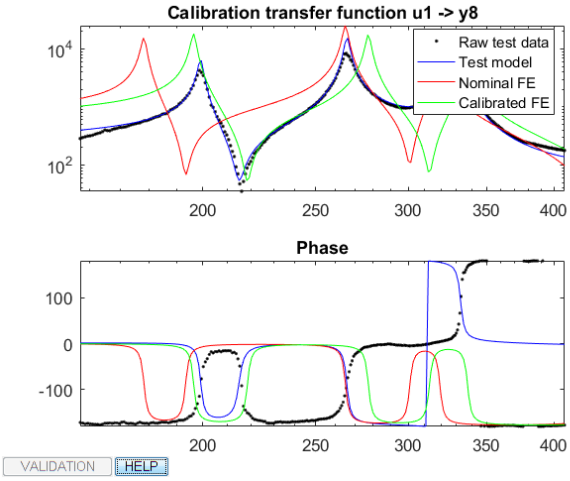


Figure C.6: Sensor 5 y-direction.

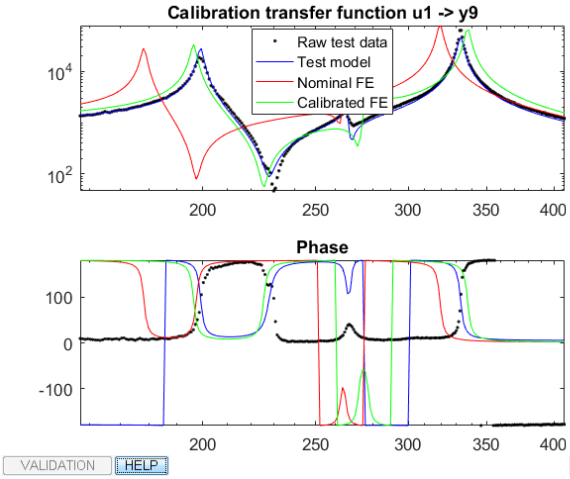


Figure C.7: Sensor 5 z-direction.

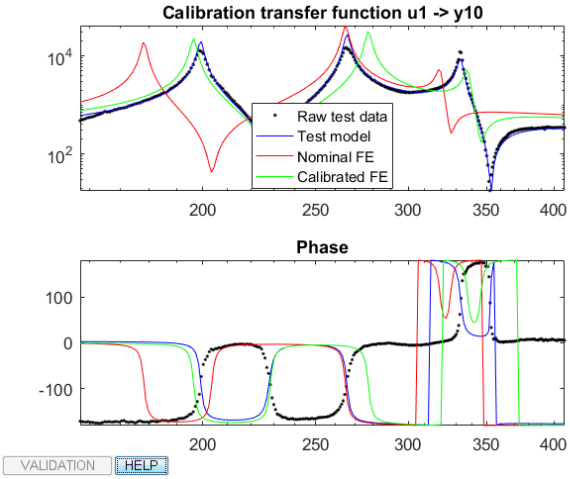


Figure C.8: Sensor 6 x-direction.

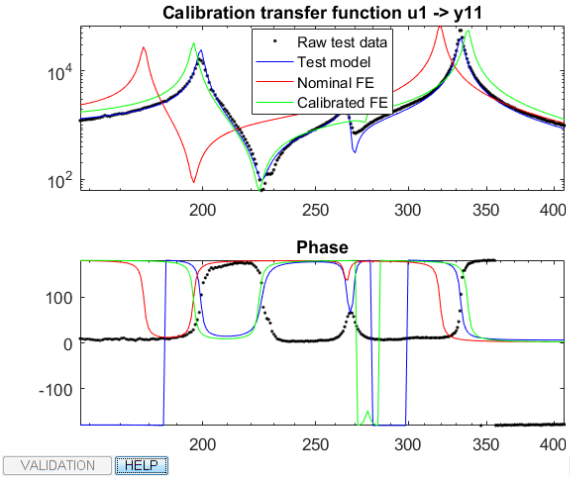


Figure C.9: Sensor 6 z-direction.

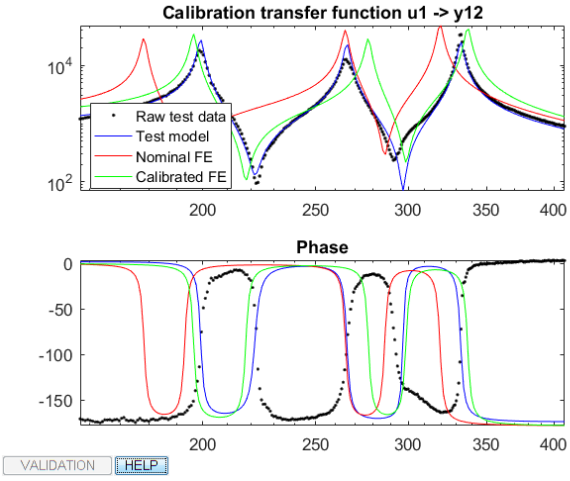


Figure C.10: Sensor 7 x-direction.

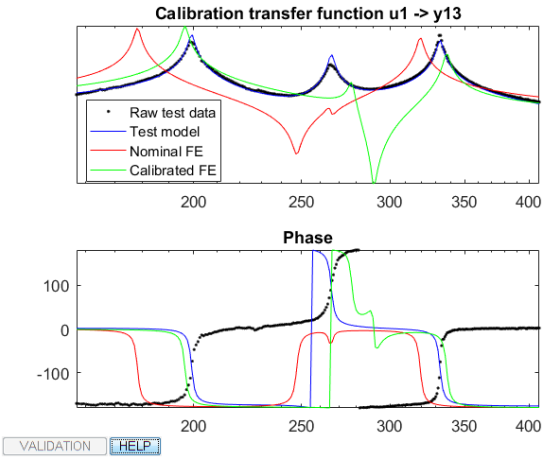


Figure C.11: Sensor 7 y-direction.

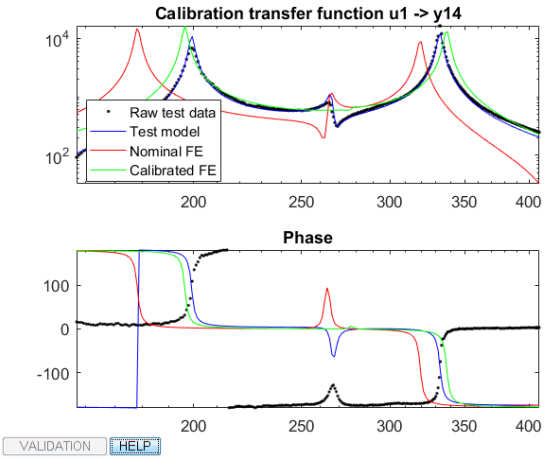


Figure C.12: Sensor 8 x-direction.

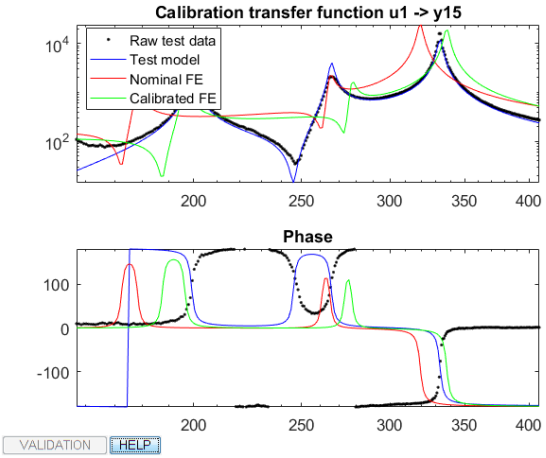


Figure C.13: Sensor 8 z-direction.

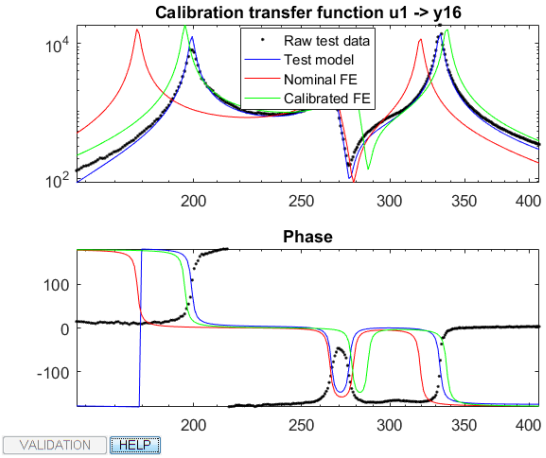


Figure C.14: Sensor 9 x-direction.

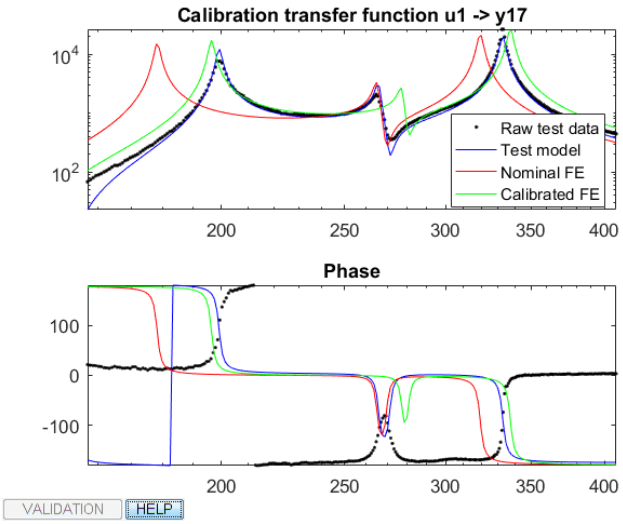


Figure C.15: Sensor 10 z-direction.

D

Matlab scripts

D.1 *our script*

The main Matlab script used for creating state-space models and doing the validation.

```
%% Task c
clear variables;
close all;
clc
load FRF_9_10acc_SI

set(0,'defaultaxesfontsize',20)
set(0,'defaulttextfontsize',20)
set(0,'defaultfigureposition',[400 350 700 500])
set(0,'defaultlinelinewidth',1)

% F == FRD object
% H == ResponseData
% f == frequency

val = input('\n 1. Choose what sensors that will be used for validation and/or calibration
val2 = input('\n 2. Choose frequency range\n Type "1" for 25-65Hz\n Type "2" for 25-200Hz\
val3 = input('\n 3. Choose ssm creation method\n Type "1" to use continuous frequency rang
val4 = input('\n 4. Choose what to compare with\n Type "1" for calibrated model\n Type "2"
val5 = input('\n 5. Choose to plot test-data and state-space models\n Type "1" for "yes"\n

if not(val==1 || val==2) == 1
    error('Error input 1')
end
if not(val2==1 || val2==2 || val2==3) == 1
    error('Error input 2')
end
if not(val3==1 || val3==2) == 1
    error('Error input 3')
end
if not(val4==1 || val4==2) == 1
    error('Error input 4')
end
if not(val5==1 || val5==2) == 1
    error('Error input 5')
end

%% Read responses
H = FRF.y_values.values;
Name = FRF.function_record.name;
```

```
% Define frequency vector
fIncr = FRF.x_values.increment;
fnVal = FRF.x_values.number_of_values;
f = (fIncr:fIncr:fnVal/(1/fIncr))*2*pi;

%For sensor 1 to 9:
%+X (Wanted direction along pipe)            = -Y (on sensor) = "+X" (name)
%+Y (Wanted direction perpendicular to pipe) = +X (on sensor) = "-Y" (name)
%+Z (Wanted direction normal to pipe)        = +Z (on sensor) = "+Z" (name)

%For sensor 10:
%+X (Wanted direction along pipe)            = +X (on sensor) = "+X" (name)
%+Y (Wanted direction perpendicular to pipe) = +Y (on sensor) = "+Y" (name)
%+Z (Wanted direction normal to pipe)        = +Z (on sensor) = "+Z" (name)

%Rearrange so that:
%H(:,1) = 1X (sensor 1, X-direction)
%H(:,2) = 1Y (sensor 1, Y-direction)
%H(:,3) = 1Z (sensor 1, Z-direction)
%H(:,4) = 2X (sensor 2, X-direction)
%H(:,5) = 2Y (sensor 2, Y-direction)
%...
%H(:,30) = 10Z (sensor 10, Z-direction)

%Sensor 1 to 10 vector positions
xInd10 = 1:3:30;
yInd10 = 2:3:30;
zInd10 = 3:3:30;

Hxyz = zeros(size(H));
xyzName = zeros(size(Name)); xyzName = num2cell(xyzName,size(xyzName));

%X-directions
Hxyz(:,xInd10) = -1.*H(:,[30 27 24 21 18 15 12 9 6 3]);
xyzName(:,xInd10) = Name(:,[30 27 24 21 18 15 12 9 6 3]);

%Y-directions
Hxyz(:,yInd10) = H(:,[28 25 22 19 16 13 10 7 4 3]);
xyzName(:,yInd10) = Name(:,[28 25 22 19 16 13 10 7 4 3]);

%Z-directions
Hxyz(:,zInd10) = H(:,[29 26 23 20 17 14 11 8 5 1]);
xyzName(:,zInd10) = Name(:,[29 26 23 20 17 14 11 8 5 1]);

Hxyz = Hxyz';
H = reshape(Hxyz,size(Hxyz,1),1,size(Hxyz,2));

%Convert from m/s^2/N to mm/s^2/N
H = H*1000;

%% Do not use blockwise state-space model method
% systemIdentification
if val3 == 1

f1 = f;
```

```
if val == 1
    H1 = H;
elseif val == 2
    H1 = H([1 2 3 5 10 11 13 14 15 16 18 19 20 22 24 25 30],:,:);
end

%Identify lower and upper frequency range
if val2 == 1
    indLow = find(f<25*2*pi);
    indUp = find(f>65*2*pi);
elseif val2 == 2
    indLow = find(f<25*2*pi);
    indUp = find(f>200*2*pi);
elseif val2 == 3
    indLow = find(f<25*2*pi);
    indUp = find(f>500*2*pi);
end

%Remove frequencies and responses
f1(indUp) = []; f1(indLow) = [];
H1(:,:,indUp) = []; H1(:,:,indLow) = [];
H1 = sum(H1);

%Create FRD-object
F1 = frd(H1,f1);

% State space model parameters
Options = n4sidOptions;
Options.Display = 'on';
Options.EstCovar = false;
Options.N4Weight = 'CVA';
Options.N4Horizon = [90 90 90];
if val2 == 3
    ssm = n4sid(F1, 36, 'Feedthrough', true, Options); %For f = 25-500Hz
elseif val2 == 2
    ssm = n4sid(F1, 18, 'Feedthrough', true, Options); %For f = 25-200Hz
elseif val2 == 1
    ssm = n4sid(F1, 6, 'Feedthrough', true, Options); %For f = 25-65Hz
end

end

%% Use blockwise state-space method
% systemIdentification
if val3 == 2

f2 = f;
if val == 1
    H2 = H;
elseif val == 2
    H2 = H([1 2 3 5 10 11 13 14 15 16 18 19 20 22 24 25 30],:,:);
end

%N4SID options
Options = n4sidOptions;
Options.Display = 'on';
```

```
Options.EstCovar = false;
Options.N4Weight = 'CVA';
Options.N4Horizon = [90 90 90];

%Block 1, 25-65Hz, 3 eigenfrequencies
f2B1 = f;
H2B1 = H2;
indLow = find(f<25*2*pi);
indUp = find(f>65*2*pi);
f2B1(indUp) = []; f2B1(indLow) = [];
H2B1(:,:,indUp) = []; H2B1(:,:,indLow) = [];
F2B1 = frd(H2B1,f2B1);
ssmB1 = n4sid(F2B1, 6, 'Feedthrough', true, Options);

%Block 2, 80-120Hz, 1 eigenfrequencies
if val2 == 2 || 3
    f2B2 = f;
    H2B2 = H2;
    indLow = find(f<80*2*pi);
    indUp = find(f>120*2*pi);
    f2B2(indUp) = []; f2B2(indLow) = [];
    H2B2(:,:,indUp) = []; H2B2(:,:,indLow) = [];
    F2B2 = frd(H2B2,f2B2);
    ssmB2 = n4sid(F2B2, 2, 'Feedthrough', true, Options);
end

%Block 3, 140-200Hz, 4 eigenfrequencies
if val2 == 2 || 3
    f2B3 = f;
    H2B3 = H2;
    indLow = find(f<140*2*pi);
    indUp = find(f>200*2*pi);
    f2B3(indUp) = []; f2B3(indLow) = [];
    H2B3(:,:,indUp) = []; H2B3(:,:,indLow) = [];
    F2B3 = frd(H2B3,f2B3);
    ssmB3 = n4sid(F2B3, 8, 'Feedthrough', true, Options);
end

%Block 4, 200-280Hz, 2 eigenfrequencies
if val2 == 3
    f2B4 = f;
    H2B4 = H2;
    indLow = find(f<200*2*pi);
    indUp = find(f>280*2*pi);
    f2B4(indUp) = []; f2B4(indLow) = [];
    H2B4(:,:,indUp) = []; H2B4(:,:,indLow) = [];
    F2B4 = frd(H2B4,f2B4);
    ssmB4 = n4sid(F2B4, 4, 'Feedthrough', true, Options);
end

%Block 5, 280-500Hz, 8 eigenfrequencies
if val2 == 3
    f2B5 = f;
    H2B5 = H2;
    indLow = find(f<280*2*pi);
    indUp = find(f>500*2*pi);
```

```
    f2B5(indUp) = []; f2B5(indLow) = [];
    H2B5(:,:,indUp) = []; H2B5(:,:,indLow) = [];
    F2B5 = frd(H2B5,f2B5);
    ssmB5 = n4sid(F2B5, 16, 'Feedthrough', true, Options);
end

%Assemble state-space models
if val2 == 1 || 2 || 3
    ssm = ssmB1;
end
if val2 == 2 || 3
    ssm2 = parallel(ssmB1,ssmB2);
    ssm2 = parallel(ssm2,ssmB3);
end
if val2 == 3
    ssm2 = parallel(ssm2,ssmB4);
    ssm2 = parallel(ssm2,ssmB5);
end

%Identify lower and upper frequency range
if val2 == 1
    indLow = find(f<25*2*pi);
    indUp = find(f>65*2*pi);
elseif val2 == 2
    indLow = find(f<25*2*pi);
    indUp = find(f>200*2*pi);
elseif val2 == 3
    indLow = find(f<25*2*pi);
    indUp = find(f>500*2*pi);
end

%Remove frequencies
f2(indUp) = []; f2(indLow) = [];
H2(:,:,indUp) = []; H2(:,:,indLow) = [];

%Create FRD object
F2 = frd(H2,f2);

%Calculate C,D matrices again
ssm = ff2cdest(ssm2,F2);

end

%% Examine phase and response
close all

if val5 == 1
    if val3 == 1
        Fta = frd(ssm,F2.Frequency);
        Frd = F2;
    elseif val3 == 2
        Fta = frd(ssm,F1.Frequency);
        Frd = F1;
    end
    opt.hold = true;
    for i = 1:length(ssm)
```

```
        str = strcat('u1 -> y',num2str(i));
        opt.title = {str;'Magnitude'};

        figure
        magphase([1 i;0 inf],Frd);
        magphase([1 i;0 inf],Fta,opt);
        legend('Test data','State-space')
    end
end

%% Extract damping and eigenfrequencies
clear q b ef dam

[natural_freqs,damping_factors] = damp(ssm.A); % Eigenmodes and damping
ef(:,1) = round(unique(natural_freqs(:,1)./2./pi),1); %The eigenfrequencys in Hz
dam(:,1) = unique(damping_factors(:,1)); %The dampings in procent

%% Create state-space for x-,y- and z-direction separately, redcued frequency spectrum

%Calculate eigenvector matrix for test data
%Size of matrix Channels*Eigenvectors
A = ssm.A;
C = ssm.C;
B = ssm.B;
D = ssm.D;
[eVect,eVal] = eig(A);
eVal = abs(eVal);
del = 2:2:size(eVal,1);
eVal(:,del) = [];
eVal(del,:) = [];
eVal = diag(eVal)/2/pi;
[eVal,index] = sort(eVal);
eVect = eVect(:,1:2:end);
for i = 1:length(index)
    newEV(:,i) = eVect(:,index(i));
end
eVect = newEV;
Z1 = C*eVect;

%Calculate eigenvector matrix for FEM data
if val2 == 2
    if val4 == 1
        FEM = readpunch('20_parameterized_rho_e_t_calibratedrun.pch');
    elseif val4 == 2
        FEM = readpunch('v4_norubber_coarsemesh_changedcoord_v2_20to280hz.pch');
    end
end

%Sensors matched with nodes in ANSA
sensor = [1 2 3 4 5 6 7 8 9 10];
nModes = length(FEM.Data);
Z2 = zeros(30,nModes);
for i = 1:nModes %28 if 28 eigenvectors
    R = 1;
    for j = 1:10 %Looped for 10 sensors
        Z2(R,i)    = FEM.Data{i}(sensor(j),4);    %X-direction
```

```
        Z2(R+1,i) = FEM.Data{i}(sensor(j),2);    %Y-direction
        Z2(R+2,i) = FEM.Data{i}(sensor(j),3);    %Z-direction
        R = R+3;
    end
end

if val == 2
    Z2_calibration = Z2([1 2 3 5 10 11 13 14 15 16 18 19 20 22 24 25 30],:);
    Z2 = Z2_calibration;
end

figure
MAC = mac(Z2_calibration,Z1);
MAC(:,[9 10 11]) = [];
matplot(MAC,[0 1])
xlabel('FEA')
ylabel('SSM')
title('Post calibration')

MACdiag = diag(MAC);
MACmean = mean(MACdiag(1:6));

%% Save data for FEMcali

%Change phase by 180degrees to match FEM
C(:,:) = C(:,:).*-1;
D(:,:) = D(:,:).*-1;

% AX = A;
% BX = B;
% CX = C;
% DX = D;
% if val3 == 1
%     ResponseData = H2;
%     Frequency = f2;
% elseif val3 == 2
%     ResponseData = H1;
%     Frequency = f1;
% end
%
% save('FEMCaliSI_calibrationSet2.mat','AX','BX','CX','DX','ResponseData','Frequency')
```

D.2 *efi.m*

Matlab implementation for the method of effective independence. The script is used for calculating the best positions for placing accelerometers.

```
function [indr,inds,d]=efi(V,N,trac,XYZ,dirs,dN,tpaus)
%EFI: Calculates the optimal sensor locations by use of EFfective Indedendence
%Given the mode matrix V, the N sensor locations giving the best observability
%is calculated by iterations. An index vector (indr) is calculated such that
%the partition V(indr,:) of V is associated to optimal sensor location.
%
%Inputs:  V     - mode matrix (nxm, m is number of modes)
%         N     - number of sensors in reduced sensor set (N>=m, default N=m)
%         trac  - if nonzero information will be given during reduction
%         XYZ   - nodal coordinate matrix (used if trac~=0)
%         dirs  - measurement direction information (1, 2 or 3)
%         dN    - delay number (Sensors are retained as marks on the plots
%                 this number of times)
%         tpaus - Pause time after each iteration (Default 0)
%Outputs: indr  - index of retained sensor locations
%         inds  - index of skipped sensor locations
%         d     - determinant of Fisher information matrix at each iteration
%Call:    [indr,inds,d]=efi(V[,N,trac,XYZ,dirs,dN,tpaus])

%Reference: Kammer D. & Brillhart R., 'Optimal Sensor Placement for Modal
%           Identification using System-Realization Methods', to appear in
%           Journal of Guidance, Control and Dynamics
%Copyleft: Thomas Abrahamsson, Linkping, Sweden
%           April 25, 1994 /TA
%Modified:  April 27, 1994 /TA
%Modified:  April 28, 1994 /TA
%Modified:  April 12, 1995 /TA

% ------------------------------------------------------------------------
%                                                                  Initiate
%                                                                  --------
[n,k]=size(V);
if ~exist('N'),N=k;elseif isempty(N),N=k;end
if ~exist('trac'),trac=[];end
if ~exist('XYZ'),XYZ=[];end
if ~exist('dN'),dN=10;elseif isempty(dN),dN=10;end
if ~exist('tpaus'),tpaus=0;elseif isempty(tpaus),tpaus=0;end
indr=1:n;inds=[];d=[];
symb=['yo';'go';'bo';'ko';'r.'];
rindsx=[];rindsy=[];rindsz=[];

Xwmax=1024;Ywmax=768;
% if strcmp(computer,'SGI'),
%   Xwmax=1272;Ywmax=992;
% elseif strcmp(computer,'PCWIN')
%   Xwmax=800;Ywmax=600;
% end

% ------------------------------------------------------------------------
%                                          If trace is on then initiate plot
```

```
%                                                 --------------------------------
if ~isempty(trac),
  if isempty(XYZ),
    clf
    plot(indr,0*ones(n,1),'g.')
    hold on,drawnow
  else
    xx=XYZ(find(dirs==1),1);xy=XYZ(find(dirs==2),1);xz=XYZ(find(dirs==3),1);
    yx=XYZ(find(dirs==1),2);yy=XYZ(find(dirs==2),2);yz=XYZ(find(dirs==3),2);
    zx=XYZ(find(dirs==1),3);zy=XYZ(find(dirs==2),3);zz=XYZ(find(dirs==3),3);
    addrx=indr(find(dirs==1));addry=indr(find(dirs==2));addrz=indr(find(dirs==3));
    if ~isempty(addrx),
       indsx(length(addrx),1)=0;
       EFIFIG1=figure;
       set(gcf,'NumberTitle','Off');
       set(gcf,'Position',[1 Ywmax-(420/560)*Xwmax/3-3 Xwmax/3 (420/560)*Xwmax/3]);
       set(gcf,'Name',' X-direction')
       plot3(xx,yx,zx,'ro')
       axis('image'),vx=axis;view(90,0);[azx,elx]=view;
       dx=vx(4)-vx(1);dy=vx(5)-vx(2);dz=vx(6)-vx(3);
       vx=[vx(1)-dx/100 vx(2)-dy/100 vx(3)-dz/100 vx(4)+dx/100 vx(5)+dy/100 vx(6)+dz/100];
       set(gca,'Visible','off');
    end
    if ~isempty(addry),
      indsy(length(addry),1)=0;
      EFIFIG2=figure;
      set(gcf,'NumberTitle','Off');
      set(gcf,'Position',[Xwmax/3 Ywmax-(420/560)*Xwmax/3-3 Xwmax/3 (420/560)*Xwmax/3]);
      set(gcf,'Name',' Y-direction')
      plot3(xy,yy,zy,'ro')
      axis('image'),vy=axis;view(90,0);[azy,ely]=view;
      dx=vy(4)-vy(1);dy=vy(5)-vy(2);dz=vy(6)-vy(3);
      vy=[vy(1)-dx/100 vy(2)-dy/100 vy(3)-dz/100 vy(4)+dx/100 vy(5)+dy/100 vy(6)+dz/100];
      set(gca,'Visible','off');
    end
    if ~isempty(addrz),
      indsz(length(addrz),1)=0;
      EFIFIG3=figure;
      set(gcf,'NumberTitle','Off');
      set(gcf,'Position',[2*Xwmax/3 Ywmax-(420/560)*Xwmax/3-3 Xwmax/3 (420/560)*Xwmax/3]);
      set(gcf,'Name',' Z-direction')
      plot3(xz,yz,zz,'ro')
      axis('image'),vz=axis;view(0,90);[azz,elz]=view;
      dx=vz(4)-vz(1);dy=vz(5)-vz(2);dz=vz(6)-vz(3);
      vz=[vz(1)-dx/100 vz(2)-dy/100 vz(3)-dz/100 vz(4)+dx/100 vz(5)+dy/100 vz(6)+dz/100];
      set(gca,'Visible','off');
    end
    drawnow
  end
end

% ------------------------------------------------------------------------
%                                                 Reduce number of sensors until
%                                                 N sensors are left. Update the
%                                                 incidences arrays indr and
%                                                 inds in the process
```

```
%                                                    ------------------------------
ns=n;I=0;
while ns > N,
  I=I+1;
  Q=V'*V;%Form information matrix
  if nargout>=3,d=[d det(Q)];end
  [Psi,lam]=eig(Q);invlam=diag(1./diag(lam));
  G=conj(V*Psi).*(V*Psi);Fe=G*invlam;Ed=sum(Fe')';
  [Edmin,ind]=min(Ed);
  inds=[inds indr(ind)];
  ns=ns-1;


  if ~isempty(trac),
    if isempty(XYZ),
      plot(inds,0*ones(length(inds),1),'k.')
      plot(inds,(n-ns)*ones(length(inds),1),'y.'),pause(0)
    else

      if ~isempty(addrx),
        indsx(length(addrx),I)=0;
        ix=find(addrx==indr(ind));if ~isempty(ix),indsx(ix,I)=1;end
        figure(EFIFIG1),clg,hold on,title('X'),axis(vx);view(azx,elx);
        set(gca,'Visible','off');
        for II=1:5,
          III=II;if II==5,III=II-1;end
          In=max(1,I-(III-1)*dN-dN+1):I-(III-1)*dN;
          if ~isempty(In),
            if ~isempty(addrx),rindsx=find(sum(indsx(:,In)')'~=0);end
            if ~isempty(rindsx),
              plot3(xx(rindsx),yx(rindsx),zx(rindsx),symb(II,:))
            end
          end
        end
        if ~isempty(addrx),rindsx=find(sum(indsx')'==0);end
        if ~isempty(rindsx),
          plot3(xx(rindsx),yx(rindsx),zx(rindsx),'ro')
        end
        if ~isempty(addrx),rindsx=find(sum(indsx')'~=0);end
        if ~isempty(rindsx),
          plot3(xx(rindsx),yx(rindsx),zx(rindsx),'y.')
        end
      end

      if ~isempty(addry),
        indsy(length(addry),I)=0;
        iy=find(addry==indr(ind));if ~isempty(iy),indsy(iy,I)=1;end
        figure(EFIFIG2),clg,hold on,title('Y'),axis(vy);view(azy,ely);
        set(gca,'Visible','off');
        for II=1:5,
          III=II;if II==5,III=II-1;end
          In=max(1,I-(III-1)*dN-dN+1):I-(III-1)*dN;
          if ~isempty(In),
            if ~isempty(addry),rindsy=find(sum(indsy(:,In)')'~=0);end
            if ~isempty(rindsy),
              plot3(xy(rindsy),yy(rindsy),zy(rindsy),symb(II,:))
```

```
          end
        end
      end
      if ~isempty(addry),rindsy=find(sum(indsy')'==0);end
      if ~isempty(rindsy),
        plot3(xy(rindsy),yy(rindsy),zy(rindsy),'ro')
      end
      if ~isempty(addry),rindsy=find(sum(indsy')'~=0);end
      if ~isempty(rindsy),
        plot3(xy(rindsy),yy(rindsy),zy(rindsy),'y.')
      end
    end

      if ~isempty(addrz),
      indsz(length(addrz),I)=0;
      iz=find(addrz==indr(ind));if ~isempty(iz),indsz(iz,I)=1;end
      figure(EFIFIG3),clf,hold on,title('Z'),axis(vz);view(azz,elz);
      set(gca,'Visible','off');
      for II=1:5,
        III=II;if II==5,III=II-1;end
        In=max(1,I-(III-1)*dN-dN+1):I-(III-1)*dN;
        if ~isempty(In),
          if ~isempty(addrz),rindsz=find(sum(indsz(:,In)')'~=0);end
          if ~isempty(rindsz),
            plot3(xz(rindsz),yz(rindsz),zz(rindsz),symb(II,:))
          end
        end
      end
      if ~isempty(addrz),rindsz=find(sum(indsz')'==0);end
      if ~isempty(rindsz),
        plot3(xz(rindsz),yz(rindsz),zz(rindsz),'ro')
      end
      if ~isempty(addrz),rindsz=find(sum(indsz')'~=0);end
      if ~isempty(rindsz),
        plot3(xz(rindsz),yz(rindsz),zz(rindsz),'y.')
      end
    end


    pause(tpaus)

  end

 end
 indr(ind)=[];
 V(ind,:)=[];

end
```

D.3 *ff2cdest.m*

This Matlab script re-estimates the C- and D-matrices of the state-space models.

```
function SSout=ff2cdest(SSin,FRD,opt)
%FF2CEST: Re-estimate the C and D matrices of the state-space model SS by
%         least-squares fitting to data
%Inputs: SSin  - The state-space model with C and D-matrices to be (re-)estimated
%        FRD   - A FRD object with frequency response data (needs to be
%                given in rad/s)
%        opt   - opt.Conly=true re-estimates only the C matrix
%                opt.Donly=true re-estimates only the D matrix
%                opt.secondorder=true gives C corresponding to 2nd order model
%                  If true, also the type of FRF needs to be specified. Give
%                  either; opt.type='receptance', opt.type='mobility' or
%                  opt.type='accelerance'
%Output: SSout - The the re-estimated state-space model
%Call:   SSout=ff2cest(SSin,FRD,opt)

%Written: 2014-03-21, Thomas Abrahamsson, Chalmers University of Technology
%Modified: 2015-07-15, Introduced option for 2nd order model /TA
%Modified: 2015-10-02, Changed logic for 2nd order model options /TA
%Modified: 2015-10-03, Included re-estimation of D /TA
%Modified: 2015-10-23, Set NaN:s to zeros /TA

%%                                                                  Initiate
if nargin<3, opt.secondorder=false;end
[A,B,C0,D0]=ssdata(SSin);
[FRF,w]=frdata(FRD);nf=length(w);
n=size(A,1);ny=size(FRF,1);nu=size(B,2);n2=n/2;
%if any(D0(:)~=0),warning('D will be set to zero');end
if opt.secondorder && abs(floor(n2)*2-n)>.1
  error('Cannot be 2nd order model. Uneven number of states.')
end
if ~isfield(opt,'Conly'),opt.Conly=false;end
if ~isfield(opt,'Donly'),opt.Donly=false;end
if opt.Conly && opt.Donly, error('opt.Conly and opt.Donly cannot both be true');end
if opt.secondorder && opt.Donly
  opt.Donly=false;
  warning('opt.secondorder and opt.Donly cannot both be true. Setting opt.Donly=false ...'
end
if opt.secondorder && ~isfield(opt,'type')
  error('opt.type needs to be given for 2nd order model')
end

%%                                                              Extract data
FreqUnit=FRD.FrequencyUnit;TimeUnit=FRD.TimeUnit;
if ~(strcmpi(FreqUnit,'rad/s') || strcmpi(FreqUnit,'rad/seconds') || ...
    (strcmpi(FreqUnit,'rad/TimeUnit') && strcmpi(TimeUnit,'seconds')))
  error('Unit of data in FRD object needs to be rad/s')
end

%%                                                  Compute state response
SSx=ss(A,B,eye(n,n),0);
X=frdata(frd(SSx,w));
```

```
if opt.secondorder && (strcmp(opt.type,'mobility') || strcmp(opt.type,'accelerance'))
  X=X(n2+1:end,:,:);
elseif opt.secondorder && strcmp(opt.type,'receptance')
  X=X(1:n2,:,:);
elseif opt.secondorder
  error('For 2nd order model opt.type must be given as either receptance, mobility or acce
end

%%                                    Set up and solve least-squares problem
FRF=reshape(FRF,ny,nu*nf);FRF=[FRF conj(FRF)];
if opt.secondorder
  X=reshape(X,n2,nu*nf);X=[X conj(X)];
else
  X=reshape(X,n,nu*nf);X=[X conj(X)];
end
if opt.Conly
  C=real((FRF-D0*repmat(eye(nu),1,2*nf))*pinv(X));
  D=D0;
elseif opt.Donly
  C=C0;
  D=real((FRF-C0*X)*pinv(repmat(eye(nu),1,2*nf)));
else
  CD=real(FRF*pinv([X;repmat(eye(nu),1,2*nf)]));
  if opt.secondorder
     C=CD(:,1:n2);
     D=CD(:,n2+1:end);
  else
     C=CD(:,1:n);
     D=CD(:,n+1:end);
  end
end

%%                                                          Eliminate NaN:s
[row,col]=find(isnan(C));;
C(row,col)=0;
[row,col]=find(isnan(D));;
D(row,col)=0;

if opt.secondorder && (strcmp(opt.type,'mobility') || strcmp(opt.type,'accelerance'))
  C=[0*C C];
elseif opt.secondorder && strcmp(opt.type,'receptance')
  C=[C 0*C];
elseif opt.secondorder
  error('Unknown opt.type')
end

%%                                                  Make state-space model
SSout=ss(A,B,C,D);
```

D.4 *magphase.m*

Matlab script for conveniently plotting the magnitude and phase of test data and state-space models.

```
function [m,p]=magphase(f,frf,opt)
%MAGPHASE: Plot and calculate magnitude and phase of frf
% If called by no output arguments only a plot will be made
% No plot will be produced if output arguments are given
%
% Alternative 1:
%Inputs:   f    - Frequencies (in Hz) associated to frf
%          frf  - Frequency response function (Complex)
%          opt.loglog Set to true creates a loglog plot (default linlog)
%          opt.hold   If true, sets figure status to "hold on" before
%                     plotting
%          opt.grid   Set to true plots a grid
%          opt.ls     LineStyle (Default is Matlab's plot default)
%          opt.color  Line color
%          opt.magn   If true, only plot magnitude
%          opt.ax     Axis setting for magnitude plot
%Output:   m    - Magnitude of frf
%          p    - Phase of frf (in degrees)
%          opt  - See above
%Call:     [m,p]=magphase(f,frf,opt)
%
%Alternative 2:
%Inputs:   [In Out;  - Identifier for input channel # (In) and output
%                      channel # (Out)
%           flo fhi] - Lower and upper frequency for plotting
%          FRDorSS   - FRD or SS object. If FRDorSS is a FRD object
%                      and has FRD.UserData.Coherence as subfield a
%                      coherence plot will be superimposed on figure
%          opt       - See alternative 1 above
%Output:   m    - Magnitude of frf
%          p    - Phase of frf (in degrees)
%Call:     [m,p]=magphase([In Out;flo fhi],FRDorSS,opt)

%Modified: Nov 11, 2001 (real and imag was switched in atan2)
%Modified: April 16, 2013 changed to use angle and added opt /TA
%Modified: March 8, 2014 modified to also handle FRD objects /TA
%Modified: April 28, 2014 avoid phase flip-flops /TA
%Modified: June 29, 2015 to also treat SS objects
%Modified: July  7, 2015 to include plot title /TA
%Modified: July 11, 2015 added coherence plot /TA

if nargin<3,opt.loglog=false;opt.hold=false;end
if ~isfield(opt,'loglog'),opt.loglog=false;end
if ~isfield(opt,'hold'),opt.hold=false;end
if ~isfield(opt,'grid'),opt.grid=false;end
if ~isfield(opt,'ls'),opt.ls='';end
if ~isfield(opt,'magn'),opt.magn=false;end
if ~isfield(opt,'ax'),opt.ax=[];end
if ~isfield(opt,'title'),opt.title=[];end
if ~isfield(opt,'rad'),opt.rad=false;end
```

```
if strcmpi(class(frf),'frd') || strcmpi(class(frf),'idfrd')
  if min(size(f))==1
    f=f(:)';
    f(2,1)=0;f(2,2)=Inf;% Default for lower and upper frequency
  end
end
if strcmpi(class(frf),'ss') || strcmpi(class(frf),'idss')
  if get(frf,'Ts')>0,error('Cannot treat discrete-time state-space models');end
  [Wn,zeta] = damp(frf);zetamin=min(zeta);
  if min(size(f))==1
    f=f(:)';
    f(2,1)=0.8*Wn(1)/2/pi;% Default for lower and upper frequency
    f(2,2)=1.2*Wn(end)/2/pi;
  end
end

range=false;

%%                                                    Get data from FRD object
if strcmpi(class(frf),'frd') || strcmpi(class(frf),'idfrd')
  if strcmpi(get(frf,'FrequencyUnit'),'rad/s')
    f0=frf.Frequency/2/pi;
  elseif strcmpi(get(frf,'FrequencyUnit'),'Hz')
    f0=frf.Frequency;
  elseif strcmpi(get(frf,'FrequencyUnit'),'rad/TimeUnit') & strcmpi(get(frf,'TimeUnit'),'s
    f0=frf.Frequency/2/pi;
  else
    error('Unknown frequency unit. Only rad/s and Hz allowed.')
  end
  ind=find(f0>=f(2,1) & f0<=f(2,2));
  try
    mrange=frf.UserData.mrange;
    mrange=squeeze(mrange(f(1,2),f(1,1),ind,:));
    range=true;
  catch
    range=false;
  end
  try
    coh=squeeze(frf.UserData.Coherence(f(1,2),f(1,1),ind));
    if max(abs(diff(coh)))<.001,coh=NaN(size(coh));end
  catch
    coh=[];
  end
  frf=squeeze(frf.ResponseData(f(1,2),f(1,1),ind));
  f=f0(ind);
else
  coh=[];
end

%%                                                     Get data from SS object
if strcmpi(class(frf),'ss') || strcmpi(class(frf),'idss')
  if opt.hold
     wlim=2*pi*get(gca,'Xlim');
     w=linspace(wlim(1),wlim(2),1000);
  else
     w=linspace(2*pi*f(2,1),2*pi*f(2,2),1000);
```

```
  end
%   if zetamin>.001 && Wn(end)/Wn(1)>2
%      w=wlogspace(0.8*Wn(1),1.2*Wn(end),5,zetamin);
%   else
%      w=.95*Wn(1):Wn(end)/1000:1.05*Wn(end);
%   end
%   w=w(w<=2*pi*f(2,2));
%   while length(w)>1600, w=w(1:2:end);end
  FRD=frd(frf,w);
  frf=squeeze(FRD.ResponseData(f(1,2),f(1,1),:));
  f=w/2/pi;
end

m=abs(frf);
p=angle(frf)*180/pi;

%%                          Make an attempt to make phase do less flip-flop
if size(p,2)~=length(f),p=p';end
if size(p,2)~=length(f),error('Size of transfer function does not match length of frequenc
for I=2:length(f)
  for J=1:size(p,1)
    if abs(abs(p(J,I))-180)<20;% Within 10 degrees of +-180 degrees?
      if abs(abs(p(J,I-1))-180)<20
        if abs(p(J,I)-p(J,I-1))>40
          if p(J,I)<0
             p(J,I)=p(J,I)+360;
          else
             p(J,I)=p(J,I)-360;
          end
        end
      end
    end
  end
end

%%                                                                  Do plots
if nargout==0,
  if opt.rad;f=2*pi*f;end

%   if ~opt.hold,axis('auto');end
  if ~opt.magn, subplot(211);end
  if opt.hold, hold on,else, hold off,end
  if ~opt.hold,axis('auto');end

%   if ~opt.hold,cla;end
  if ~opt.loglog
    if range
      fr=[f(:) f(:) f(:)]';fr=fr(:);
      mrange=[mrange NaN*zeros(size(mrange,1),1)]';
      mrange=mrange(:);
      hl=semilogy(fr,mrange,f,m,opt.ls);
      set(hl(1),'Color',[.6 1 .6],'LineWidth',.01);
      set(hl(2),'MarkerSize',4);
      try set(hl(2),'Color',opt.color);catch,end
    else
      if isempty(coh)
```

```
        try
          hLine1=semilogy(f,m,opt.ls);
        catch
          hLine1=semilogy(f,m);
        end
        try, hLine1.Color=opt.color;catch,end
      else
        [hl,hLine1,hLine2]=plotyy(f,m,f,coh,'semilogy','plot');
        YTick=hl(1).YTick;YTick=10.^[log10(YTick(1)):log10(YTick(end))];
        hl(1).YTick=YTick;
        try,hLine1.LineStyle=opt.ls;catch,end
        try, hLine1.Color=opt.color;catch,end
        hLine2.Color=[.6 .6 .6];
      end
%       hl=semilogy(f,m,opt.ls);
%       try set(hl(1),'Color',opt.color);catch,end
    end
    axis tight
    ax=axis;
%     expo=10^floor(log10(f(end)));
%     axis([round(f(1),1,'significant') (round(f(end),1,'significant')/expo+1)*expo ax(3)
%     ax=axis;
  else
    if range
      fr=[f(:) f(:) f(:)]';fr=fr(:);
      mrange=[mrange NaN*zeros(size(mrange,1),1)]';
      mrange=mrange(:);
      hl=loglog(fr,mrange,f,m,opt.ls);,
      set(hl(1),'Color',[.6 1 .6],'LineWidth',.01);
      set(hl(2),'MarkerSize',4);
      try set(hl(2),'Color',opt.color);catch,end
    else
      if isempty(coh)
        try
          hLine1=semilogy(f,m,opt.ls);
        catch
          hLine1=semilogy(f,m);
        end
        try, hLine1.Color=opt.color;catch,end
      else
        [hl,hLine1,hLine2]=plotyy(f,m,f,coh,'loglog','semilogx');
        try,hLine1.LineStyle=opt.ls;catch,end
        try, hLine1.Color=opt.color;catch,end
        hLine2.Color=[.6 .6 .6];
      end
    end
%     expo=10^floor(log10(f(end)));
    axis tight
    ax=axis;
%     axis([round(f(1),2,'significant') (round(f(end),2,'significant')/expo+1)*expo ax(3)
    axis([round(f(1)-.49,2,'significant') round(f(end)+.49,2,'significant') ax(3) ax(4)]);
    ax=axis;
  end
  if ~isempty(opt.ax),axis(opt.ax);end
  if ~isempty(opt.title), title(opt.title),else, title('Magnitude'),end
  if ~isempty(coh),Tith=get(hl(1),'Title');Tith.String=[Tith.String ' with coherence'];end
```

```
  if opt.rad, xlabel('w [rad/s]');else, xlabel('f [Hz]');end
  if opt.grid, grid on,end
%   if opt.hold, hold on,end

  if ~opt.magn
    subplot(212)
    if opt.hold, hold on,else hold off,end
    if ~opt.hold,cla;end
    if ~opt.loglog
      hl=plot(f,p,opt.ls);
      set(hl(1),'MarkerSize',4);
      try set(hl(1),'Color',opt.color);catch,end
    else
      hl=semilogx(f,p,opt.ls);
      set(hl(1),'MarkerSize',4);
      try set(hl(1),'Color',opt.color);catch,end
      ax0=axis;
      axis([ax(1) ax(2) ax0(3) ax0(4)]);
    end
%     v=axis;
%     axis([v(1) v(2) -200 200]);
    axis([ax(1) ax(2) -200 200]);
    title('Phase')
    if opt.rad, xlabel('w [rad/s]');else, xlabel('f [Hz]');end
    if opt.grid, grid on,end
    if opt.hold, hold on,end
  end
end
```

D.5 *readpunch.m*

This script is used for reading the .pch-files (punch files) that contains the mode shopes with corresponding eigenfrequency from a Nastran SOL103 simulation.

```
function D=readpunch(punchfile,opts)
%% READPUNCH   Reads a NASTRAN punch file
%Inputs:   punchfile  - Name of punch file
%          opts       - Optional settings
%                       opts.verbose=0 gives quite mode
%                       opts.verbose=1 gives verbose mode (default)
%                       opts.verbose=2 gives very verbose mode
%                       opts.uniform=1 for uniform data gives data in 3D matrix
%                       opts.uniform=0 gives data in cell structure (default)
%                       opts.extseout=1 reads SuperElement data put on the
%                                       punch file by the EXTSEOUT command
%                       opts.clipboard=1 Used clipboard to speed up
%Output:   D          - Data object
%                         D.Header  - Header to data set
%                         D.Data    - Data set
%
%Call:     D=readpunch(punchfile,opts)

%Written:   2011-12-23, Thomas Abrahamsson
%Modified:  2012-08-29, /TA, Problem with D.GRID
%Corrected: 2013-04-17, Corrected bug for reading GRID /TA
%Modified:  2013-04-17, Now reads both long and short format for GRID /TA


%%                                                          Initiate and check
fid=fopen(punchfile);
if fid==-1, error(['READPUNCH cannot find file: ' punchfile]);end
if nargin<2,opts=[];end
if ~isfield(opts,'verbose'), opts.verbose=1;end
if ~isfield(opts,'uniform'), opts.uniform=false;end
if ~isfield(opts,'extseout'), opts.extseout=false;end
if ~isfield(opts,'clipboard'), opts.clipboard=false;end

if opts.clipboard,par='no';else,par='yes';end

if opts.extseout
    D=readpunch_se(fid,punchfile,opts,par);
else
    D=readpunch_od(fid,punchfile,opts,par);
end

%%                                                                  Terminate
fclose(fid);


% ==========================================================================
function D=readpunch_od(fid,punchfile,opts,par)

KnownTypes={'ACCELERATION' 'DISPLACEMENTS' 'VELOCITY' 'EIGENVECTOR' ...
                                                    'OLOADS' 'SPCF'};
```

```
Ch10=char(10);

%%                                                    Make room for much data
Dir=dir(punchfile);VeryLongLine=char(zeros(1,Dir.bytes));
Header=char(zeros(1,1e5));

%%                                                          Read punch file
SubCase=0;
while 1
  txt=fgets(fid,82), if ~ischar(txt),break,end
  if txt(1)=='$',                                        % Read case header
    SubCase=SubCase+1;EOH=0;
    if SubCase>1;% Spool data
        if strcmp(par, 'yes')
            tmp_nam=tempname;
            fid2=fopen(tmp_nam,'w+');
            fprintf(fid2,VeryLongLine(1:EOL));
            fclose(fid2);
            if opts.uniform
                D.Data(:,:,SubCase-1)=importdata(tmp_nam);
                delete(tmp_nam);
            else
                D.Data{SubCase-1}=importdata(tmp_nam);
                delete(tmp_nam);
            end
        else if strcmp(par, 'no')
                clipboard('copy',VeryLongLine(1:EOL));
                if opts.uniform
                    D.Data(:,:,SubCase-1)=importdata('-pastespecial');
                else
                    D.Data{SubCase-1}=importdata('-pastespecial');
                end
            end
        end
    end
    Line=1;
    while txt(1)=='$'
       if Line==4
         Type=deblank(txt(2:72));
         if ~any(strcmp(Type,KnownTypes))
           error(['Unknown type: ' Type '. Modify READPUNCH!'])
         end
       end
       Header(EOH+1:EOH+72)=[txt(2:72) Ch10];EOH=EOH+72;
       txt=fgets(fid,82);
       Line=Line+1;
    end
    D.Header{SubCase}=Header(1:EOH);
    if ispc
      fseek(fid,-82,'cof');% Rewind one line
    elseif isunix
      fseek(fid,-81,'cof');% Rewind one line
    end
    if opts.verbose>1, disp(Header(1:EOH)),end
    EOL=0;
  else                                                    % Read case data
```

```
      if txt(1)~='-'
        VeryLongLine(EOL+1:EOL+71)=[Ch10 txt([1:16 19:72])];EOL=EOL+71;
      else
        VeryLongLine(EOL+1:EOL+54)=txt(19:72);EOL=EOL+54;
      end
  end
end

%%                                                              Last data set
if strcmp(par, 'no')
clipboard('copy',VeryLongLine(1:EOL));
if opts.uniform
  D.Data(:,:,SubCase)=importdata('-pastespecial');
else
  D.Data{SubCase}=importdata('-pastespecial');
end
else if strcmp(par, 'yes')
        tmp_nam=tempname;
        fid2=fopen(tmp_nam,'w+');
        fprintf(fid2,VeryLongLine(1:EOL));
        fclose(fid2);
        if opts.uniform
            D.Data(:,:,SubCase)=importdata(tmp_nam);delete(tmp_nam)
        else
            D.Data{SubCase}=importdata(tmp_nam);delete(tmp_nam)
        end
    end
end


% ========================================================================
function D=readpunch_se(fid,punchfile,opts,par)

%%                                                  Make room for much data
VeryLongLine=char(zeros(1,1e5));
Header=char(zeros(1,1e5));
Ch10=char(10);



while 1,%% Spool initial comments
  txt=fgets(fid,82); if ~ischar(txt),break,end
  if txt(1)~='$',break,end
end

if strcmp(txt(1:11),'BEGIN SUPER')
    D.SEID=str2num(txt(12:end));
else
    error('Cannot understand EXTSEOUT data')
end

while 1
  txt=fgets(fid,82);
  hdr=deblank(txt(1:min([end 8])));hdr(find(hdr==' '))=[];
  if hdr(1)=='$',hdr='$';end
  switch hdr
```

```
case 'GRID'
  rewindline(fid,txt);
  EOL=0;
  while 1,%% Read GRID DATA
    txt=fgets(fid,82); if ~ischar(txt),break,end
    if txt(1)=='$'
      break
    else
      Str=[txt(9:24) ' ' txt(25:32) ' ' txt(33:40) ' ' txt(41:48) Ch10];
      VeryLongLine(EOL+1:EOL+length(Str))=Str;EOL=EOL+length(Str);
    end
  end
  if strcmp(par, 'no')
    clipboard('copy',VeryLongLine(1:EOL));
    GRIDMat=importdata('-pastespecial');
    else if strcmp(par, 'yes')
      tmp_nam=tempname;
      fid2=fopen(tmp_nam,'w+');
      fprintf(fid2,VeryLongLine(1:EOL));
      fclose(fid2);
      GRIDMat=importdata(tmp_nam);delete(tmp_nam)
    end
  end
  D.GRID=GRIDMat;

case 'GRID*'
  rewindline(fid,txt);
  EOL=0;
  while 1,%% Read GRID DATA
    txt=fgets(fid,82); if ~ischar(txt),break,end
    if txt(1)=='$'
      break
    else
      Str=[txt(9:24) ' ' txt(25:40) ' ' txt(41:56) ' ' txt(57:72) ' '];
      txt=fgets(fid,82); if ~ischar(txt),break,end
      Str=[Str txt(9:24) Ch10];
      VeryLongLine(EOL+1:EOL+length(Str))=Str;EOL=EOL+length(Str);
    end
  end
  if strcmp(par, 'no')
    clipboard('copy',VeryLongLine(1:EOL));
    GRIDMat=importdata('-pastespecial');
    else if strcmp(par, 'yes')
      tmp_nam=tempname;
      fid2=fopen(tmp_nam,'w+');
      fprintf(fid2,VeryLongLine(1:EOL));
      fclose(fid2);
      GRIDMat=importdata(tmp_nam);delete(tmp_nam)
    end
  end
  D.GRID=GRIDMat;

case 'CORD2C'
  rewindline(fid,txt);
  EOL=0;
  while 1,%% Read CORD2C DATA
```

```
    txt=fgets(fid,82); if ~ischar(txt),break,end
    if txt(1)=='$'
      break
    else
      Str=[txt(9:24) ' ' txt(25:40) ' ' txt(41:56) ' ' txt(57:72) Ch10];
      VeryLongLine(EOL+1:EOL+68)=Str;EOL=EOL+68;
    end
  end
  if strcmp(par, 'no')
    clipboard('copy',VeryLongLine(1:EOL));
    CORD2CMat=importdata('-pastespecial');
    else if strcmp(par, 'yes')
      tmp_nam=tempname;
      fid2=fopen(tmp_nam,'w+');
      fprintf(fid2,VeryLongLine(1:EOL));
      fclose(fid2);
      CORD2CMat=importdata(tmp_nam);delete(tmp_nam)
    end
  end
  D.CORD2C=CORD2CMat;

case 'CORD2R'
  rewindline(fid,txt);
  EOL=0;
  while 1,%% Read CORD2R DATA
    txt=fgets(fid,82); if ~ischar(txt),break,end
    if txt(1)=='$'
      break
    else
      Str=[txt(9:24) ' ' txt(25:40) ' ' txt(41:56) ' ' txt(57:72) Ch10];
      VeryLongLine(EOL+1:EOL+68)=Str;EOL=EOL+68;
    end
  end
  if strcmp(par, 'no')
    clipboard('copy',VeryLongLine(1:EOL));
    CORD2RMat=importdata('-pastespecial');
    else if strcmp(par, 'yes')
      tmp_nam=tempname;
      fid2=fopen(tmp_nam,'w+');
      fprintf(fid2,VeryLongLine(1:EOL));
      fclose(fid2);
      CORD2RMat=importdata(tmp_nam);delete(tmp_nam)
    end
  end
  D.CORD2R=CORD2RMat;

case 'ASET'
  rewindline(fid,txt);
  EOL=0;
  while 1,%% Read ASET DATA
    txt=fgets(fid,82); if ~ischar(txt),break,end
    if txt(1)=='$'
      break
    else
      for I=24:16:length(txt)
        VeryLongLine(EOL+1:EOL+18)=[txt(I-15:I-8) ' ' txt(I-7:I) Ch10];EOL=EOL+18;
```

```
        end
      end
    end
    if strcmp(par, 'no')
      clipboard('copy',VeryLongLine(1:EOL));
      D.ASET=importdata('-pastespecial');
    else
      if strcmp(par, 'yes')
        tmp_nam=tempname;
        fid2=fopen(tmp_nam,'w+');
        fprintf(fid2,VeryLongLine(1:EOL));
        fclose(fid2);
        D.ASET=importdata(tmp_nam);delete(tmp_nam)
      end
    end
  case 'ASET1'
    dof1=str2num(txt(17:24));thru=txt(29:32);dof2=str2num(txt(33:40));
    if strcmp(thru,'THRU')
      D.ASET1=dof1:dof2;
    else
      error('Does not recognize this type of ASET1 data');
    end
  case 'SPOINT'
    rewindline(fid,txt);
    EOL=0;
    while 1,%% Read SPOINT DATA
      txt=fgets(fid,82); if ~ischar(txt),break,end
      if txt(1)=='$'
        break
      else
        for I=24:16:length(txt)
          VeryLongLine(EOL+1:EOL+18)=[txt(I-15:I-8) ' ' txt(I-7:I) Ch10];EOL=EOL+18;
        end
      end
    end
%     disp('Did not store SPOINT data')
  case '$'
%       Do nothing, it's just a comment line
  case 'DMIG'
    break;% Break the loop and read DMID entries
  otherwise
    error(['Do not understand: ' hdr])
  end
end


dofind=[];
for I=1:size(D.ASET,1)
  dofind=[dofind (D.ASET(I,1)-1)*6+abs(int2str(D.ASET(I,2)))-48];
end
try dofind=[dofind(:);D.ASET1(:)];catch,end

% Now the pointer should be at the DMIG card
rewindline(fid,txt);

while 1,
```

```
  txt=fgets(fid,82); if ~ischar(txt),break,end
  if ~strcmp(txt(1:4),'DMIG'),break,end
  DMIGtype=txt(9:16);
  IFO=str2double(txt(25:32));TIN=str2double(txt(33:40));NCOL=str2double(txt(65:72));
  if opts.verbose>0
      hwb=waitbar(0,['Reading ' deblank(DMIGtype) ' data ...'],'Name','Reads DMIG data fro
      ticdelta=NCOL/10;nexttic=ticdelta;
  end
  if opts.verbose>1
      disp(['Reading ' deblank(DMIGtype) ' of column size ' int2str(NCOL)]);
  end

  if IFO==6 && NCOL~=0; %IF symmetric (IFO=6) and size known
      M=spalloc(NCOL,NCOL,round(0.02*NCOL*NCOL));
  end
  while 1
     txt=fgets(fid,82); if ~ischar(txt),break,end
     if ~strcmp(txt(1:4),'DMIG'),break,end
     GJ=str2double(txt(33:40));CJ=str2double(txt(49:56));dof=(GJ-1)*6+CJ;
     if CJ~=0
         cdofind=find(dofind==dof);
     else;%CJ=0 for scalar
         cdofind=find(dofind==GJ);
         if isempty(cdofind),cdofind=1;end;%Undefined scalar
     end
     if opts.verbose>0
         if cdofind>nexttic
            waitbar(cdofind/NCOL,hwb);nexttic=nexttic+ticdelta;
         end
     end
     EOL=0;
     while 1
       txt=fgets(fid,82); if ~ischar(txt),break,end
       if txt(1)~='*'
         break
       else
         Str=[txt(17:24) ' ' txt(33:40) ' ' txt(41:56) Ch10];
         VeryLongLine(EOL+1:EOL+35)=Str;EOL=EOL+35;
       end
     end
     VeryLongLine(find(VeryLongLine=='D'))='E';
     if strcmp(par, 'no')
        clipboard('copy',VeryLongLine(1:EOL));
        Data=importdata('-pastespecial');
    else if strcmp(par, 'yes')
            tmp_nam=tempname;
            fid2=fopen(tmp_nam,'w+');
            fprintf(fid2,VeryLongLine(1:EOL));
            fclose(fid2);
            Data=importdata(tmp_nam);delete(tmp_nam);
        end
     end
%      clipboard('copy',VeryLongLine(1:EOL));
%      Data=importdata('-pastespecial');
     GI=Data(:,1);CI=Data(:,2);dof=(GI-1)*6+CI;
     rdofind=[];
```

```
      for I=1:length(GI)
        if CI(I)~=0
          rdofind(I)=find(dofind==dof(I));
        else;%CJ=0 for scalar
          rdofind(I)=find(dofind==GI(I));
        end
      end
%      rdofind=find(ismember(dofind,dof));
      M(rdofind,cdofind)=Data(:,3);
      if ispc
        fseek(fid,-length(txt),'cof');% Rewind one line
      elseif isunix
        fseek(fid,-length(txt)+1,'cof');% Rewind one line
      end
      if ~strcmp(DMIGtype,txt(9:16));% Store data and break
          if IFO==6
              eval(['D.' DMIGtype '=M+M''-diag(diag(M));']);
          else
              eval(['D.' DMIGtype '=M;']);
          end
          clear('M');
          if opts.verbose>0,close(hwb);end
          break
      end
  end
  if ~strcmp(txt(1:4),'DMIG'),break,end
end

if opts.verbose>1
    disp('Warning: Data after last DMIG data set in contiguous order of Punch File may be
end



function rewindline(fid,txt)
% Rewind one line
if ispc
    fseek(fid,-length(txt),'cof');
elseif isunix
    fseek(fid,-length(txt)+1,'cof');
end
```

D.6 *aset2dofno.m*

Matlab script for extracting degrees of freedom in Nastran model at specific grid points.

```
function adofs=aset2dofno(f06filename,gridlist)
%ASET2DOFNO Creates list of adof indices associated to grid point dof list
%         It reads a Nastran f06 file that needs to be prepared beforehand.
%         Place the two PARAM lines given below in the bulk data section of
%         your Nastran input file to create that Nastran f06 file
%           PARAM,USETPRT,12
%           PARAM,CHECKOUT,YES
%
%Inputs: f06filename - The filename of your Nastran f06 file
%        gridlist    - A list (vector) of grid point and dof indices as
%                      grid.dof, e.g. gridlist=[11.3 200.1] for 3rd dof
%                      of 11th gridpoint and 1st dof of 200th gridpoint
%Output: adofs       - The dof indices of adofs associated with gridlist
%Call:   adofs=aset2dofno(f06filename,gridlist)

%Copyleft: Thomas Abrahamsson, Chalmers University of Technology, Sweden
%Written: 2015-09-25


fid=fopen(f06filename);

testlin='U S E T  D E F I N I T I O N  T A B L E  ( E X T E R N A L  S E Q U E N C E ,
R O W  S O R T )';
adoflin='A        DISPLACEMENT SET';
fdoflin='F        DISPLACEMENT SET';

usetfound=false;
asetfound=false;
Gdofs=[];
while 1,
  lin=fgetl(fid);
  if lin==-1,break,end
  if length(lin)>116
    if ~any(lin(17:117)-testlin),usetfound=true;end
  end
  if usetfound
    if length(lin)>79
      if ~any(lin(56:80)-adoflin),asetfound=true;end
      if ~any(lin(56:80)-fdoflin),asetfound=false;end
    end
    if asetfound
      ind=findstr(lin,'=');
      if length(ind)>1
        str=lin(10:122);
        str=strrep(str,'-','.');
        Gdofs=[Gdofs;str2num(str)'];
      end
    end
  end
end
fclose(fid);
```

```
[~,~,adofs]=intersect(gridlist,Gdofs,'stable');
```

D.7 *mac.m*

Matlab script to compute the MAC-matrix using the mode matrices.

```
function MAC=mac(Z1,Z2,opt,depthZ2)
%MAC: Computes the MAC-matrix using the mode matrices Z1 and Z2
%Inputs:   Z1     - First modal matrix containing modes as column vectors
%          Z2     - Second modal matrix containing modes as column vectors. If
%                   depthZ2 > 1 the groups of modes are stoved on top of each
%                   other
%          opt    - 1 gives standard MAC calculation (default if depthZ2=1)
%                   2 gives cosines of subspace angles
%Output:   MAC    - MAC-matrix
%Call:     MAC=mac(Z1,Z2,opt,depthZ2)

% Copyleft: Thomas Abrahamsson, Linkping, Sweden
% Written:  Oct    4, 1993
% Modified: April 16, 1994 /TA
% Modified: June  14, 1995
% Modified: June  19, 1995
% Modified: Nov   11, 2001 (real added to ensure that MAC is real also for
%                           complex vectors)
% Modified: Dec   12, 2013 Removed bottleneck for set opt and depthZ2
%                          and made MAC computations more efficient /TA

% ------------------------------------------------------------------------
%                                                          Initiate and check
%                                                          ------------------
tol=1.e6*eps;
% if ~exist('opt'),opt=1;elseif (isempty(opt) & nargin<4),opt=1;end
% if ~exist('depthZ2'),depthZ2=1;elseif isempty(depthZ2),depthZ2=1;end
if nargin<2,
  error('Too few input arguments to MAC')
elseif nargin<3
  opt=1;depthZ2=1;
elseif nargin<4
  depthZ2=1;
end

[n1,m1]=size(Z1);[n2,m2]=size(Z2);n2=n2/depthZ2;
if n1~=n2,
  error('The row dimension of the two mode matrices must be the same')
end
if (depthZ2~=1 && opt==1),
  error('Error in MAC: opt=1 is not allowed when depthZ2>1')
end

if opt==1,
% ------------------------------------------------------------------------
%                                                              Standard MAC
%                                                              ------------
MAC=Z2'*Z1./(sqrt(diag(Z2'*Z2))*sqrt(diag(Z1'*Z1))');
MAC=(MAC').'.*MAC;

% Old, less efficient code
```

```
%    for I=1:m1,
%     for II=1:m2,
%       MAC(II,I)=real((Z1(:,I)'*Z2(:,II))*conj(Z1(:,I)'*Z2(:,II)))/ ...
%                 norm(Z1(:,I))^2/norm(Z2(:,II))^2;
%     end
%    end

else
% ------------------------------------------------------------------------
%                                                 MAC using square of cosine of
%                                                 subspace angle
%                                                 ----------------------------
  for I=1:m1,
    for II=1:m2,
%      MAC(II,I)=cos(subspac2(Z1(:,I),reshape(Z2(:,II),n2,depthZ2),tol));
      MAC(II,I)=(cos(subspac2(Z1(:,I),reshape(Z2(:,II),n2,depthZ2),tol)))^2;
    end
  end
end

function theta = subspac2(A,B,tol)
%SUBSPAC2 Angle between two subspaces.
%  SUBSPAC2(A,B,tol) finds the angle between two subspaces specified by the
%  columns of A and B.  If A and B are vectors of unit length, this is the
%  same as ACOS(A'*B).
%  If the angle is small, the two spaces are nearly linearly dependent. The
%  rank of A and B will be checked using: rank(A,tol) and rank(B,tol).  Rank
%  deficient matrices will be substituted will lower column order matrices
%  of full rank.
%Inputs: A,B   - Subspace matrices
%        tol   - Tolerance parameter for rank determination
%Output: theta - Angle between subspaces
%Call:   theta=subspac2(A,B,tol)
%See also: SUBSPACE

%Copyright: Thomas Abrahamsson, Saab Military Aircraft, Linkoping, Sweden
%Written:  June  19, 1995
%Modified: June  27, 1995

% ------------------------------------------------------------------------
%                                                       Initiate and check
%                                                       ------------------
if nargin<3,tol=eps;elseif isempty(tol),tol=eps;end
[na,ma]=size(A);[nb,mb]=size(B);
if na ~= nb
   error('Row dimensions of A and B must be the same.')
end

% ------------------------------------------------------------------------
%                           Calculate rank and create non-rank-deficient matrices
%                           -----------------------------------------------------
[U,S,V] = svd(A);U=U(:,1:ma);S=S(1:ma,1:ma);
if na ~= 1, s = diag(S); else, s = S(1,1); end
r = sum(s > tol);
A = U(:,1:r);
```

```
[U,S,V] = svd(B);U=U(:,1:mb);S=S(1:mb,1:mb);
if nb ~= 1, s = diag(S); else, s = S(1,1); end
r = sum(s > tol);
B = U(:,1:r);

[na,ma]=size(A);[nb,mb]=size(B);

% --------------------------------------------------------------------------
%                                        Compute the angle between the subspaces
%                                        ---------------------------------------
[QA,ignore] = qr(A);QA=QA(:,1:ma);
[QB,ignore] = qr(B);QB=QB(:,1:mb);
s = svd(QA'*QB);
% The max singular value is the correct one to choose
% but should have magnitude no more than 1.
theta = acos(min(min(s),1));
```

D.8 *matplot.m*

Matlab script used to give a graphical representation of a matrix, for example a MAC.

```
function matplot(M,Range,XTickLabels,YTickLabels)
%MATPLOT: Shows a graphical representation of a matrix
%Inputs: M     - The matrix to plot
%        Range - The range of data values, defaults [min(M(:) max(M(:))]
%                Range cannot be smaller than default
%Call:   matplot(M,Range,XTickLabel,YTicLabel)

%Written: 2011-09-01 by Thomas Abrahamsson
%Modified: 2014-05-19, Took care of case when "matrix" is scalar /TA
%Modified: 2015-03-31, Introduced labels /TA

%%                                                                  Initiate
if nargin<2, Range=[min(M(:)) max(M(:))];end
if isempty(Range),Range=[min(M(:)) max(M(:))];end
if nargin<3, XTickLabels=[];end
if nargin<4, YTickLabels=[];end

[r c]=size(M);
X=[];Y=[];Z=[];
for I=1:r
  for J=1:c
    X=[X,[0;1;1;0]+(J-1)];
    Y=[Y,[0;0;1;1]+(I-1)];
    Z=[Z M(I,J)];
  end
  X=[X,[0;eps;eps;0]];Y=[Y,[0;0;eps;eps]];Z=[Z Range(1)];
  X=[X,[0;eps;eps;0]];Y=[Y,[0;0;eps;eps]];Z=[Z Range(2)];
end

% cm=flipud(colormap('gray'));
% cm=colormap(graymap('5',256));
cm=flipud(colormap('pink'));

% hF=figure;set(hF,'Renderer','OpenGL'); x
if r*c==1,Z=Z(1);end
hP=patch(X,Y,Z);colormap(cm);
% axis off
axis equal
ax=gca;
try
  set(ax,'Box','on','Color',.94*[1 1 1],'TickLength',[0 0],'XTickLabelRotation',90);
catch
end
xt=get(ax,'Xtick');yt=get(ax,'Ytick');
set(ax,'XTick',xt(1):.5:xt(end));
set(ax,'YTick',yt(1):.5:yt(end));
xt=get(ax,'Xtick');yt=get(ax,'Ytick');

J=0;
for I=1:length(xt)
  if xt(I)>0 & xt(I)<c & floor(I/2)*2==I
```

```
    J=J+1;
    indx(J)=I;
  end
end
J=0;
for I=1:length(yt)
  if yt(I)>0 & yt(I)<r & floor(I/2)*2==I
    J=J+1;
    indy(J)=I;
  end
end

if length(XTickLabels)==length(indx)
  J=0;
  for I=1:length(xt)
    if any(indx==I)
      J=J+1;
      XLab{I}=XTickLabels{J};
    else
      XLab{I}='';
    end
  end
else
  J=0;
  for I=1:length(xt)
    if any(indx==I)
      J=J+1;
      XLab{I}=int2str(J);
    else
      XLab{I}='';
    end
  end
end
if length(YTickLabels)==length(indy)
  J=0;
  for I=1:length(yt)
    if any(indy==I)
      J=J+1;
      YLab{I}=YTickLabels{J};
    else
      YLab{I}='';
    end
  end
else
  J=0;
  for I=1:length(yt)
    if any(indy==I)
      J=J+1;
      YLab{I}=int2str(J);
    else
      YLab{I}='';
    end
  end
end

set(ax,'XTickLabel',XLab,'YTickLabel',YLab);
```

```
colorbar
```

E

Python script

Python script used in ANSA to create a normal cordinate systems in a set of grid points.

```
# PYTHON script
import os
import ansa
from ansa import base,mesh


def main():

    set = base.GetEntity(0,"SET",2)
    shells = base.CollectEntities(0,set,"SHELL")

    for shell in shells:
        gridlist =[]
        cords = base.GetNormalVectorOfShell(shell)
        type = base.GetEntityCardValues(0,shell,("type","G1","G2","G3"))
        g1 = base.GetEntity(0,"GRID",type["G1"])

        g1cords = base.GetEntityCardValues(0,g1,("X1","X2","X3"))
        gridlist.append(g1)
        x=float(g1cords["X1"])*float(cords[0])
        y=float(g1cords["X2"])*float(cords[1])
        z=float(g1cords["X3"])*float(cords[2])
        pt = base.Newpoint(float(g1cords["X1"]),float(g1cords["X2"]),...
            float(g1cords["X3"]))
        pt_id = base.GetEntityCardValues(0,pt,("ID",))

        args = (pt,g1)
        base.GeoTranslate("COPY",0,"SAME PART","NONE",float(cords[0]),...
            float(cords[1]),float(cords[2]),args,keep_connectivity=True,...
            draw_results=False)

        pt2 = base.GetEntity(0,"POINT",pt_id["ID"]+1)
        pt_crd = base.GetEntityCardValues(0,pt2,("X","Y","Z"))
        pt_coords = (pt_crd["X"],pt_crd["Y"],pt_crd["Z"])

        tritos = base.NearestNode(pt_coords,2)
        #print(tritos[0]._id)
        #g1cp=base.GetEntity(0,"GRID",nid)
        g2 = base.GetEntity(0,"GRID",type["G2"])
        gridlist.append(g2)
        g3 = base.GetEntity(0,"GRID",type["G3"])
```

```
        gridlist.append(g3)
        trias=[]
        try:
            type2 = base.GetEntityCardValues(0,shell,("type","G4"))
            if type2["G4"]:
                g4 = base.GetEntity(0,"GRID",type2["G4"])
                gridlist.append(g4)
        except KeyError:
            trias.append(shell)

        fields = {"G1":type["G1"],"G2":tritos[0]._id,"G3":type["G2"]}
        cord_sys = base.CreateEntity(0,"CORD_NODES_R",fields)

        for grid in gridlist:
            base.SetEntityCardValues(0,grid,{"CD":cord_sys._id})
            to_del = (pt,pt2)

        base.DeleteEntity(to_del,True)


if __name__ == '__main__':
    main()
```