Machine Learning Based Sensor Selection for Modal Testing

a project presented to
The Faculty of the Department of Aerospace Engineering
San José State University

in partial fulfillment of the requirements for the degree
Master of Science in Aerospace Engineering

by

Todd W Kelmar

March 2023

approved by
Maria Chierichetti
Faculty Advisor

© 2023
Todd W Kelmar
All Rights Reserved

Abstract

Machine Learning Based Sensor Selection for Modal Testing

Todd W Kelmar

Modal testing is a common step in the aerostructure design process and is often conducted to verify natural frequencies and mode shapes predicted by computational techniques. When conducting this testing, sensor placement is crucial to being able to measure the desired natural frequencies. Existing methodologies for sensor placement can be time consuming and require an iterative approach. Using machine learning techniques like those being developed for structural health monitoring may offer a more optimal sensor selection methodology. This work contains a description of three machine learning based sensor selection methodologies and compares their performance with current techniques. Performance is evaluated with a simple 1D numerical model, a more complex 2D model, and finally through physical experimentation. Although not conclusive, initial results are promising and show a random forest regressor based model is at least as effective at estimating natural frequencies as the current effective independence technique.

Acknowledgements

First and foremost, I extend my deepest appreciation to my advisor, Dr. Chierichetti, for guiding me past the horizon into a realm of aerospace waiting to be understood by man and machine. I feel privileged to have worked and studied in her orbit for the past two years.

I would like to extend my appreciation to Dr. Mourtos and the entire faculty at San Jose State University. They are a solar wind, providing a constant push to me and my peers voyages. I am privileged to have journeyed through their starfield of knowledge.

I would also like to thank my undergraduate advisor, Dr. Perucchio, at the University of Rochester. His belief in my abilities as an undergraduate propelled me to strive for excellence in my studies and research. His encouragement and advice played a pivotal role in setting my course to pursue this Master's degree.

I would also like to thank my family, my forever crew mates, for their loyalty and their love.

And to Olivia, my true North.

Table of Contents

Table of Figures

Table of Tables

Table of Symbols

Symbol	Definition	Units
$\widehat{H}$	Response Function	-
k	Stiffness	-
m	mass matrix	-
X	DFT of Response Signal	-
F	DFT of Excitation Signal	-
E	Young's Modulus	GPa
u	Modal Coordinate Vector	-
q	Target Modal Coordinate	-
i	$\sqrt{-1}$	-
h	Response Frequency	Hz
M_{aa}	TAM Mass	
f	Frequency	Hz
Greek Symbols		
ω	Natural Frequency	Hz
ζ	Damping Ratio	-
ν	Poisson Ratio	-
ρ	Density	kg/m^3
Φ	Modal Vector	-
Γ	Eigenvectors of Fisher Information Matrix	-
Λ	Eigenvalues of Fisher Information Matrix	-
η	Structural Damping	-
λ	Eigenvalue	-
Ψ	Displacement Shape	-
Subscripts		
r	Response Signal	-
s	sensed value	-
a	Analysis DOF	-
c	Classic Guyan	-
f	Free DOF	-
Acronyms		
FEA	Finite Element Analysis	-
ML	Machine Learning	-
DOF	Degree of Freedom	
FRF	Frequency Response Function	-
EIM	Effective Independence Methods	-
MEIM	Mass Weighted Effective Independence Method	-
IRKE	Iterative Residual Kinetic Energy	-
RKE	Residual Kinetic Energy	-
SISO	Single Input Single Output	-
SIMO	Single Input Multiple Output	-
SDOF	Single Degree of Freedom	-
MDOF	Multiple Degree of Freedom	-
DFT	Discrete Fourier Transform	-
LSCE	Least Square Complex Exponential	-
LSRF	Least Square Rational Function	
EFI	Effective Independence	-
TAM	Test Analysis Matrix	-

1 Introduction

1.1 Motivation

Whether looking at pogo oscillation on a rocket or ground resonance in a helicopter, vibration analysis is crucial when designing aerospace structures. In industry, modal analysis typically begins with computational methods such as finite element analysis (FEA or FE) which are then followed by experimental vibration testing [1,2]. Real world analysis is constrained by the need to use a finite number of sensors and exciters to determine the modal response of a given system. In experimental setup, the location of the excitation(s) and the location of the sensor(s) can have dramatic impacts on observed modal response of the structure [3]. These errors can lead to mischaracterizations of the system and as such the placement of both sensors and excitation must be carefully determined [4].

Existing methodology for sensor placement typically relies on developed knowledge and established best practices. When developing novel aerostructures where best practices cannot be relied on, multiple rounds of testing may be required with different sensor placements to determine the most effective experimental setup [5]. This process is both time consuming and labor intensive and may still not result in an optimal configuration [5].

Finding patterns within large datasets is an ideal application for machine learning and may yield more optimal configurations with fewer sensors than traditional approaches. Machine learning (ML) is not magic however, and care must be taken to ensure that optimal sensor placement in one case can be replicated reliably. Additionally, the ML model is only as accurate as the starting FE model, which must be verified through additional testing for more complex geometries. While there are established methodologies for measuring the success of machine learning models, there is some understandable apprehension about using results that were derived through a nontransparent function. Therefore, the ML derived sensor placement will be compared against current best practices.

1.2 Literature Review

1.2.1 Overview

The following literature review is divided into three main sections for the sake of clarity. The first subsection describes the process and uses of modal testing as it relates to structural dynamics in general and aerostructures in particular. In the second section, an overview is provided of the current state of the art in determining sensor placement for modal testing. The third section describes current trends and techniques in the application of machine learning structural dynamics. As the field ML is rapidly evolving, this section attempts to provide a general picture of current methodologies and techniques. The specific techniques to be used in this paper will be described in more detail in the relevant section(s).

1.2.1.1 Modal Testing and Analysis

All structures are subject to vibrations either internal, such as engine vibration, external such as turbulence, or a combination of both. Characterizing the behavior of a system under vibration or other dynamic forces is crucial to good engineering design [1,2]. The main objective of most modal testing is to determine the natural frequencies and vibrational modes of the structure being tested[1,2]. While the accessibility and speed of modern computers and FEA solvers may seem to obviate the need for physical modal testing, the results are only as accurate as the model being tested [5]. The results of modal testing can be compared to those of the theoretical model and used establish if the model accurately describes the structure being analyzed [1]. Additional uses of modal testing include creating mathematical models of structures for integration into other analyses or developing models for structural health monitoring [1].

Modal analysis can be conducted on data acquired in laboratory conditions or from data acquired while the structure is in regular use [6]. This work concentrates on the former, as laboratory testing allows the inputs to be more precisely controlled and allows for experimental variables to be controlled to a greater extent. In modal testing, sensors placed on the structure being tested—typically accelerometers and/or strain gauges—are measured to record their response to an excitation. The input can be provided by a modal shaker—a device which takes a signal as an input and applies that signal to the structure under test or a modal hammer, where an

impact is made against the structure to represent an instantaneous excitation [6]. In more complex tests or for large structures, multiple modal shakes may be used to induce a measurable excitation in the structure [1]. The output of the sensors are then post processed, with the exact methodology dependent on the excitation signal [1]. These data may then be analyzed with a frequency response function in order to derive the natural frequencies and mode shapes of the structure under test [1].

Placing the sensors on the structure must be done with care, as it can substantially influence the results of a modal test. The goal of optimizing sensor placement is to determine the most information about the structure's behavior while minimizing the required number of sensors [7]. Large numbers of sensors increase the cost of testing in both equipment and labor required to set up the test. As modal testing is principally concerned with structural dynamics, an ideal sensor selection would result in the least number of sensors where each sensor's individual contribution to the analysis is greatest [7].

1.2.2 Current Methodologies in Sensor Selection

Early modal testing relied on engineering judgement and institutional knowledge derived from fundamentals of vibration [8]. While this may still be used in certain situations, such as with a well understood structure or for simple geometries, novel structures present difficulties for this approach [1]. Additionally, tight timelines due to budget constraints or limited access to testing facilities reduce the time available to refine sensor placement during testing [5]. As such, determining efficient methodology for sensor placement has real implications for increasing efficiency. As a result, several methodologies have been developed to assist engineers in determining appropriate sensor placement for modal testing and structural health monitoring. Some of the current methodologies include the Effective Independence Method (EIM) and Iterative Residual Kinetic Energy approach (IRKE) [9]. Other techniques, such as those using information entropy, have also been developed [10]. A brief overview of these techniques will be provided below. Detailed methodology for each technique will be provided in later sections.

1.2.2.1 Effective Independence Method (EIM)

The Effective Independence Method, also known as the effective independence algorithm, begins with a set of target mode shapes that encompass the set of candidate sensor

locations derived from experimental results or an FE model [11]. The algorithm attempts to predict the independence of each node based on the expected measured mode shape, with higher values indicating increased independence [11]. In this method, it is required to know both the expected mode shape as well as have candidate sensor locations located. These candidate locations are then ranked according to the algorithm, removing the lowest ranking sensor and recalculating [11]. As potential locations are eliminated, the relative independence of the remaining solutions increases, and the process is repeated until the required number of sensor locations is reached [11]. This methodology does not assist in determining the number of sensor or the initial placements of those sensors [12]. Were the structure divided into a coarse enough grid, it is theoretically possible to iterate through all possible sensor locations. When operating this way, some research suggests that more optimal results may be produced compared to kinetic energy methods, although at the cost of less ability to measure unexpected modes [3,11].

1.2.2.2 Mass Weighted Effective Independence (MEIM)

A drawback to non-mass-weighted EIM is that it selects only for linear independence of the points and neglects orthogonality [13]. When MEIM is used, modes that contribute least to self-orthogonality are removed in each iteration as opposed to purely focusing on linear independence when selecting features [13]. Cross-orthogonality checks are used to determine how analytical and empirical modal testing results correlate [13]. There are several different methods for calculating the decomposition of the mass matrix needed for this process. Using the Guyan reduced mass matrix appears to result in the best performance computationally as well as producing the most optimized output with respect to other mass weighting techniques [13].

1.2.2.3 Residual Kinetic Energy Method (RKE)

The RKE method, which also has several sub methods based on it, is a technique used by NASA to determine sensor placement for modal testing based on detailed FEA models [14]. The principle concept is to ensure residual kinetic energy is minimized in all degrees of freedom and modes under consideration [15]. When this is computed, DOFs with high residual kinetic energy indicate that additional refinement is needed in order to measure the corresponding degree of freedom in a given mode [15]. After another sensor is added to cover that degree of freedom, the residual kinetic energy is recomputed [15]. This process is repeated until the solution is suitably

orthogonal. This methodology works well when applied to existing analysis points to identify additional DOFs that are undermeasured by the initial sensor placement, and has been adopted by NASA and others to meet NADA and Department of Defense standards for modal testing [16].

1.2.3 Machine Learning Feature Selection

Machine learning (ML) techniques are a promising approach for determining sensor placement in modal analysis. In supervised machine learning, an input dataset is provided consisting of both the input data and the output. In this case, the input would be data derived from a finite element model and the output would be the mode shapes and natural frequencies. Based on this information, the model is then trained to be able to predict outputs based on new input data. As many of the previously discussed methods for sensor placement are iterative approaches, the problem of solving for sensor placement seems to be one to which machine learning is well suited [17].

There are a wide range of different machine learning techniques available, although not all are suitable for all tasks. Most supervised learning algorithms rely on large datasets to allow correlation of inputs to outputs. Typically more inputs require larger datasets to ensure that the data is not overfitted [17,18][18]. Based on prior work done in the field of structural health monitoring, the focus will be on neural networks.

One of the limitations of neural networks is the requirement for a large amount of data for training and testing the network. In this case, with an input derived from a finite element model, the data acquisition can also be time consuming and computationally expensive. Introducing physical laws into the algorithm of a neural network can help reduce the needed size of the training dataset and improve overall performance compared to a traditional neural network technique [18].

In the field of structural health monitoring, research has been conducted to use neural networks, trained on FEA models, to predict stresses and strains on a structure from a finite number of sensor inputs [19]. Additional work has also been done using neural networks to extract mode shapes and natural frequencies from data gathered when the input signal is unknown (output-only modal testing) [20].

1.3 Objective

The objective of this project is to develop sensor placement methodology for modal analysis using machine learning and finite element analysis and compare the results to those obtained using traditional sensor placement techniques. Prior work in machine learning as applied to structural health monitoring and feature selection for stress prediction makes modal testing sensor selection a promising avenue of research. The ideal outcome of the project is the development of a methodology for sensor selection using machine learning that can be applied to arbitrary structures and produce results equivalent with or better than current methodologies. The problem of sensor selection is by no means trivial, this project may reveal information about applying ML to the problem of modal testing sensor selection even without developing a general methodology.

1.4 Methodology

The proposed methodology can be split into two primary parts. The first part will consist of initial testing of machine learning techniques for feature selection on a simple geometry with established best practice sensor placement. This is likely to be an iterative process as different methodologies are explored and tested. Once an established methodology or methodologies have been developed, they will be applied to an aircraft wing for comparison with existing sensor selection techniques. The anticipated steps of the project are outlined below:

1. ML Model Development & Preliminary Verification
    a. Conduct a modal analysis of a 2D Beam using FEA
    b. Determine which features should be selected for by the ML model
    c. Develop ML model using training data derived from the 2D Beam and verify performance against the test dataset.
2. Complex Geometry Verification
    a. Conduct a modal analysis of a 2D plate modeled in FEA.
    b. Train ML model on new geometry using methodology developed in step 1
    c. Compute sensor placement based on current best practices
    d. Conduct physical modal testing with the different sensor placements

2 Modal Testing Theory

2.1 Chapter Summary

Modal testing is the process of determining the natural frequencies, damping ratios, and mode shapes of a given structure through physical experimentation. This chapter presents a brief overview of the theoretical basis and key principles of modal testing as will be applied to this project. For theoretical modal analysis, the mass, stiffness, and damping matricies are used to create a spatial model [1]. This spatial model allows the modal model, which describes the natural frequencies, damping ratios, and mode shapes, to be computed [1]. From the modal model, the response model of the structure can then be determined [1]. For experimental modal testing, the inverse process must be applied. The response model is recorded from sensors on the structure, and experimental modal analysis allows the modal model of the structure to be estimated [1]. From the modal model, if needed, a spatial model of the system can then be calculated. In this paper, the focus will be on improving the estimation of the modal model from the measured response and therefore the primary area of interest is the process of experimental modal analysis.

In modal testing, an instrumented structure has an excitation applied to it, and the excitation and response of the structure is measured and recorded. The process of modal testing typically involves [1]:

1. selecting sensor types and locations
2. selecting an appropriate excitation or excitations
3. exciting the structure in one or multiple places
4. Measuring the response of the structure
5. Processing the recorded response
6. Analyzing the response to extract the desired information.

As the goal of this paper is to develop a novel methodology for sensor placement, discussion of step 1 will be reserved for Chapter 4. Additionally, this chapter will focus on Single Input Multiple Output (SIMO) Multiple Degree of Freedom (MDOF) systems. MDOF systems were chosen as the process of sensor selection on a single degree of freedom system is trivial, and SIMO measurement was chosen due to constraints in available testing equipment. The model applied here for modal testing assuming linear behavior of the structure.

2.2 Frequency Response Function

The frequency response function, or FRF, is a function that represents the response of the system based on the modal model. This equation allows translation from stiffness and natural frequency to the response: either an acceleration, velocity, or displacement. It is a ratio of the response of a system to the harmonic excitation [1].It can be represented in its simplest form for displacement of an undamped SDOF system as

$$\mathrm{H}(\omega) = \frac{1}{(\mathrm{k} - \omega^2 \mathrm{m})} \tag{2.1}$$

This equation represents the displacement of a SDOF system based on the mass m the natural frequency ω, the stiffness k[1].

In modal testing, the displacement, acceleration, or velocity are known, as is the applied excitation, and the FRF must be approximated to fit the known data. Due to the periodicity of the excitation signal, a Fourier transform of the input allows the periodic input, even if not sinusoidal to be related to the Fourier transform of the response function. The FRF can then be defined as a ratio of the response Fourier series to the excitation Fourier series.

As the complexity of estimating the FRF function grows rapidly as the degrees of freedom in the system increase, a preexisting estimator in MATLAB, *modalfrf*, is used to calculate the FRF for the SIMO system. This function allows the estimation of the FRF of a vibrational system by taking the excitation, response signals, and sample rate as inputs and providing the FRF, frequencies, and coherence as outputs [21]. This existing implementation also allows for easy implementation of windowing, which is important for isolating leakage from the response signals.

2.3 Excitation Signals

Choosing an appropriate excitation signal is crucial to modal analysis, and the choice of signal can vary depending on the geometry, available equipment, and time constraints. No matter which signal type is chosen, the signal must excite the beam at double the highest desired frequency [1]. Although it is possible to excite a structure individually at each frequency of interest, this is often inefficient, although it may be useful for examining structures with multiple

modes at similar frequencies. Using more complex excitation signals and leveraging modern signal processing tools, it is often more efficient to use one of the following methods of excitation:

2.3.1 Periodic Chirp Excitation

A periodic chirp function is transient function with an initial burst covering a range of frequencies, followed by an amount of time where there is no applied load. This has the advantage of allowing a single input to measure multiple frequencies. With slower sweeps, a measurement is taken at each frequency of interest after allowing the structure to stabilize at that frequency [1]. While this allows for testing of particular modes of interest with high resolution, the ability of modern signal processing makes that methodology less efficient when determining overall structural properties[1]. Using a chirp function allows for the capture of a broad range of frequencies with a single measurement process. This also makes computing the FRF much easier, at it is simply the discrete Fourier transform of the input over the discrete Fourier transform of the output in the following form:

$$\mathrm{H}(\omega) = \frac{\mathrm{X}(\omega)}{\mathrm{F}(\omega)} \tag{2.2}$$

where $H(\omega)$ is the FRF, $X(\omega)$ is the DFT of the response signal and $F(\omega)$ is the DFT of the excitation signal [1]. Since the excitation signal and the response signal can both be approximated by a DFT and are recorded, the FRF can be easily computed.

Additionally, the use of a consistent chirp function allows for the averaging of the response for repeated samples, which can help eliminate experimental error. It is crucial that the maximum frequency expected is at least double the sample rate in hertz [22]. Failure to ensure this can lead to aliasing, which may cause distortion of the results in the high frequency range. Application of an anti-aliasing filter can mitigate the impact of aliasing but does not allow for frequencies higher than half the sample rate to be determined. In physical testing, combinations of excitation signals can be used to allow for detailed isolation of specific natural frequencies. For example, an initial modal analysis conducted using a chirp signal can identify the natural frequencies of a structure which can then be studied in detail with a high-resolution slow sine sweep to identify the mode more accurately.

2.3.2 Gaussian White Noise Excitation

Compared with the period chirp excitation, a gaussian white noise excitation has the advantage of exciting all the frequencies relatively equally and at the same time. This allows for a theoretically shorter test period, although it required more advanced signal processing to extract the responses [1]. With modern computers the processing time is trivial, and the built in modalfrf MATLAB function is capable of automatically extracting the natural frequencies and mode shapes from either type of excitation.

While it may seem to be an ideal excitation signal, the real difficulties appear in real world testing. To excite a structure with gaussian white noise, the shaker chosen must be both strong enough to accurately impart the excitation frequency onto the physical structure as well as responsive enough to frequency shifts in the excitation signal. For this reason, while useful for computational testing, it is not always practical or physically possible to use in real world testing.

2.4 Modal Extraction

Extracting modal properties requires fitting a curve to describe the measured FRF at each given point. The FRF, as its name suggests, is in the frequency domain and as such any curve fitting must also occur in the frequency domain. As a different FRF is generated for each recorded point on the structure, the combination of these FRFs can allow for the global behavior the structure to be interpreted [1]. While the natural frequencies and the damping ratios can be extracted from the individual FRFs, combining the FRFs allows for a more accurate fit than any single individual curve fitting algorithm. This process can be broadly described as global modal analysis and is well suited to SIMO systems, as each output generates a discrete FRF.

The natural frequencies and damping ratios of any given structure are inherent properties of the material and geometry. In an ideal world, these parameters would be identical at every point where the FRF is calculated, and as a result the natural frequency could be calculated from a single point; however, when multiple FRFs are calculated from a SIMO test, the natural frequencies may not align exactly [1]. The sum of each FRF may be taken to yield a function $H(\omega)$ that represents the average natural frequency and damping ratio for the structure.

2.4.1 Peak Picking

The most basic form of modal extraction is peak picking. As the name suggests, in this method the FRF is plotted on the frequency domain with each resonance peak taken as the natural frequency ω_r of the given mode[1]. The damping ratio is determined by taking the amplitude of the peak $\widehat{H}$ and dividing by $\sqrt{2}$ to identify two points on either side of the peak, ω_b and ω_a [1]. The damping ratio can then be calculated by the following equation [1].

$$2\zeta = \frac{\omega_a^2 - \omega_b^2}{2\omega_r^2} \cong \frac{\widehat{H}}{\omega_r\sqrt{2}} \tag{2.3}$$

This method allows for easy estimation of natural frequencies from the output of a single FRF and can assist in checking to see if the natural frequencies are in the expected range; however, it has several serious drawbacks that limit its utility. It depends highly on the accuracy of the amplitude of the FRF, which can be subject to error during data collection both from experimental error but also due to other factors such as the type of excitation frequency being used [1]. Additionally, the peak picking method assumes that the resonant peaks are fully independent of each other and are not influenced by adjacent resonances, which cannot be assumed to be the case [1]. The limitations of this methodology make it useful for estimation and for checking if the behavior of the FRF is as expected, but it doesn't take advantage of more advanced computational techniques.

2.4.2 Least-Squares Complex Exponential Method (LSCE)

The *modalfit* function in MATLAB implements the LSCE algorithm to allow for modal parameter extraction from multiple FRF functions. As implemented, this algorithm uses the individual FRFs to fit a response using Prony's method [23]. As implemented, this algorithm also generates a reconstructed FRF, allowing the actual FRF to be compared to the FRF estimated by the fit curve. Comparison of the two FRFs allows for visual identification of regions where the fit curve may not align well with the actual system behavior. This allows for comparisons of the accuracy of the various sensor placements to be compared to each other.

2.4.3 Least-Squares Rational Function (LSRF)

The *modalfit* function in MATLAB also implements the LSRF estimation method. Although broadly similar in terms of practical uses to the LSCE method, this algorithm has the advantage of performing better when there is less data available for fitting the FRF. Like the LSCE method, it also allows the FRF to be reconstructed for visual comparison.

2.5 Experimental Considerations

When conducting physical experiments, there are several additional considerations that must be considered. These can broadly be separated into hardware considerations and signal processing considerations.

2.5.1 Experimental Configuration

When it comes to hardware, it is important that the physical model matches the FEA model as closely as possible to ensure the results are comparable. This means that the article under test must be physically constrained in such a way to match the boundary conditions established in the FEA model. Connections must be of sufficient rigidity to allow the article under test to be excited appropriately.

In addition to ensuring the article under test is appropriately constrained, a mounting system for the modal shake must be sufficiently sturdy such that the excitation is imparted into the test article and not into the support structure for the shaker. It is also important that the modal shaker not influence the natural response of the structure under test [1]. A thin stinger rod can be used to attach the modal shaker to the test structure. By allowing axial forces from the modal shaker to pass but being flexible in all other directions, the stinger reduces the impact attaching the shaker can have on modes that are not purely axial in relation to the direction of the excitation force [1].

2.5.2 Signal Processing

Data collected from the accelerometers, strain gauges, and force transducers during testing will invariably contain some level of noise. Although shielded cables and minimizing cable length can reduce signal noise, it is generally not feasible to eliminate it. Similarly, Although an ideal excitation frequency is fed into the driver for the modal shaker, the analog

output may contain noise not present in the original digital signal. Furthermore, this analog signal may not be perfectly translated into physical excitation in the modal shaker due to inefficiencies in the physical system. For these reasons, it is important that the output to the modal shaker be recorded in addition to the sensor inputs.

All the data recorded from the system should be recorded with a sampling rate of at least double that of the highest desired natural frequency (Nyquist frequency). To prevent aliasing from occurring, where high frequency components above the Nyquist frequency can be interpreted as lower frequencies—anti-alias filtering must be applied to the signal [24]. Typically an analog low pass filter is sufficient, although more complex filters are available with digital signal processing equipment [1].

3 Sensor Selection Methodologies

3.1 Chapter Summary

Before an ML model can be developed, a methodology must be established to extract the natural frequencies and mode shapes from sensors such as those used during physical testing. Modal analysis is used process data recorded from sensors to determine the natural frequencies, mode shapes, and damping ratios of the structure under test. To ensure the methodology used for modal analysis is accurate, an initial test of the modal analysis will be conducted using data derived from an FEA simulation of a beam. Using FEA based data ensures the data is free of noise and allows comparison of the extracted modes to those calculated in FEA. The method chosen for modal analysis will then be applied to the various sensor selections so that the differences in modal analysis methods cannot influence the extracted natural frequencies.

3.2 Experimental Models

The following sections outline the material properties and geometries to be used for numerical and experimental testing.

3.2.1 1D Cantilever Beam

To demonstrate and verify the techniques to be used for modal extraction, a simplified 1D cantilever beam will first be considered. The properties of the beam are outlined in the following table. The properties of this beam are based on a physical beam that is in the lab to enable experimental verification of the modes determined by the sensor selection as described in the next sections.

Table 3.1 1D Cantilever Beam Properties

Property	Value
Length	0.242 m
Width	0.032 m
Thickness	0.00305 m
E	70 GPa
ν	0.33
ρ	2700 kg/m^3

A finite element model of the beam under consideration was implemented in MATLAB, building on existing code used for 1D beam analysis [19]. This code was modified to extract the data needed for modal analysis so it could be used to test different techniques for sensor selection.

The use of a finite element model allows for rapid iteration when trialing different techniques as sensors do not need to be physically repositioned. Instead, the load type, point of application, and sensor locations can be selected easily in software. This methodology should allow for a reasonable estimation of the behavior of the various sensor selection techniques, which can then be verified through physical experimentation. Additionally, the finite element method provides a numerical computation of the modes for the structure, which allows the real modes to be directly compared modes derived from sensor data.

3.2.2 2D Cantilever Plate

To allow testing to be expanded to more complex geometry, a 2D Plate with a large hole through the center will be tested. This geometry will contain more complex vibrational modes, but is still relatively easy to analyze, model, and physically implement for experimental verification. Plate dimensions are show in Fig. 3.1 Plate GeometryThe plate was composed of A36 structural steel with the following shown in Table 3.2.

Table 3.2 Plate Properties

Property	**Value**
Height	54.72 cm
Width	30.48 cm
Thickness	0.61 cm
E	200 GPa
ν	0.3
ρ	7850 kg/m^3
Hole diameter	10.16 cm

The plate was modeled in ANSYS mechanical 2023 R1. The plate was meshed with hex dominated quadratic elements in a single layer. The mesh contained a total of 17106 nodes and 2910 elements. The FEA mesh is visible in Fig. 3.2. To account the fact that sensors mounted to

the flat plate can primarily measure acceleration in bending, the first 13 primary bending modes were identified. The natural frequencies and mode numbers are as follows:

Table 3.3 Bending Modes of 2D Plate

Mode	Frequency (Hz)
1	24.057
3	147.39
6	420.73
9	843.47
15	1343.8
16	1433.1
18	1518.1
21	1973.6
24	2231.9
25	2458.8
30	2998.9
44	4405.4

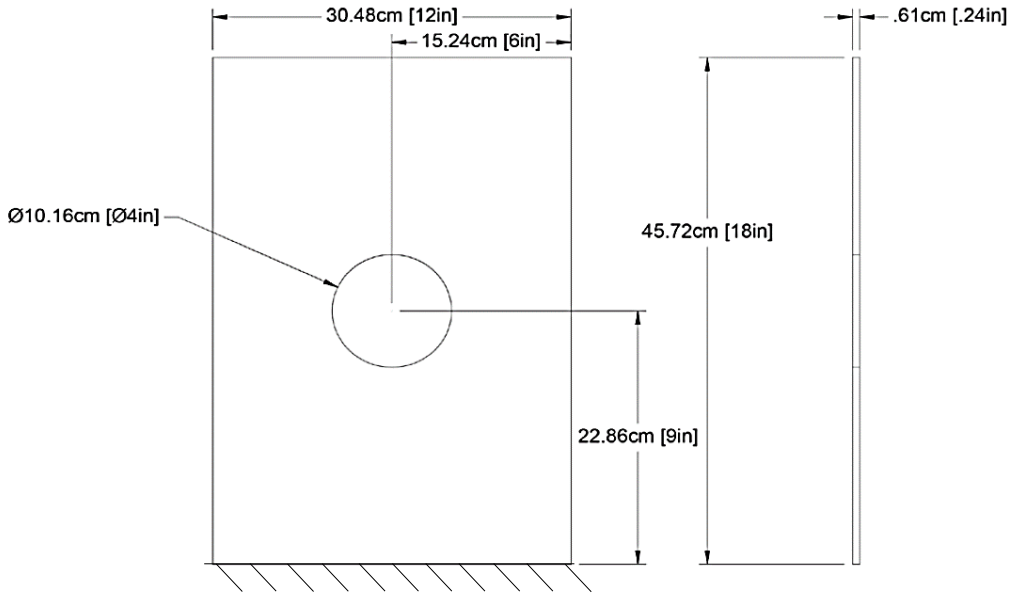

Fig. 3.1 Plate Geometry

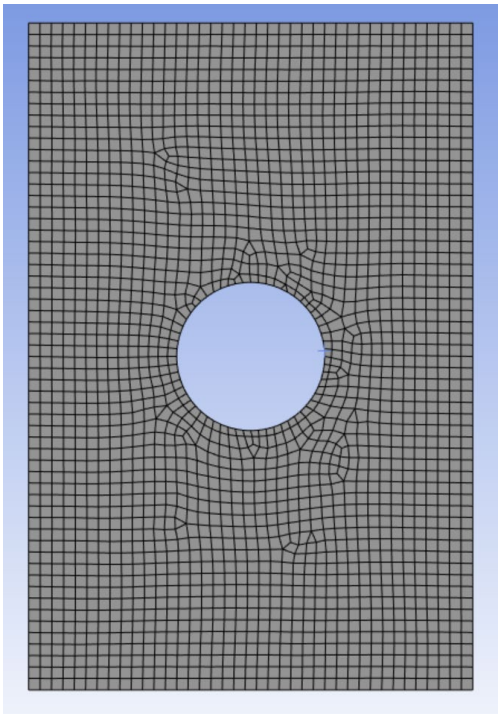

Fig. 3.2 Plate Mesh

3.3 Modal Extraction

During physical experimentation, input and response signals will be recorded digitally, processed, and analyzed in MATLAB. Therefore, the techniques described here to process the FEA data will also be applicable to processing real world data. It is important to note, however, that real world signal data often includes some level of noise, and additional pre/post processing may be needed to compensate. Further discussion of noise rejection methodology is discussed in section **Error! Reference source not found.**.

For validation of the modal extraction methodology, a concentrated chirp load was applied to the end of the beam. The chirp load is summarized in Table 2 and was described by the MATLAB expression *chirp(t,20,0,20000)*, where *t* is a vector of times from 0 to 5 seconds sampled at 44100 Hz.

Table 3.4 FEA Concentrated Load

Property	Value
Duration	5s
Frequency Range	20-20000 Hz
Frequency Sweep	Linear
Magnitude	10 N
Sampling Frequency	44100 Hz

To begin the modal analysis process, a table of the displacement, velocity, and acceleration at each node for each time step is extracted from the FEA results. Likewise, the magnitude of the load is also extracted for each time step. Each node is considered as a potential sensor location. While the data collected both in FEA and in physical testing is in the time domain, to extract the natural frequencies from the system, the data must be transformed into the frequency domain. After selecting the number of sensors, the input and output data are processed with the frequency response function *modalfrf* in MATLAB. The frequency response function (FRF) takes the input excitation signal and the response signals from the time domain into the frequency domain [1]. For periodic excitation, the FRF is the ratio of the Fourier transform of the excitation to the Fourier transform of the response [1]. Thus, the number of FRFs should match the number of sensors used. The *modalftf* function in MATLAB provides outputs in the form of the FRFs and coherence, which is a measure of how accurate the FRF is for the given data, with a coherence of 1 representing the best case.

After determining the FRF, it is fed into *modalfit* which calculates the natural frequencies, damping ratio, mode shapes, and reconstructed FRF. While the natural frequencies are of principle interest in this work, the mode shapes can also be compared to ensure that the reconstructed mode shapes accurately reflect the real system. A large divergence between these results indicates that the frequencies calculated may not reflect real-world natural frequencies.

Using this technique, it should be possible to extract natural frequencies that match those determined using FEA. In practice, however, this has proven substantially more difficult. Minor changes in signal windowing, even for an idealized numerical signal, can have dramatic results in the ability of the algorithms implemented in *modalfit.* Multiple algorithms are built into *modalfit*. Of those, peak-picking is least accurate, with least-square complex exponential coming

in second and least square rational function giving the best results. Unfortunately the *lsrf* method is the most computationally expensive and has limitations on the number of modes it can output depending on the input data. With experimentation, proper windowing allows modalfit to achieve a relatively high degree of accuracy at least for the first seven natural frequencies of the simple 1D beam.

As seen in Fig. 3.3, the prediction of natural frequencies is accurate for the first seven natural frequencies, but the 8th natural frequency is not captured. The 8th natural frequency captured by the *lsrf* algorithm matches closely the 9th natural frequency calculated by the FEA model (not picture below).

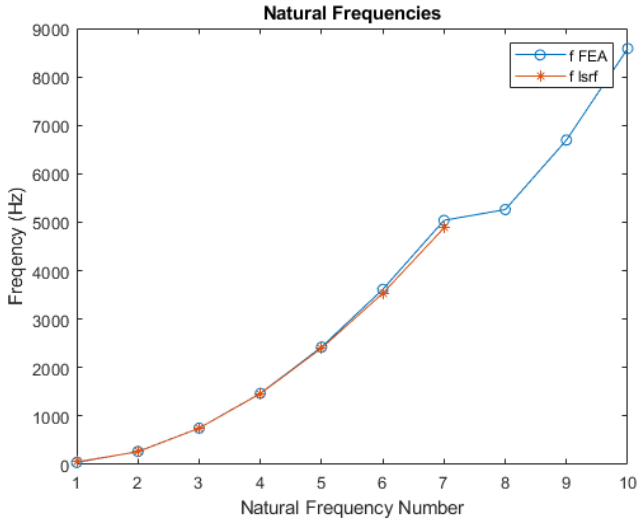

Fig. 3.3 Comparison of natural frequencies determined using FEA and modalfit 'lsrf'

The second function, modalsd, produces a stabilization diagram, shown below in Fig. 3.4. This figure represents the estimated natural frequencies and damping ratios for the first 50 modes. The fit methodology used for the stabilization diagram is the same algorithm as used for the modalfit algorithm; however, the stabilization diagram appears to represent the real modes of the system more accurately when compared to the estimates provided by the modalfit function.

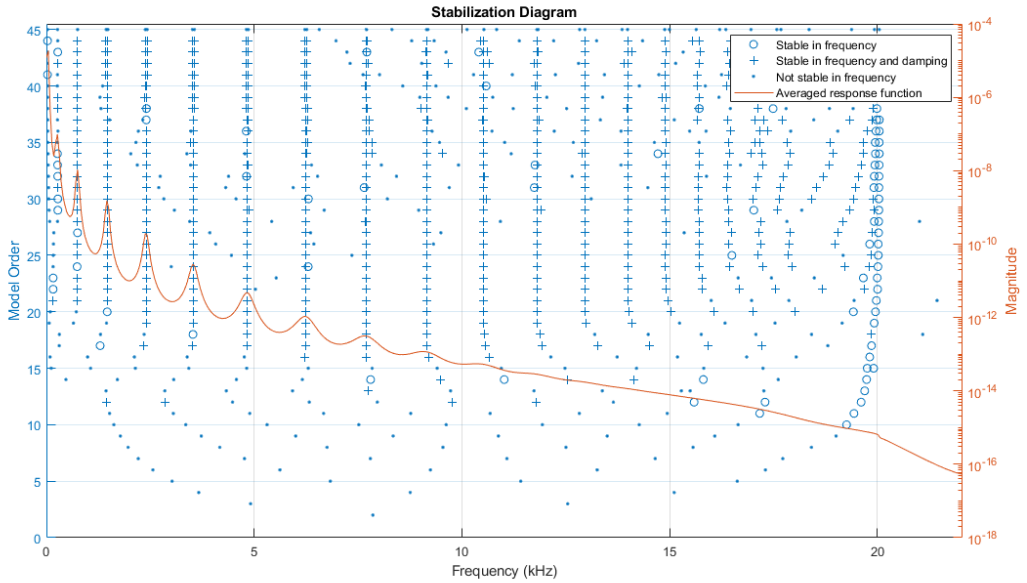

Fig. 3.4 Stabilization diagram for chirp excitation and 10 response

. The average response function, plotted in orange, displays peaks at each of the system's natural frequencies. The crosses represent predicted modes that are stable in both frequency and damping, and it is these frequencies that represent the natural frequencies of the system. Nonphysical modes—those that exist as artifacts in the data—and unstable poles can be eliminated this way. A comparison of the frequencies determined by the various modal extraction methodologies, as well as the real natural frequencies of the system, are presented in the following table.

Table 3.5 Estimated natural frequencies of a 1D cantilever beam

Natural Frequency Number	FEA Natural Frequency (Hz)	*Modalfit lsrf* Natural Frequency (Hz)	% Error	*modalsd* Natural Frequency (Hz)	% Error
1	42.83	55	28%	44.06	3%
2	268.26	263.92	2%	264.59	1%
3	750.45	749.41	0%	749.83	0%
4	1468.6	1463.3	0%	1466.35	0%
5	2423.4	2401.6	1%	2403.2	1%
6	3612.5	3534.3	2%	3544.3	2%
7	5033.1	4773.4	5%	4839.5	4%

As shown in Table 3.5, the derived natural frequencies for the first seven modes of *modalsd* align well with the real natural frequencies. Additionally, the first 7 modes from the *modalfit* algorithm are also relatively close. From the stabilization diagram, it is apparent that the peaks at higher frequencies are substantially less pronounced compared to the initial 5 frequencies. Even then, the divergence is within 10%. This is similarly able to be confirmed by looking at the ideal FRF.

To compute the ideal FRF function of the 1D beam, the FRF is directly calculated using the mass matrix M, damping matrix C, and stiffness matrix K derived from the FEA model. An idealized force vector of 1 is applied to the end of the beam. For each frequency from 1 to 10 kHz the FRF is calculated according to the following equation:

$$H(\omega) = (-[M]\omega^2 + \mathrm{j}[\mathrm{C}]\omega + [\mathrm{K}])^{-1}[\mathrm{F}] \tag{3.1}$$

Where $H(\omega)$ is the FRF function, ω is the frequency in rad/s, and $[F]$ is the force vector. Plotting H for each degree of freedom yields the ideal FRF represented in Fig. 3.5. From the plotted FRF, it is evident that modes 1 through 4 exhibit clearly defined peaks. The code used to generate the ideal FRF function may be found in Appendix G: Ideal FRF MATLAB Code. Mode 5 through 7 show clear peaks, but they are not as well defined as the first modes. Modes above the 7th mode are much less well defined, potentially due to damping in the structure. Due to this limitation, the analysis here will principally focus on the first 7 modes.

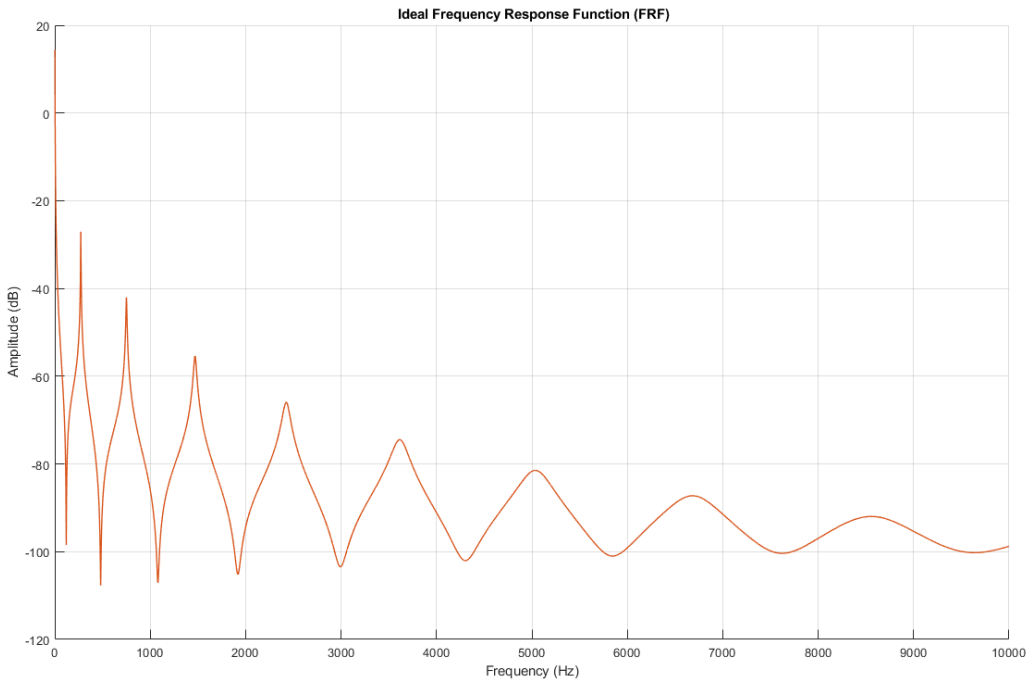

Fig. 3.5 Idealized FRF to unit excitation

Attempts to rectify this issue have occupied the majority of the semester, although several different estimators have been tried. One feature that has recently been identified that may solve the difficulties in extracting the natural frequency is that all these estimators seem highly sensitive to the type, magnitude, and duration of the excitation function.

While theoretically the magnitude of the excitation function and type of excitation should not matter so long as the frequencies of interest are captured, increasing the magnitude and duration of the chirp function has improved the fit of the function to better capture the real natural frequencies. Until the past week, the closest predicted mode was within 10Hz of the real natural frequency. Compared with the current closest predicted natural frequency, which is within 1hz, this is a major improvement. Further adjustment to the excitation frequency may improve the ability of the *modalfit* algorithm to better fit the high frequency range.

4 Sensor Selection Techniques

4.1 Chapter Summary

The following chapter summarizes two different methodologies that will be examined in detail and implemented for comparison with the novel machine learning approach. Both the EIM and the RKE methodologies are currently used in industry due to their relative speed and accuracy. RKE methodologies are considered the gold standard for modal analysis instrumentation selection by NASA and the US Department of defense [16]. Both methodologies require identifying the expected modal behavior of the structure through analytical or numerical methods. Based on prior knowledge of structures, an initial estimation of sensor placement is made, and then EIM or RKE methods are used to test the effectiveness of the selection provide feedback on how to improve the sensor placement. In each methodology, an iterative aspect is involved where the algorithm is slightly changed and rerun. With large arrays of possible sensor locations, such as those derived from detailed FEA models, even these methods can be computationally expensive to run for large structures. A summary of the two current methodologies is presented below.

The effective independence method (EIM) was implemented in MATLAB and is presented with a description of the implementation algorithm, followed by a verification of the implementation. After verification has been verified, EIM is then applied to several example structures to demonstrate the operation of the algorithm and how current sensor positions can be determined algorithmically using data derived from finite element analysis. Demonstrating the variation in selected locations on a cantilevered beam for different targeted mode shapes, the methodology is then expanded to a more complex geometry, a 2D cantilevered plate.

4.2 Current Methodology

4.2.1 Effective Independence method (EIM)

By maximizing the independence of each sensor—i.e. ensuring data collected is different from sensor to sensor—duplicate data is not included in the response dataset allowing the largest amount of information to be collected with the lowest amount of replication [11]. This method

relies on the identification of mode shapes for a given structure. As the purpose of modal testing is to identify the natural frequencies and mode shapes, these are not known; therefore, the mode shapes must be identified through modal analysis conducted in FEA [11]. The use of known or estimated mode shapes in the algorithm allows the reasonable assumption that the results will not converge to non-representative mode shapes [12] This is one of the principle drawbacks to EIM, as sensor selection cannot account for unknown modes that may occur in the real world but do not appear in FEA. If there are specific modes in the FEA results that are of more interest, EIM can provide targeted sensor selection for those modes which may require less sensors than needed to capture the full behavior of the structure.

The algorithm used to implement EIM is derived from that developed by Kammer in "Sensor Placement for On-Orbit Modal Identification and Correlation of Large Space Structures" [25]. A summary of the method is presented here for reference, although a more comprehensive description can be found there. Equation 3.1 represents the sensor output, where u_s represents the sensed modal coordinate vector, Φ_s represents the mode shape vector for a given sensor location, and q is the target modal coordinates based on the FEA analysis [25].

$$\{u_s\} = [\Phi_s]q \tag{4.1}$$

The effective independence vector EfI can be represented by the diagonal of the matrix calculated in equation 4.2 [25].

$$\{E\ f\ I\} = \left\{ \left[[\Phi][\Gamma][\Lambda]^{-\frac{1}{2}} \right] \left[[\Phi][\Gamma][\Lambda]^{-\frac{1}{2}} \right] \right\} \tag{4.2}$$

The higher the EfI score of a node, the more important a given location is for calculating the independence of the modal vectors [11].

In the EfI method, sensor locations with the lowest value are eliminated, and the EfI is then recalculated from the subset of candidate locations [11]. The process is complete when the desired number of sensors is reached or when all remaining sensor locations have similar EfI values.

4.2.2 Residual Kinetic Energy Methods

The RKE method is an alternate method for sensor selection which may offer improved performance over EIM. The goal of the RKE method is to ensure all recorded modes are fully orthogonal and therefore independent from each other [16]. The selected points for sensor placement can be tested by predicting the orthogonality of the test modes [16].

$$[OR] = [\Phi_a]^T[M_{aa}][\Phi_a] \tag{4.3}$$

OR is the estimated orthogonality, Φ_a is the instrumented mode, and M_{aa} is the Test analysis matrix which is defined on degrees of freedom for the sensor placement, usually determined by the FEA model [16]. The simulated modes are then used to computer an error matrix [R] [16]. The error matrix is defined as

$$[\mathrm{R}] = [\Phi_\mathrm{f}] - [\Phi_\mathrm{fa}] = [\Phi_\mathrm{f}] - [\Psi_\mathrm{fa}][\Psi_\mathrm{a}] \tag{4.4}$$

The error matrix is a subtraction of the free DOF modes from the free sensor DOF modes [16]. In this case $[\Phi_f]$ is determined through FEA or other analytical methods, and is then compared against the instrumented free mode $[\Phi_{fa}]$ [16]. The error matrix [R] is then used with the free mass matrix augmented by the error matrix $[M_{ff}R]$[14]. In the following equation, $\otimes$ represents element-wise matrix multiplication.

$$[\mathrm{RKE}] = [\mathrm{M_{ff}R}]\otimes[\mathrm{R}] \tag{4.5}$$

The RKE is a scaled value, with each column in the RKE matrix representing a mode and the each DOF in the column sums to 1 [16]. The RKE matrix indicates which DOFs should have sensors placed on them as they will have higher values in the column [16]. The RKE matrix column will add to much less than 1 if that mode is already appropriately instrumented [16]. By iterating through this matrix you can determine if enough degrees of freedom are covered, as well as identifying DOFs where additional sensor placement is needed.

4.3 Effective Independence Method Implementation

4.3.1 EIM Implementation

Using the form of the EIM matrix *EfI* as described previously, a MATLAB script was written to implement the algorithm on the modal analysis results produced by a finite element modal. While this script was integrated into the finite element analysis code for the beam as described in Chapter 2, the code can be used on data derived from any modal analysis, so long as the natural frequencies and mode shapes are provided.

The script, found in Appendix A: Effective Independence Code, takes as an input a matrix of the target mode shapes (eigenvectors) Φ_s. Each row in the matrix represents a node, and each column in the matrix is a mode shape for the corresponding degree of freedom. The input matrix is also augmented by the addition of a leading index column, which identifies the individual node. The effective independence of each node is then calculated by the diagonal following equation [25]:

$$EfI = \Phi_s[\Phi_s^T \Phi_S]^{(-1)} \Phi_S^T \tag{4.6}$$

This diagonal vector *E* yields a value between 0 and 1. A row with a value of 0 indicates that node cannot be used to sense the target modes, while a value of 1 indicates that node is required to observe the target modes [25]. Where the values of *E* are equal, either node could be removed from the set of sensors without impacting the linear independence of the target modes [25].

The lowest ranking row of E and the corresponding row in Φ_s are eliminated, and the new Φ_s is then input into equation (**4.6**. The process is repeated until the desired number of sensors is reached. The sum of the column vector *E* must always be equal to the number of target modes and as result *E* must be recomputed whenever a node is removed as irrelevant [25]. As such, it is optimal to remove only one node per iteration. Additionally, it is impossible to have fewer sensors than target modes.

The determinate of the fisher information matrix A_0 as calculated below can be used to measure how much information is covered by a given sensor set [25].

$$A_0 = \Phi_s^T \Phi_s \tag{4.7}$$

The determinate of A_0 provides a quantitative value representing the amount of information covered by a given sensor set [25].

4.3.2 EIM Verification

In the paper by Kammer, a brief example of the EIM is provided for a case with three sensors and two target modes [25]. The matrix Φ_s is given as [25]:

$$\Phi_s = \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{4.8}$$

This given value was used to test the algorithm implemented in MATLAB. Both the effective independence vector, and the eigenvalues and eigenvectors of the Fischer information matrix A_0 match the values given. This simple example serves to validate the methodology as implemented in MATLAB.

4.3.3 EIM Applied to a Cantilever Beam

The following sections apply the methodology outlined above to the same cantilever beam used previously for modal analysis. First, an 11-node version of the beam is used to select 5 sensors to identify the first three bending modes. Although theoretically it is possible to identify three modes with only three sensors, signal noise, variations in sensor placement, and imperfect conditions require additional sensors. For all the following cases, additional sensors were selected such that 150% of the theoretical minimum required sensors were present.

4.3.3.1 11 Node Cantilever Beam

Further qualitative validation was conducted by examining an 11-node finite element model of a cantilever beam. The fixed end of the beam was placed at node 1, and the free end at node 11 For the beam, the first three bending modes at $f_1 = 42.83$ Hz and $f_2 = 268.28$ Hz, and $f_3 = 750$ Hz. The mode shapes are plotted in the figure below.

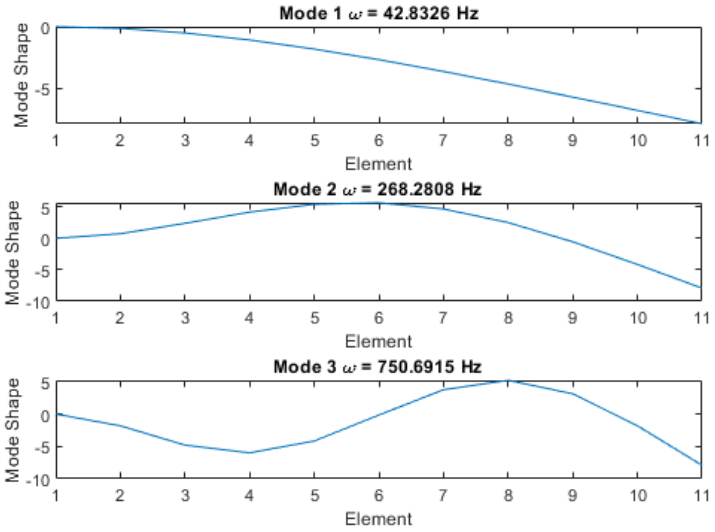

Fig. 4.1 Mode shapes for the first three bending modes (11 DOF)

The EIM algorithm was then run to select 5 of the 11 nodes for sensor placement. The results of the column vector *E* representing effective independence are tabulated for each iteration in Table 4.1. Eliminated nodes are indicated by '-'.

Table 4.1 Nodal effective independence for cantilever beam

	Iteration Number					
Node	**1**	**2**	**3**	**4**	**5**	**6**
1	0	-	-	-	-	-
2	0.023	0.023	-	-	-	-
3	0.176	0.176	0.180	-	-	-
4	0.339	0.339	0.346	0.421	0.485	0.503
5	0.318	0.318	0.323	0.352	0.476	0.478
6	0.242	0.242	0.244	0.258	-	-
7	0.291	0.291	0.291	0.292	0.352	0.424
8	0.342	0.342	0.344	0.356	0.383	0.511
9	0.268	0.268	0.270	0.281	0.287	-
10	0.291	0.291	0.291	0.292	0.292	0.349
11	0.709	0.709	0.710	0.719	0.725	0.735

In the first iteration, node 1 is removed as it is located at the support condition where a sensor would be fixed. As node 1 is zero, and the sum of the effective independence must equal the number of target modes—three in this case—the effective independences in the second iteration are the same. This provides an opportunity to improve the algorithm because in cases where a node or nodes are 0, both those nodes and the next lowest node may be removed in the same iteration without any loss in fidelity of the results.

Initially, the nodes being removed are those closest to the support condition of the beam. Logically, this makes sense as nodes closest to the support condition will experience smaller accelerations and displacements than nodes closer to the end of the beam. Indeed, the most important node in all 6 iterations is the node at the free end of the beam. The pattern of removing nodes near the support condition is broken in the 5th iteration where node 6 is removed before node 4 and 5. Examining the mode shape plots reveals that node 6 is located near a node of the third mode shape. In the 6th iteration, node 9 is removed because it is also near a node of the 2nd and 3rd mode shape.

4.3.3.2 101 Node Cantilever Beam

Expanding on the prior case, the model of the beam is expanded to be comprised of 100 elements with a length of 1.27 cm each. During early experimentation, an FEA mesh of this density was used when attempting to determine if increasing mesh density and therefore the corresponding number of FRFs available for the *modalfit* command would increase the accuracy of that command. Although it did not substantially change the modes predicted by the *modalfit* command, the larger dataset is used here to illustrate a simple structure with many DOFs. The first 10 bending modes are the target modes for this analysis for prediction with 15 sensors. The first 5 bending modes are plotted Fig. 4.2 along with their corresponding natural frequencies.

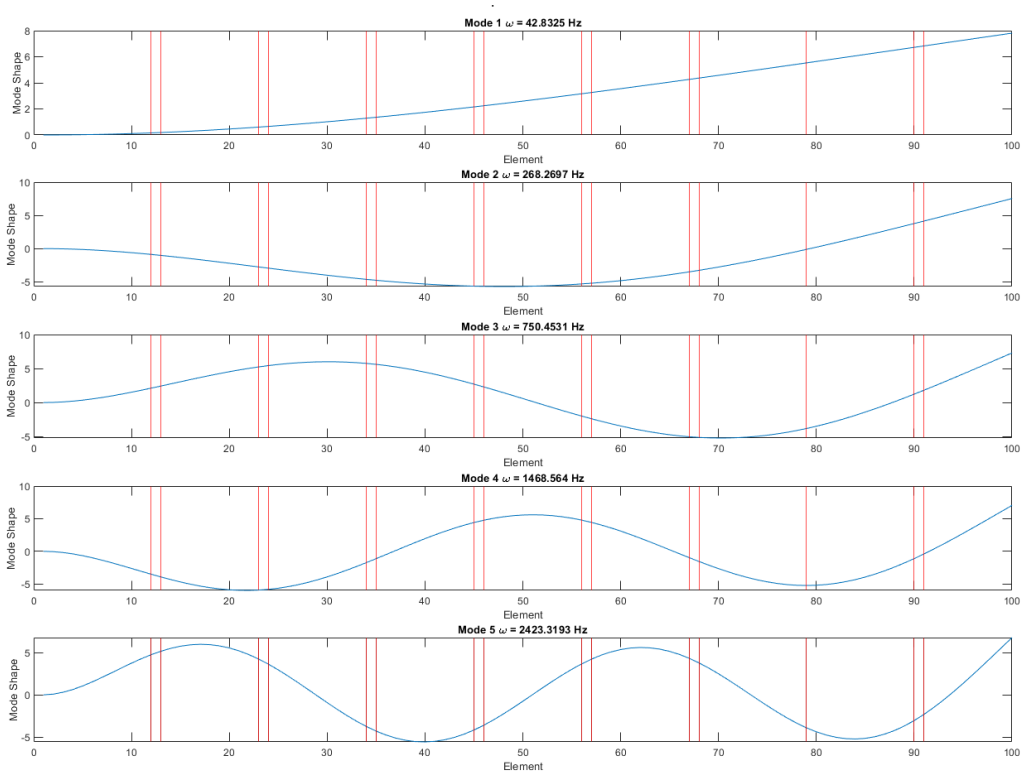

Fig. 4.2 The first 5 bending mode shapes for a 101 DOF system.

The selected sensor positions illustrated in Fig. 4.2 correspond to the points shown in Table 4.2. The effective independence of each node is either very close to 0.5 or 1, indicated that all the remaining sensors maintain similar levels of importance compared to each other. Node 79 and node 101 are the nodes with the highest effective independence. Node 101, being located at the free end of the beam, logically, is an important point. Node 79 is the only other node which appears by itself. This node is again located close to the end of the beam.

Table 4.2 Effective independence of selected sensor positions to predict the first 10 bending modes

Node	Effective Independence
12	0.500
13	0.501
23	0.502
24	0.500
34	0.501
35	0.499
45	0.500
46	0.500
56	0.499
57	0.501
67	0.498
68	0.506
79	0.992
90	0.500
91	0.504
101	1.000

4.3.4 EIM Applied to a Flat Plate

The method outlined above will now be applied to the flat plate with a hole located in the center. The dataset is derived from a FEA model as described in Chapter 2. The modes and natural frequencies of the plate were exported from ANSYS after conducting a modal analysis for the first 50 modes. The target mode shapes were identified as the first 13 transverse bending modes. The selected modes and natural frequencies are outlined in Table 3.3. The following figures illustrate the selected modes. These modes were chosen as the exhibit the largest deflections in the +/- Y direction as established by the model, and as such are the modes that are easiest to excite and measure with the experimental equipment available. The dataset was then further reduced to only encompass those nodes which were located on the surface of the plate, to ensure all possible sensor locations are physically accessible. An ANSYS APDL script found in Appendix B was used to export the nodal coordinates, node number, and mode shape for each surface node at the chosen natural frequencies. After export, the dataset was imported to MATLAB for postprocessing.

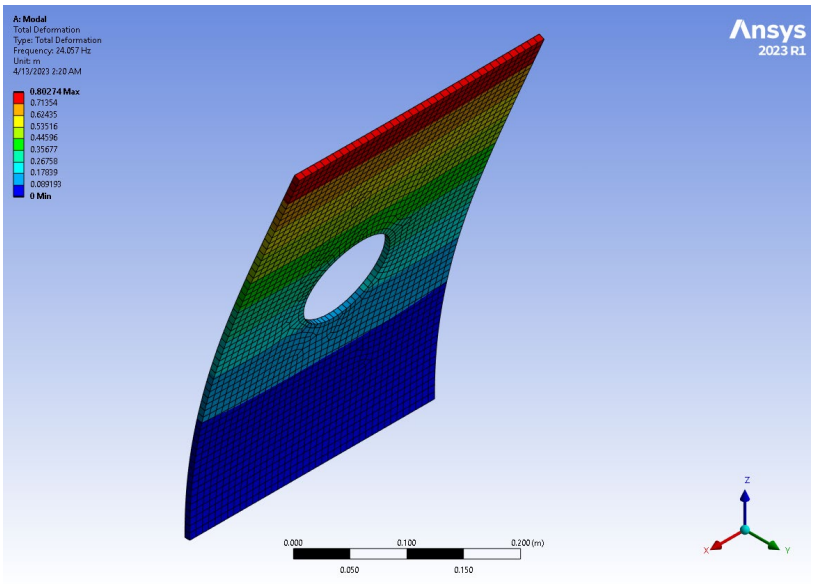

Fig. 4.3 Mode 1: 24.057 Hz

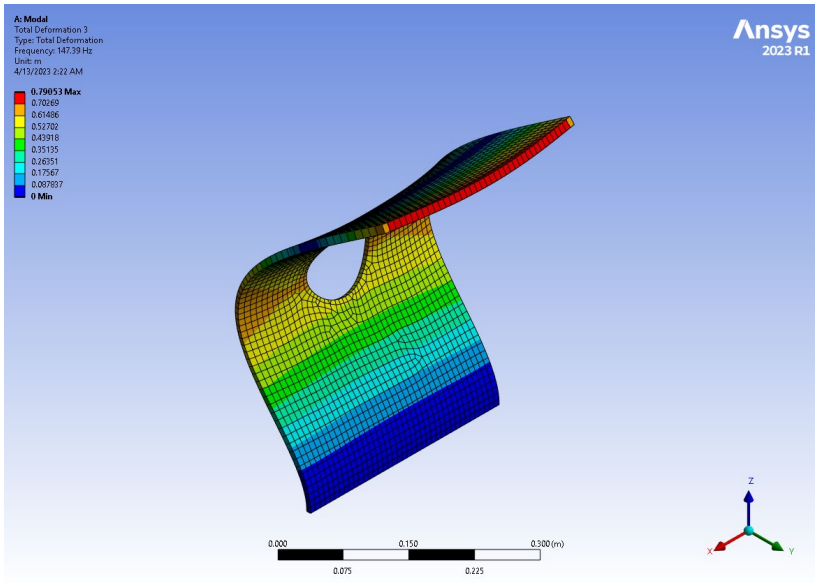

Fig. 4.4 Mode 3: 147.39 Hz

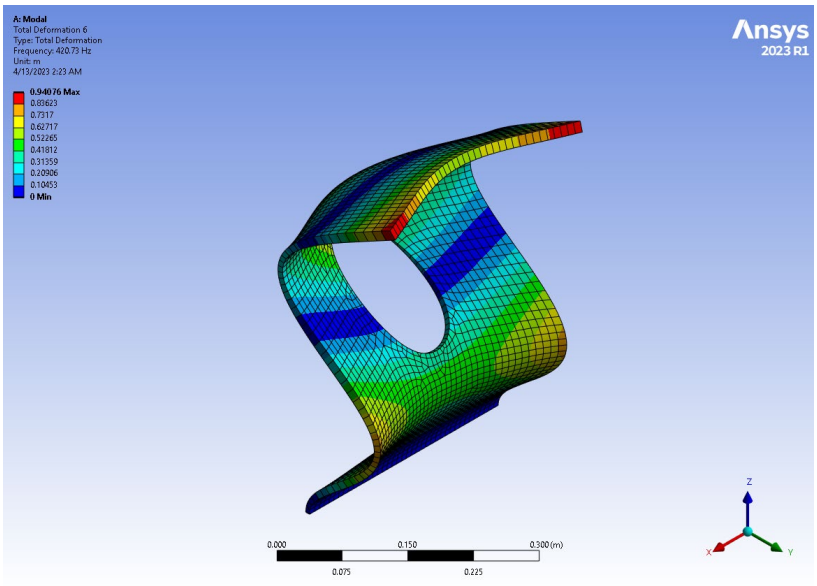

Fig. 4.5 Mode 6: 420.73 Hz

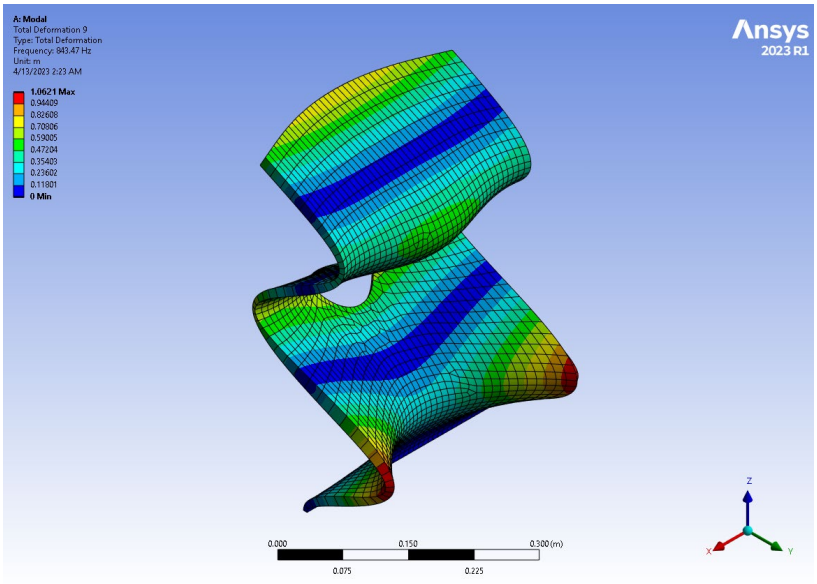

Fig. 4.6 Mode 9 843.47 Hz

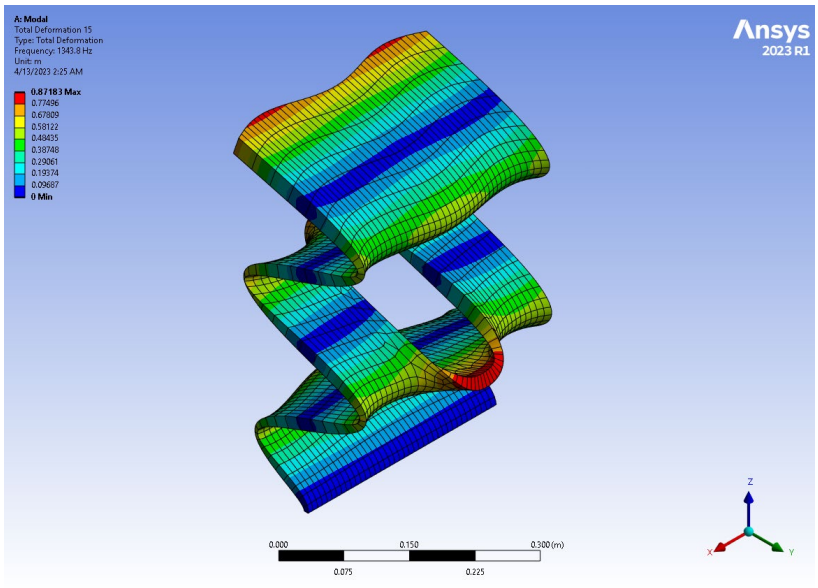

Fig. 4.7 Mode 15: 1343.8 Hz

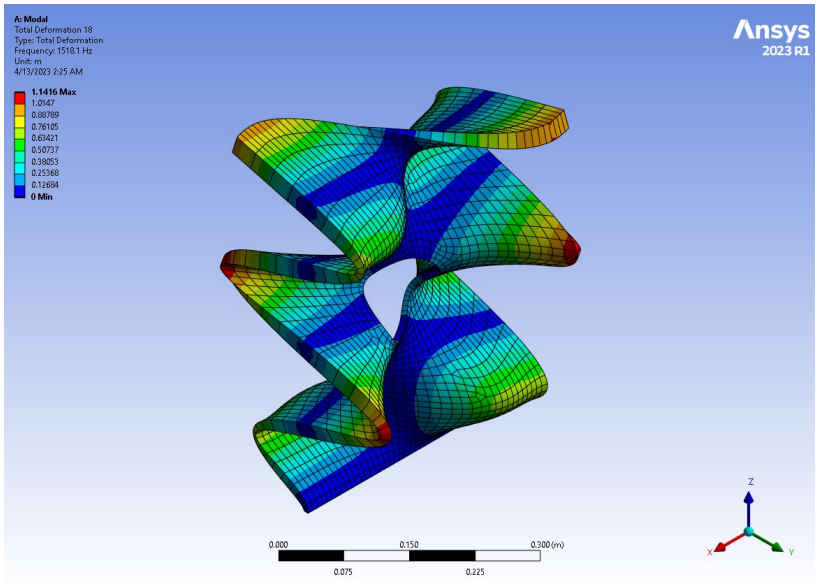

Fig. 4.8 Mode 16: 1433.1 Hz

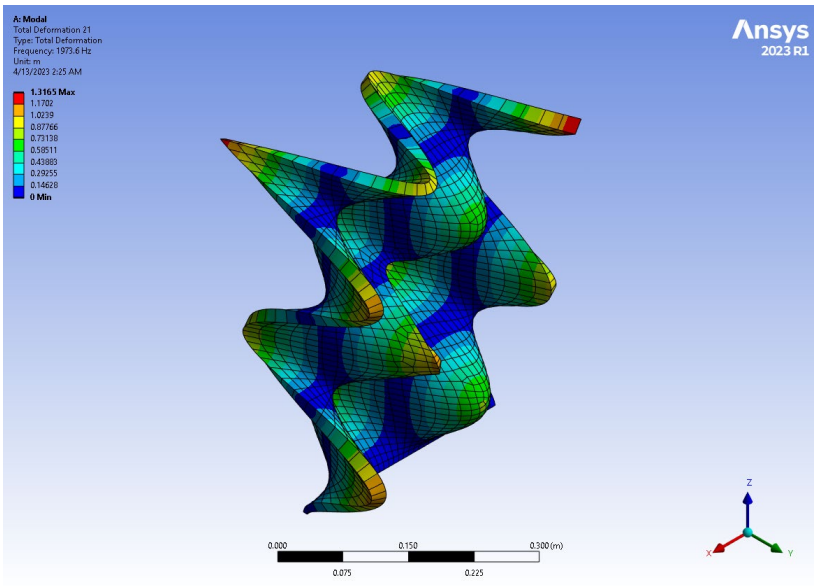

Fig. 4.9 Mode 21: 1973.6 Hz

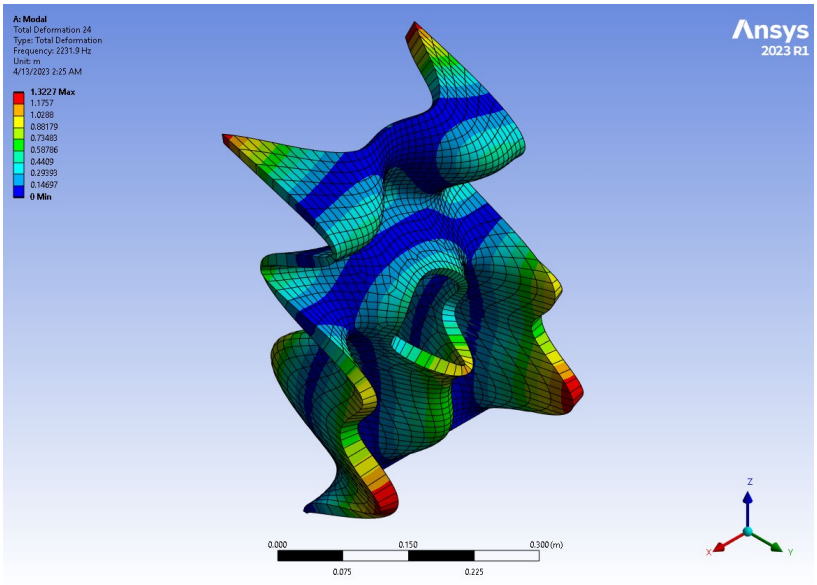

Fig. 4.10 Mode 24: 2331 Hz

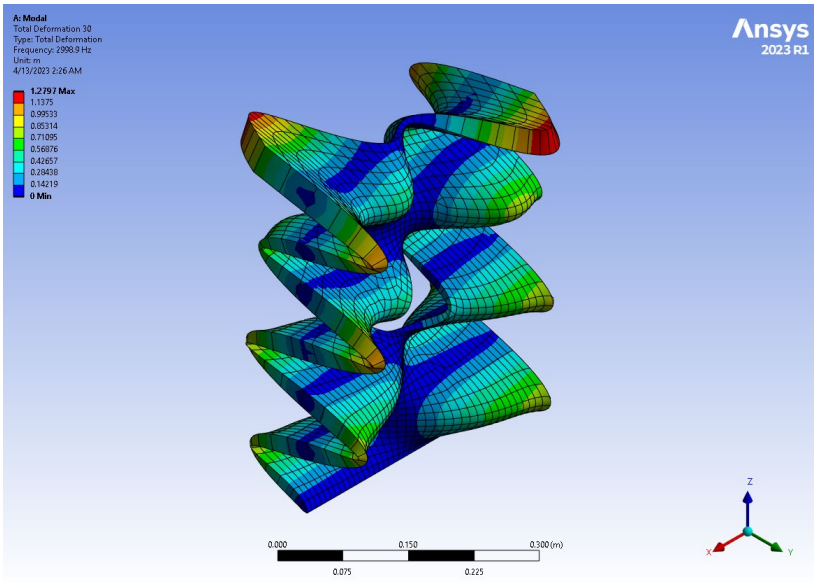

Fig. 4.11 Mode 30: 2998 Hz

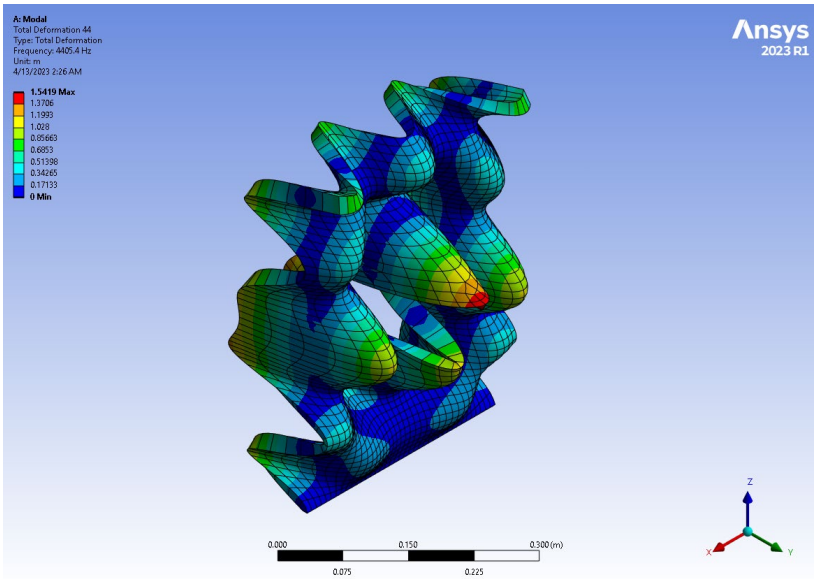

Fig. 4.12 Mode 44: 4405 Hz

Using the EFI algorithm implemented previously in the chapter, the dataset was processed to identify the optimal locations for 20 sensors. As the EfI algorithm implemented is iterative, with only one node removed per iteration, sensor selection for the plate is more computationally expensive, taking on average 200 seconds to process the 6839 by 14 array of DoFs and mode shapes. The resulting selected sensors are illustrated below in Fig. 4.13.

Inspecting the sensor placement and the mode shapes, the sensors seem likely to capture most of the mode shapes. However, the EFI algorithm favored points on the left side of the plate

much more heavily. This should not impact the observability of the chosen natural frequencies as the out-of-plane modes are expected to be symmetric.

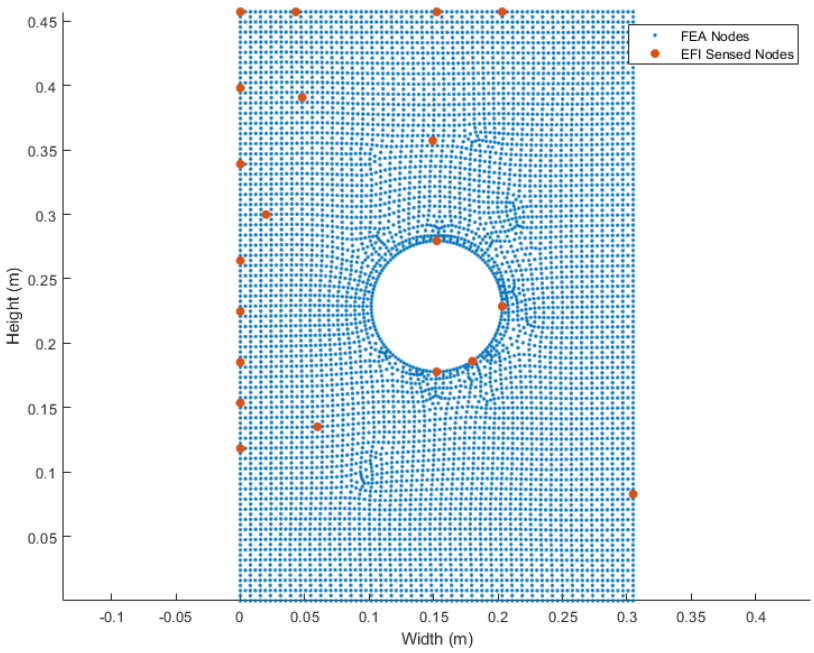

Fig. 4.13 EfI selected plate sensor locations

5 Machine Learning Sensor Selection

5.1 Chapter Summary

Emerging methodologies in machine learning present an opportunity to improve on exiting methodologies for sensor selection. Regression models, such as the random forest regressor, have been shown to effectively select sensor locations for structural health monitoring (SHM) [19][26]. Similar methodologies can be applied for sensor selection that must be conducted prior to physical modal testing.

Compared with traditional techniques, ML methodologies may improve fidelity and improve results compared to effective independence methods. For the following techniques, only accelerometer data was considered.

5.2 Random Forest Regressor

The random forest regressor (RFR) was selected for sensor placement based on performance in SHM sensor selection applications [19]. The RFR is an ensemble learning method in the averaging method family. Averaging ensemble methods combine the results from multiple estimators and average the predicted results to reduce variance [27].

The estimators in this algorithm use decision trees (DTs) to predict the chosen output variable. The RFR method splits the main dataset into several subsets, and generates a DT for each of the new training data sets [27]. Splitting the data into these random smaller datasets and then averaging the results of the individual DTs decreases the variance of the results, as DTs by themselves can overfit the data [27]. In addition to the randomness introduced by varying the input data for each DT, random perturbations in the DTs are also introduced [27]. Once the model has been run, the R squared value and mean squared error (MSE) can be used to evaluate performance.

The random forest regressor algorithm was implemented as found in the scikit-learn python library. The code for the implementation of this algorithm can be found in Appendix C, and is based on prior code written for structural health monitoring as found in Choppala et al. [26].

5.2.1 Dataset Creation

The dataset for the RFR was extracted from FEA models and reformatted as specified in

Table 5.1. Each row in the dataset is a frequency for which the FRF of each node is calculated. Each row in this case represents the operational deflection shape (ODS) for a given frequency *f* for every node in the FEA model. The frequency is used primarily for tracking and is not input into the RFR. The output column in this table represents the value the model should attempt to select. The model will output a table of all the features (nodes) input and their corresponding importance when attempting to predict the output value.

Table 5.1 RFR input data formatting

Freq (Hz)	**Node 1**	**Node 2**	…	**Node *n***	**Output**
f	Node 1 FRF(*f*)	Node 2 FRF(*f*)	…	Node *n* FRF(*f*)	Output (*f*)

As a result of this method, choosing a parameter for the output is crucial to producing results that reflect an optimal sensor placement. Three distinct values were chosen to be examined: raw ODS, normalized ODS, and average FRF.

In the case of the raw ODS, to produce a scalar:

$$ODS_{\omega}^{T} ODS_{\omega} \tag{5.1}$$

is computed, which results in a distinct value for each ODS at each frequency. The ODS and therefore this value is dependent on the load condition of the beam and will change depending on the magnitude of the load applied to the beam.

To decrease the sensitivity of the output to the load conditions, a normalized form was computed:

$$\frac{ODS_{\omega}^{T} ODS_{\omega}}{|\mathrm{ODS}|^{2}} \tag{5.2}$$

Dividing the ODS product by the magnitude of the ODS at that frequency eliminates the effect of loading magnitude on the beam. The last output chosen was the average FRF at a given frequency, where *n* is the number of nodes and the sum of the FRF at a given frequency is taken across all nodes *n*.

$$\frac{\sum FRF(f)}{n} \tag{5.3}$$

The results of selecting for these different values are presented in the following section.

5.3 RFR Applied to a Cantilever Beam

A 100-element cantilever beam identical to the beam presented in Chapter 5.2.3 was created. The beam was subjected to a broadband gaussian white noise excitation from 0 to 50 kHz as shown in Fig. 5.1.

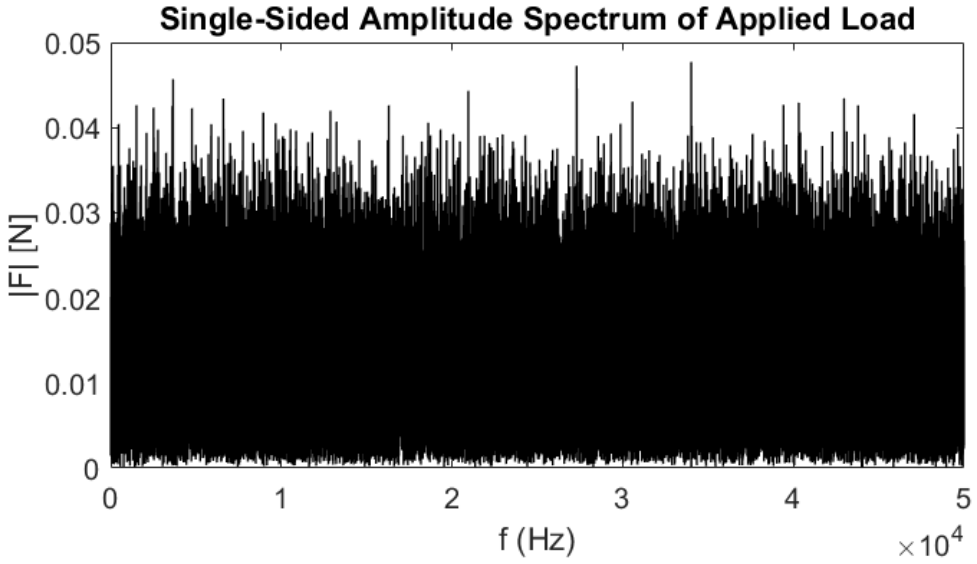

Fig. 5.1 Excitation spectrum of load applied to a 100-element cantilever beam

The FRF for each DOF was then computed using the MATLAB modalfrf function, with a Hanning window of size 500 with 0 overlap. Four distinct tables were created in the format outlined in Table 5.1 with the ODS, normalized ODS, average FRF, and all three output functions. Each table was exported as a .csv file before being imported into python for the ML algorithm.

While the error values in Table 5.2 appear excellent, with R2 values approaching 1 and MSE that are effectively 0, this may represent overfitting of the model. This may result in the RFR being unable to generalize. As the algorithm is being trained directly on the data and is not expected to generalize, this does not necessarily invalidate the methodology.

Table 5.2 Errors of RFR for 100 element cantilever beam

	ODS	**Normalized ODS**	**Avg. FRF**
R2	-	0.950	0.985
MSE	-	3E-11	7.40E-14

The 10 most important features from each element were then taken as candidate sensor locations. A table of the 10 selected sensor locations is presented along with the candidate sensor

locations determined in Chapter 5.2.3 using the effective independence method in Table 5.3. It is important to note that the sensor locations presented in this table are listed in numerical order and not sorted by importance. In the case of the EIM, the candidate sensors are all equally important whereas for the RFR derived sensors, the features are ranked by importance.

The first 10 features of the normalized ODS account for 28.80% of the variance, while the first 10 features of the RFR Avg. FRF account for 32% of the variance. The first 10 features of the ODS RFR account for 31% of the variance. In all cases, approximately 20 features are needed before at least 50% of the variance is accounted for; however, the ODS RFR features exhibit slightly higher individual variance in the first 3 features, dropping off substantially to importance more in line with the other selection methods. In this example case adjacent nodes are not eliminated.

Table 5.3 Most important sensor locations on a 100-element cantilever beam in numerical order based on different modal sensor selection approach.

Effective Independence Method	RFR ODS	RFR Normalized ODS	RFR Avg. FRF
9	12	16	33
16	43	35	34
25	44	39	38
32	53	43	42
39	69	59	56
48	70	73	73
63	83	80	82
70	86	86	86
78	87	91	93
94	96	99	101

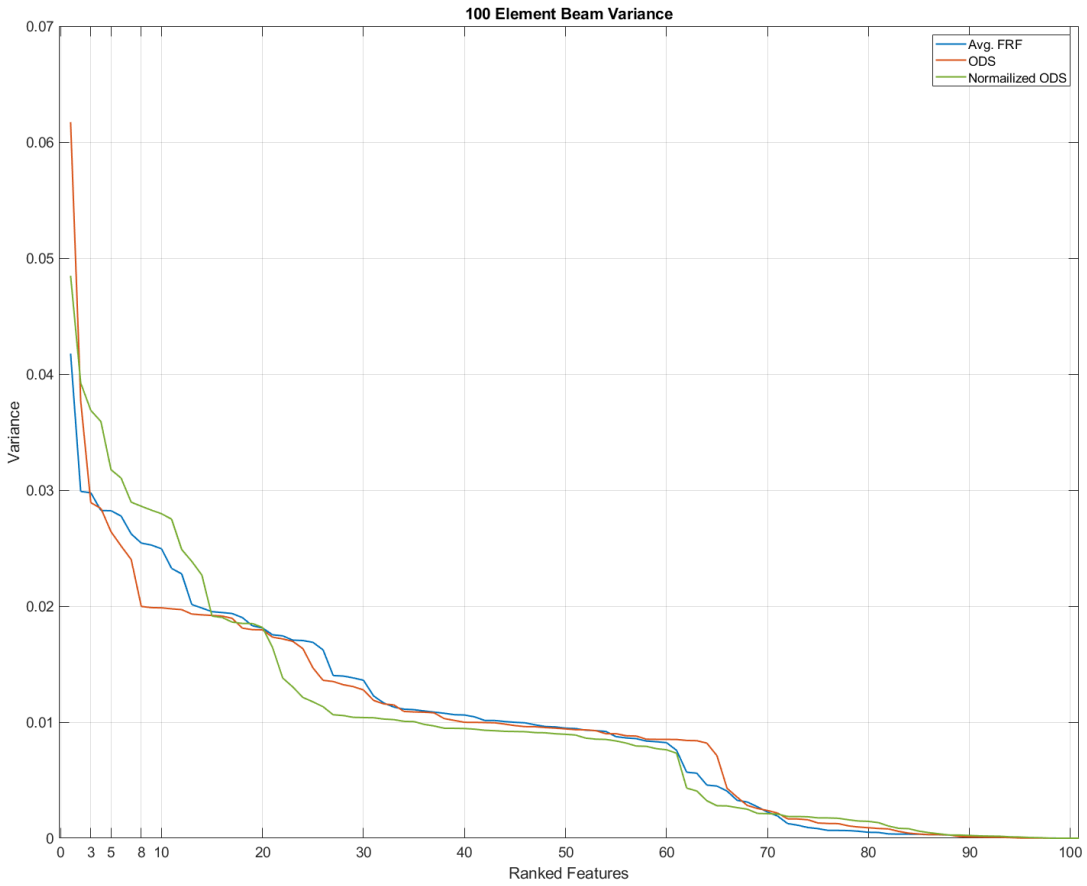

Fig. 5.2 Variance as a function of number of features for a 100-element cantilever beam

To estimate the mode shapes the RFR derived methods may predict, value of the actual mode shape was taken at each candidate sensor selection. While not representing the actual FRF and modal extraction it allows a prediction to be made for which modes may not be able to be captured by a given sensor set.

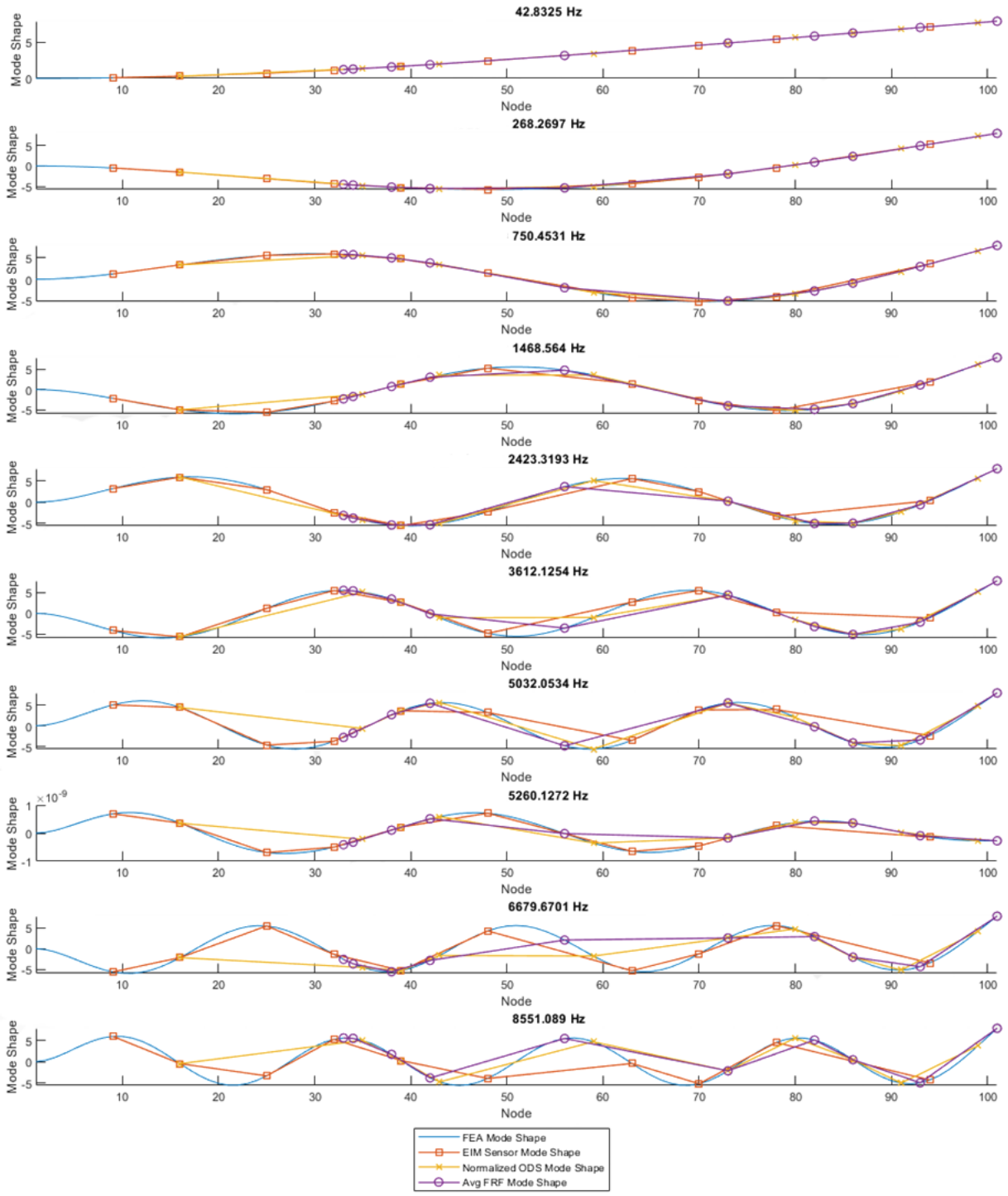

Fig. 5.3 First 10 predicted mode shapes for a 100-element cantilever beam

As shown in Fig. 5.3, all three sensor sets can predict the first four modes with relative accuracy. However, the RFR derived sensor sets favor the free end of the beam, and so are unable to capture the behavior of the beam closer to the fixed end. This weakness becomes apparent as soon as the 3rd mode, when the Avg FRF method linearizes the mode shape from the

30^{th} node to the origin. As mode number increases, the predictions made by the ML derived techniques fail to capture the behavior of the first third of the beam, and all methods weight the free end of the beam more heavily.

Initially this behavior was suspected to be a result of the excitation frequency, as a frequency chirp was used which produced a higher excitation at lower frequencies; however, this behavior remains despite changing the excitation frequency to a broadband gaussian white noise which excites all frequencies between 0 and 50 kHz equally. In the next chapter, the selected sensors will be run through the modal extraction technique developed in Chapter 2 to further characterize the effectiveness of the ML based sensor selections.

6 Comparison of Sensor Selection Techniques

6.1 Chapter Summary

To compare the impact of the various sensor selection methodologies, a methodology was developed to test sensor configurations using FEA derived data. The use of computational data for this initial comparison allows for the comparison between 'ideal' cases. All the datasets are free of error introduced by noise, variability in experimental configuration, inaccuracies in sensors, etc.

The dataset is still sensitive to changes in sampling rate, signal windowing, and mesh density; however, these are all able to be directly controlled in software, enabling direct comparisons to be made between the various sensor selection options. The details for each analysis are contained in the following sections, but the process is similar for both geometries:

1. A FEA analysis is conducted to extract the data needed for sensor selection: modal analysis in the case of the effective independence and transient excitation for the machine learning dataset.
2. The sensor selection algorithms are run, and the sensed nodes are identified for the different selection methodologies.
3. Sensor data is extracted from the nodes for the test condition.
4. The FRF is extracted from the sensor data.
5. Natural frequencies and mode shapes are extracted from the FRF function.

There are only minor differences in the process between the 1D plate and the 2D plate, mainly due to the different software used to conduct the FEA analysis.

The number of sensors and excitation frequencies chosen were constrained by available data acquisition equipment in the lab for physical verification in the next chapter.

6.2 1D Beam

For the 1D beam sensor selection, 2 different excitation frequencies were chosen for comparison. Although traditional methods should select sensors independently of the excitation frequency, the RFR based method is sensitive to the chosen excitation frequency. The first excitation is a linear chirp, and the second excitation case uses a gaussian white noise input signal.

6.2.1 Gaussian White Noise Excitation

Now, the same procedure as in the previous section is repeated, with the only difference being the excitation signal. In this case, a gaussian white noise signal was generated. The excitation had a duration of 2 seconds and a sample rate of 44100 Hz. The single-sided amplitude spectrum is show in Fig. 6.1.

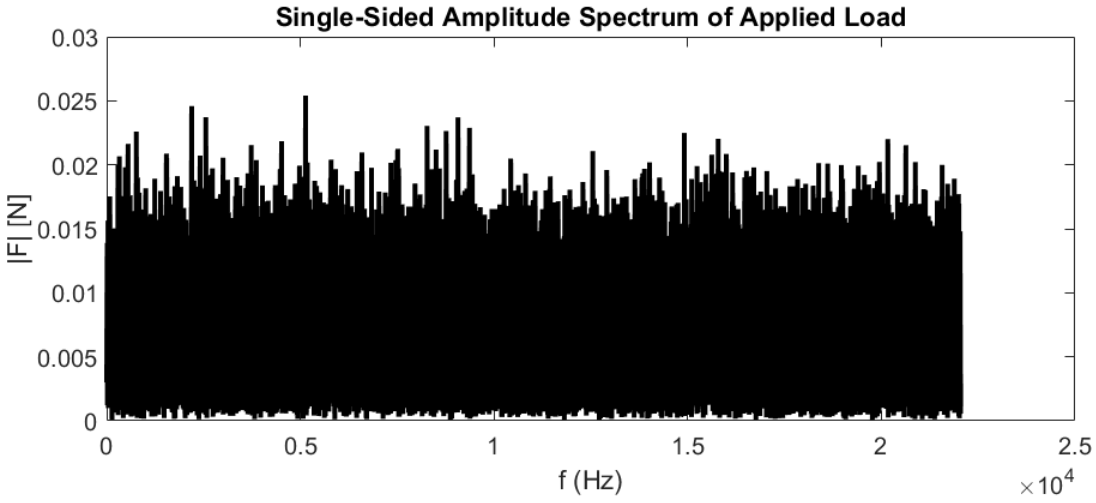

Fig. 6.1 Gaussian white noise excitation signal

As before, the sensor locations on the various mode shapes are show in Fig. 6.2 and Fig. 6.3. Examining the plots, the ODS sensor selector appears to perform worse, with almost all the sensors placed at inflection points for the 7th natural frequency. The FRF data computed for the RFR function was windows with a Hamming window of size 1000 and 600 overlap; however, during modal extraction a Hamming window of size 500 and 200 overlap was found to produce better results. Again, this illustrates the sensitivity of modal extraction to windowing, and with a different excitation signal and duration, a different window can be more optimal.

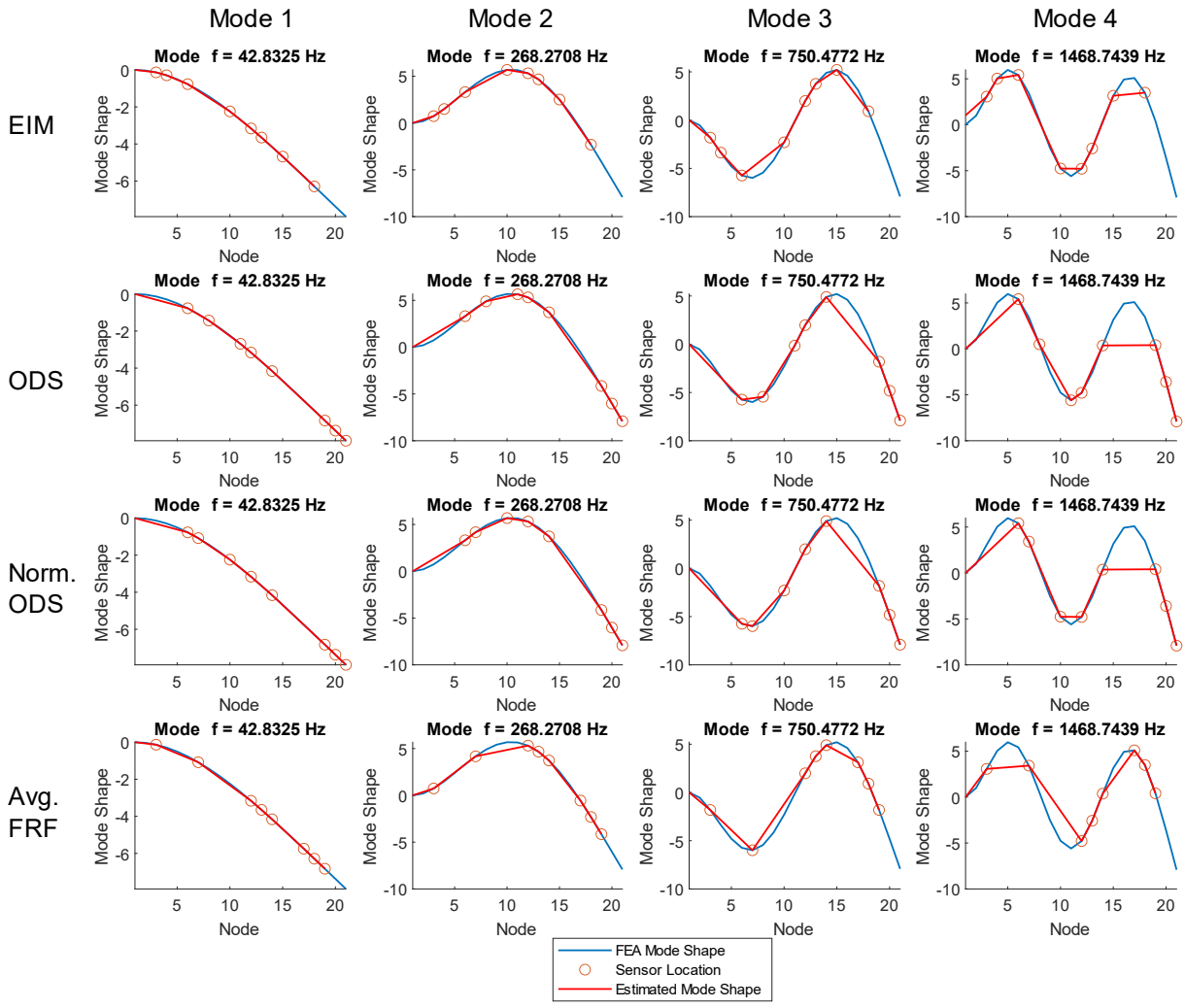

Fig. 6.2 Sensor locations from gaussian white noise excitation for mode 1 through 4

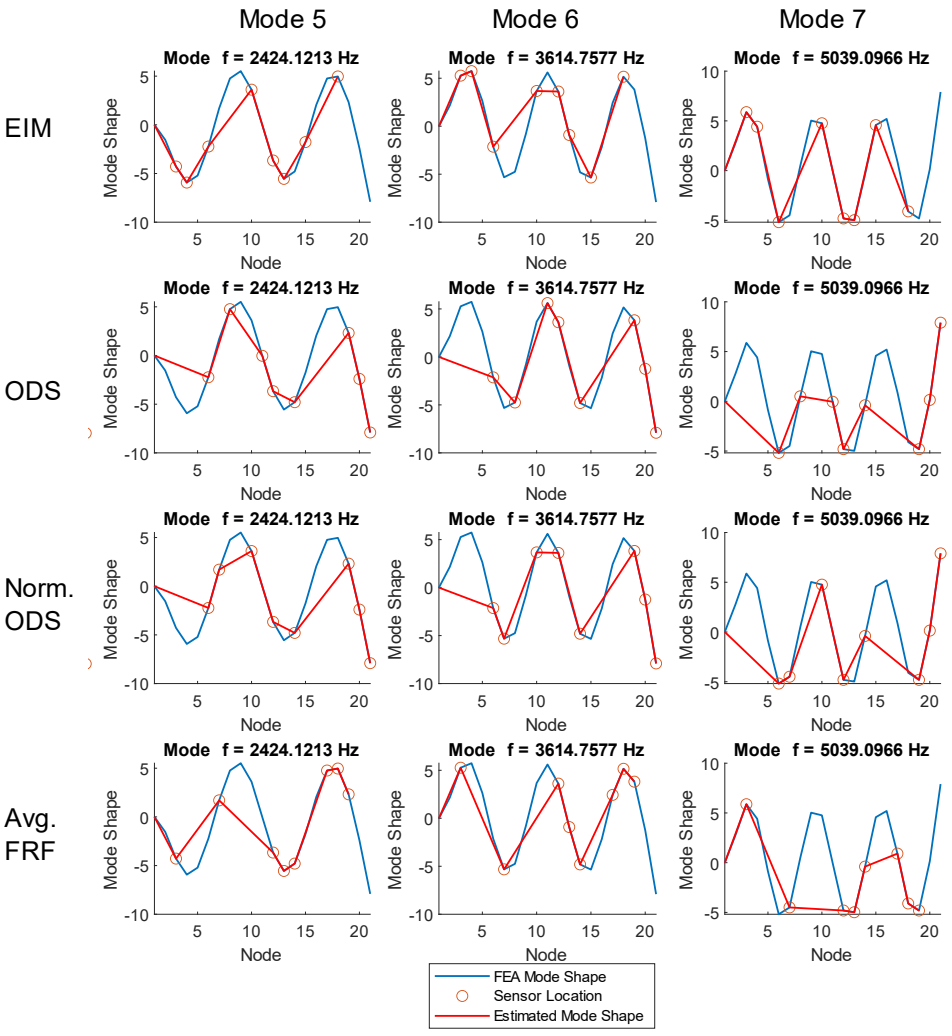

Fig. 6.3 Sensor locations from gaussian white noise excitation for mode 5 through 7

The modal extraction from the given sensor sets was conducted identically to that in the prior excitation signal, with the exception that a Hamming window of size 500 with 200 overlap was used. This window was chosen through an iterative process and appeared to yield frequencies closest to the FEA derived frequencies. Again, the EIM and ODS sensor selection methodologies yield results that closely match the FEA derived frequencies; however, all sensor configurations are poor predictors of the lowest natural frequency, with percent error ranging from 15% to 30% as seen in Table 6.1. In the mid-range frequencies, both the EIM and ODS method again perform best, beginning to diverge above the 6th mode. The *lsrf* algorithm as implemented in MATLAB is dependent on window size and in this case, the window is too small

to estimate more than the first 6 natural frequencies; however, increasing the window beyond this has the effect of increasing error of all sensor selection methodologies by up to 100%.

Table 6.1 Natural frequencies, 1D beam, gaussian white noise excitation

FEA	EIM		ODS		norm ODS		FRF	
f (Hz)	f (Hz)	% Err	f (Hz)	% Err	f (Hz)	% Err	f (Hz)	% Err
42.8	–	100%	–	100%	54.3	27%	–	100%
–	84.1	–	118.1	–	–	–	107.6	–
–	192.2	–	–	–	198.9	–	200.3	–
268.3	278.9	4%	277.5	3%	281.6	5%	278.0	4%
–	–	–	–	–	385.0	–	–	–
–	649.7	–	650.3	–	661.3	–	655.6	–
750.5	755.6	1%	755.7	1%	757.5	1%	751.7	0%
–	–	–	–	–	–	–	911.3	–
–	1398.5	–	1348.9	–	–	–	–	–
1468.7	1482.5	1%	1471.0	0%	1462.8	0%	1462.6	0%
–	–	–	–	–	–	–	–	–
2424.1	2396.0	1%	2397.1	1%	2395.1	1%	2398.2	1%
3614.8	3523.6	3%	3526.0	2%	3531.6	2%	3499.3	3%
–	–	–	–	–	–	–	–	–
5039.1	4837.4	4%	4834.3	4%	4747.1	6%	–	–
5261.4	–	–	–	–	–	–	5470.3	4%
6696.0	–	–	–	–	–	–	–	–

6.2.2 Linear Chirp Excitation

The 1D beam described in Table 3.1 was modeled with 20 elements in a MATLAB FEA script. The beam was excited with a linear chirp signal with the properties described in Table 3.4. The single-sided amplitude of the excitation frequency is plotted in Fig. 6.4.

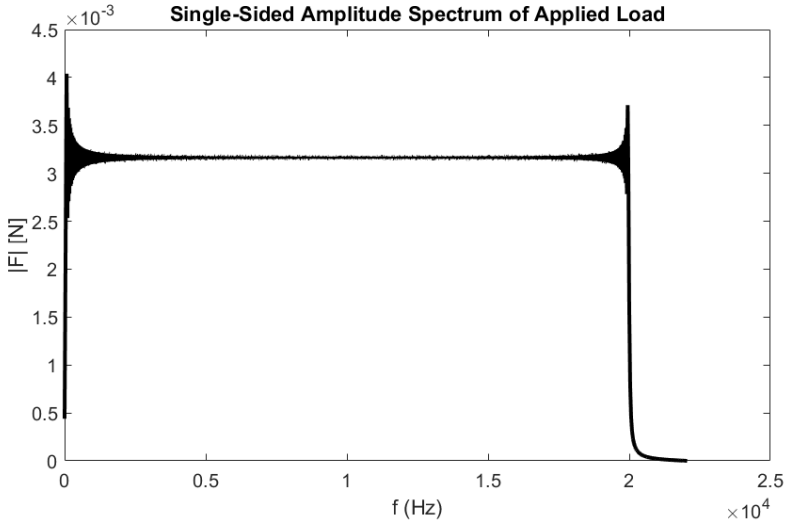

Fig. 6.4 Single-sided amplitude of linear chirp from 20 Hz to 20 kHz

The sensor selection process was conducted using the effective independence method, ODS based RFR, normalized ODS RFR, and FRF RFR selection methodologies. To better visualize the locations of the sensors and to visualize the potential for the sensors to capture the desired mode shapes, each selected sensor location was plotted on the FEA derived mode shape for the first 7 modes. This is illustrated in Fig. 6.5. Each column of charts represents a mode, and the rows display the different sensor locations. Sensor locations do not change in each mode but are plotted on top of the different mode shapes. By examining where the sensors are placed, the ability of the sensors to measure a given mode shape can be approximated visually. The errors for the RFR method are displayed in Table 6.2.

Table 6.2 RFR errors for sensor selection

	ODS	**ODS Norm**	**FRF**
R2	0.979	0.973	0.987
MSE	2.249	1.98	2.77

Examining the sensors, all the selection methodologies appear to track the first two mode shapes adequately; however, the normalized ODS and FRF based RFR sensor selection

methodologies miss more peaks than the EIM and ODS methods, especially at higher frequencies. This suggests that placing sensors at those locations may cause misidentification of the mode shapes. Both the normalized ODS and FRF based methods place sensors at higher frequency inflection points and so will not be able to capture those frequencies.

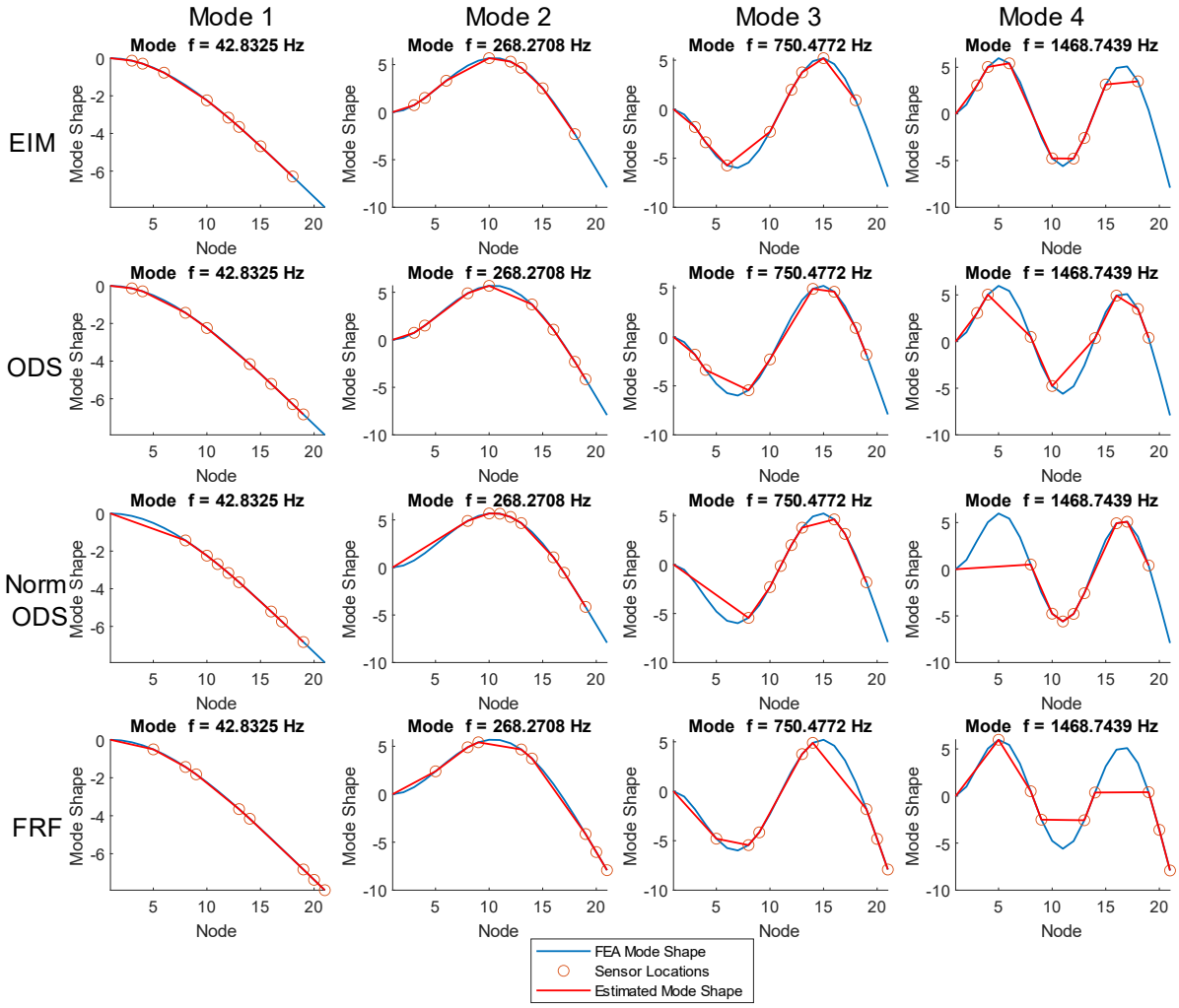

Fig. 6.5 Sensor locations from chirp excitation for mode 1 through 4

When the mode extraction methodology developed in Chapter 2 is applied, the first seven natural frequencies are extracted using the *modalfit* function with the *lsrf* curve fitting methodology. The H_1 FRF function is calculated for the given sensor set and filtered with a Hamming window of size 1000 with 600 overlap using the MATLAB *modalfrf* function. The extracted modes were then plotted against the FEA derived natural frequencies for comparison. The calculated natural frequencies are also available in Table 6.3.

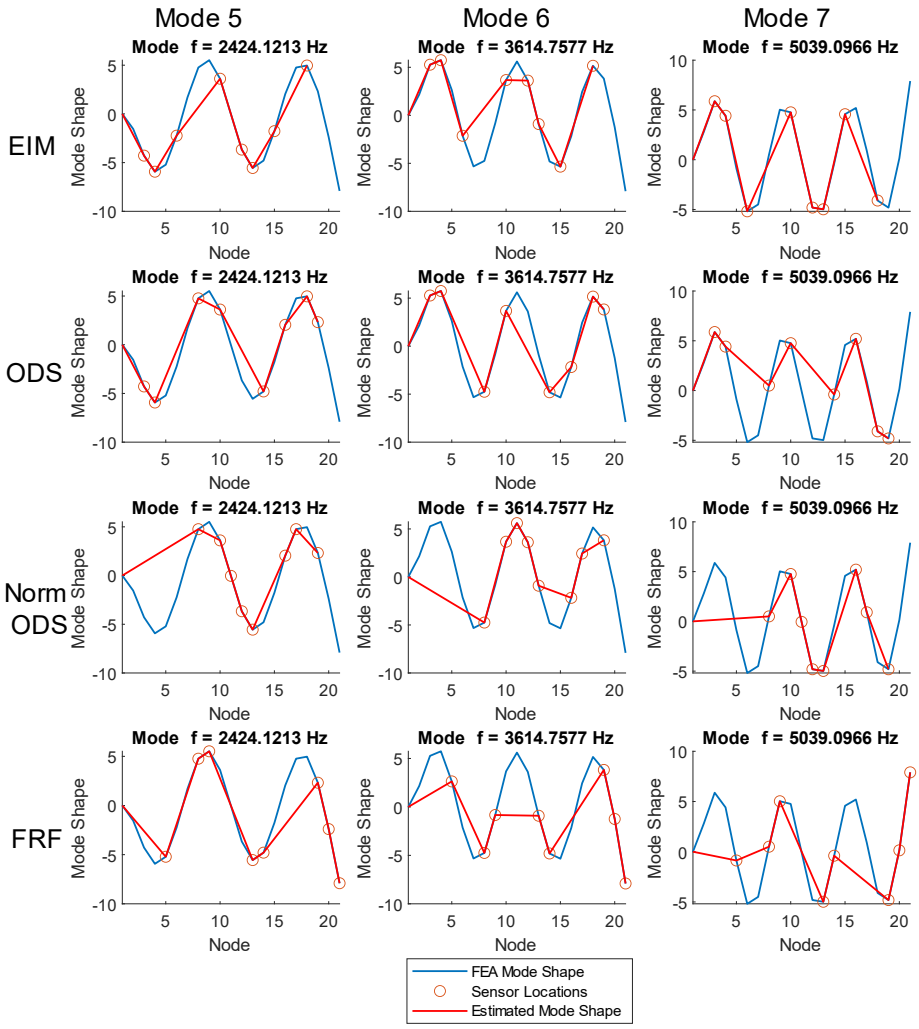

Fig. 6.6 Sensor locations from chirp excitation for mode 5 through 7

It is interesting to note that although the FRF method had the best R2 score and the normalized ODS method had the lowest MSE, the ODS method performed best when estimating the natural frequencies. Additionally, it was observed that the average FRF based sensor selection was highly reliant on the windowing being used. Since a FRF must be computed before exporting the FRF data to the RFR script, changing the windowing can impact the sensor locations returned when using the FRF based sensor selections. In this case, the same windowing was used for all signals.

Table 6.3 Natural frequencies, 1D beam, chirp excitation

FEA	EIM		ODS		norm ODS		Avg FRF	
f (Hz)	f (Hz)	% Err	f (Hz)	% Err	f (Hz)	% Err	f (Hz)	% Err
42.8	49.4	15%	49.5	16%	52.0	21%	51.1	19%
–	230.1	–	230.1	–	239.2	–	230.6	–
268.3	279.7	4%	279.7	4%	291.8	9%	280.1	4%
–	724.8	–	724.8	–	723.6	–	724.5	–
750.5	767.3	2%	767.4	2%	765.8	2%	767.3	2%
1468.7	1463.0	0%	1463.0	0%	1463.3	0%	1463.5	0%
–	–	–	–	–	–	–	2098.9	–
2424.1	2401.0	1%	2401.0	1%	2400.7	1%	2399.8	1%
3614.8	3539.1	2%	3539.1	2%	3537.1	2%	3525.5	2%
5039.1	4833.1	4%	4832.8	4%	–	–	–	–
5261.4	–	–	–	–	–	–	–	–
6696.0	6317.7	6%	6317.1	6%	6306.8	6%	6459.3	4%

6.3 2D Plate

For the case of the 2D plate, a linear chirp excitation was applied to the FEA model of the plate. This excitation was conducted as a transient analysis in ANSYS. The location of the load applied, and the corresponding points where acceleration data was exported were chosen such that the conditions could be replicated in the lab for physical verification of the sensor selection. After conducting the transient analysis, acceleration data was exported from ANSYS and reformatted for input into the same RFR algorithms outlined in the previous section.

6.3.1 FEA Transient Analysis

The FEA model used for this analysis is identical to the plate used in Chapter 4 for the modal analysis, with the same geometry, material properties and mesh parameters. A chirp signal with the characteristics shown in Table 6.4 was used to excite the plate at a node located at (151.97 mm, 0 mm, 441.2mm). This coordinate was chosen to match the real-world plate and load cell most closely as set up in the lab.

Table 6.4 Chirp excitation characteristics for 2D Plate

Property	Value
Duration	0.5s
Frequency Range	20-15000 Hz
Frequency Sweep	Linear
Magnitude	10 N
Sampling Frequency	20 kHz

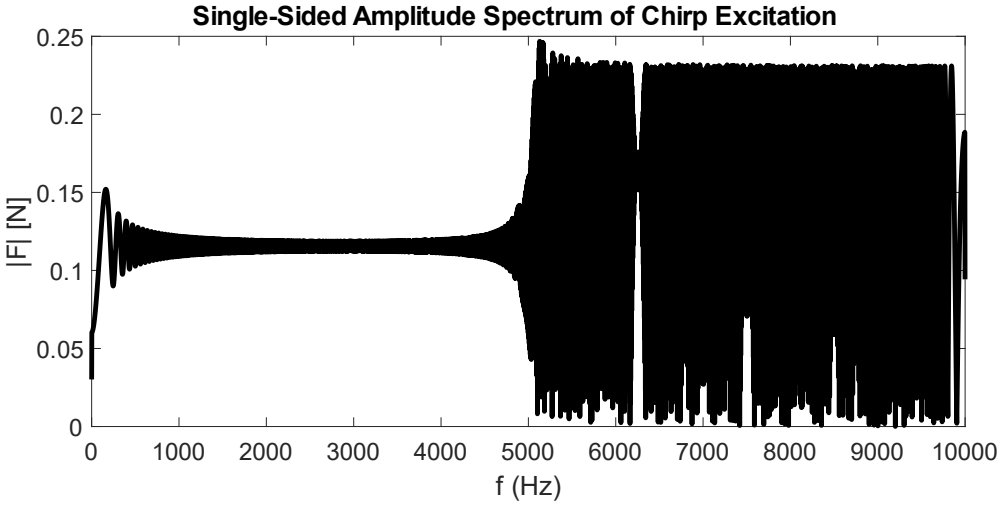

Fig. 6.7 Chirp Excitation Signal imported into ANSYS

The load as applied to the plate is shown in Fig. 6.8 overlayed on the FEA model. The transient analysis was run with a timestep frequency of 20 kHz to match the excitation signal. A 20 kHz sampling rate was chosen so that Nyquist frequency of 10 kHz would be large enough to capture the previously selected modes for the plate.

After completing the analysis in ANSYS, the acceleration data in the Y direction was exported for each node on the Y=6.1 mm plane. This plane represented the face of the plate opposite the point of load application where the accelerometers would be placed on the physical plate. This data was exported using an APDL script (Appendix D: Transient Analysis APDL Export Script); however, technical issues either with the script or ANSYS prevented the entire time-history of accelerations from being extracted. The dataset up to the last completed time step was then imported into MATLAB and processed for input into the RFR sensor selection script.

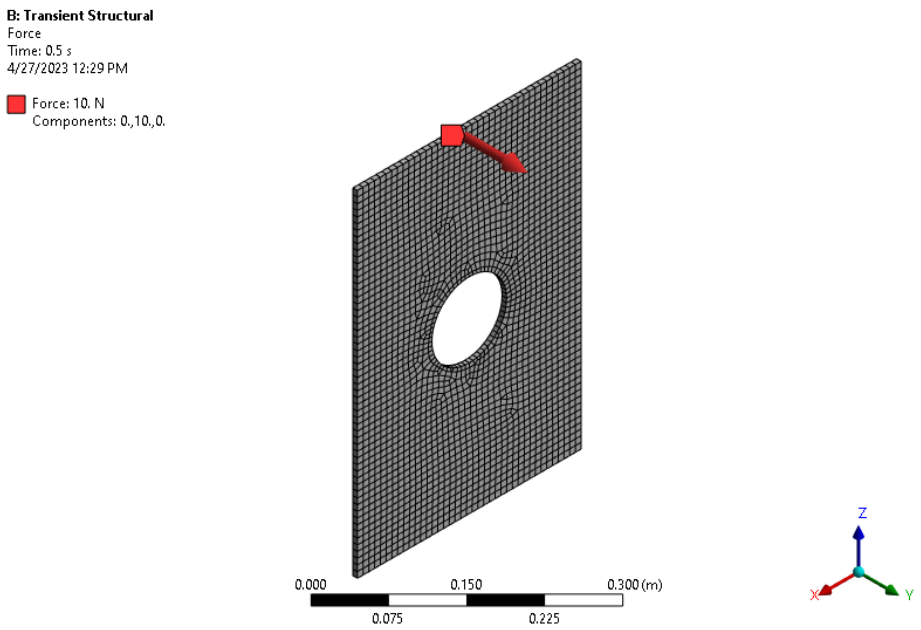

Fig. 6.8 Point transient load as applied to plate. Note the bottom edge is fully constrained.

6.3.2 FEA Data Processing

After importing the data into MATLAB, the data was processed according to the code found in Appendices E and F. The data was formatted for use with the *modalfrf* function. The FRF was calculated using a Hamming window of size 100 and 50 overlap. The small window size was chosen due to the FEA exported dataset containing only 200 time steps. The calculated FRF function is shown in Fig. 6.9.

As the FRF function of a numerical simulation, the FRF should be relatively ideal; however, this is not the case. Although coherence dips as expected at resonances and anti-resonances, it is relatively low. This is likely a direct result of the relatively low number of time steps able to be exported from ANSYS since coherence in the high frequency range is very good. The better coherence in the high frequency range may also be a result of a lack of uniformity in the excitation signal, as the chirp signal has a much higher amplitude in the high frequencies compared to the low frequencies.

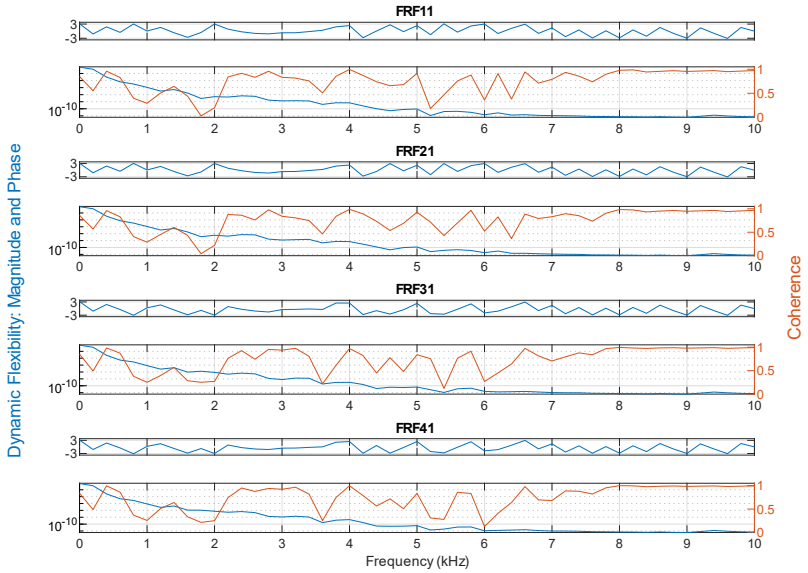

Fig. 6.9 FRF Function for first 4 DOFs of the plate under chirp excitation

6.3.3 Sensor Selection

Sensor selection was conducted using the FEA modal analysis results for the EIM sensors and the transient FEA simulation for the RFR derived selection methodologies. For each selection technique, 7 sensor locations were requested. The number of accelerometers was constrained by equipment availability for experimental verification.

Unlike the 1D beam case, the EIM algorithm was not near instantaneous, and speed can vary depending on the hardware used. For the following section, all programs were run on a desktop computer running 64-bit Windows 11. The computer was equipped with a 16-core AMD 7950X at 5.4GHz and 64 GB DDR5 RAM. Data was stored locally on a solid state drive. As the algorithm eliminates only one node on each iteration, requesting 7 sensor locations requires n-7 iterations, where n represents the total number of nodes. The number of iterations and therefore time required is directly proportional to the number of nodes in the model. For this case, 6,839 nodes were processed out of the total 17,106 nodes, representing the nodes on the surface where sensors could be placed. The nodes correspond to approximately 5mm spacing when represented physically on the plate. The EIM algorithm for 6,839 FEA nodes and 14 vibrational modes took 3.5 minutes to run in MATLAB R2022b , discounting data ingest time. Although techniques exist to improve the run time of the algorithm, such as eliminating multiple low ranking nodes per iteration, this can degrade the accuracy of the final result [1]. To allow for

a more accurate comparison between sensor selection methodologies, the best case of a purely iterative implementation of EIM was used.

In comparison to the EIM method, the RFR methods were not measurably slower computationally compared to the 1D beam case, taking just 2.17 seconds to run all three RFRs. It must be cautioned that the speed of the RFR method is mitigated by the longer ANSYS transient analysis that must be computed before the regressor can be used. Both the algorithms and FEA were run on the same computer previously described. The data in Table 6.5 should be taken to be illustrative, and all times were typical with only minor run-to-run variance. The table below does not take into account data export from ANSYS or import/reformatting done in MATLAB prior to the algorithm run. MATLAB code was run in R2022b and the RFR was implemented using scikit-learn version 1.1.2 on Python 3.9.13.

Table 6.5 Algorithm time comparison

Algorithm	Algorithm Time (s)	FEA Type	FEA Time (s)	Total Time (Excluding FEA data export and Ingest)
EIM	210	Modal (60 Modes)	6	216
RFR-FRF	2.17 (combined)	Transient (5e-5 s timestep, 0.5s simulation)	3114	3116.7
RFR-ODS				
RFR-nODS				

The following figures display the chosen sensor configurations. They are oriented looking down the Y-axis ‘through’ the plate. Looking at physical accelerometers in these configurations would appear flipped.

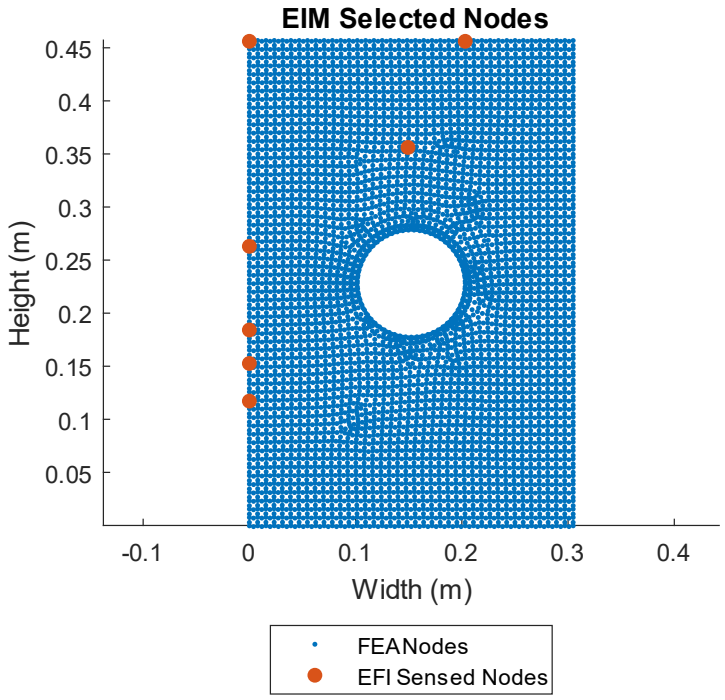

Fig. 6.10 Sensed nodes for EIM Method

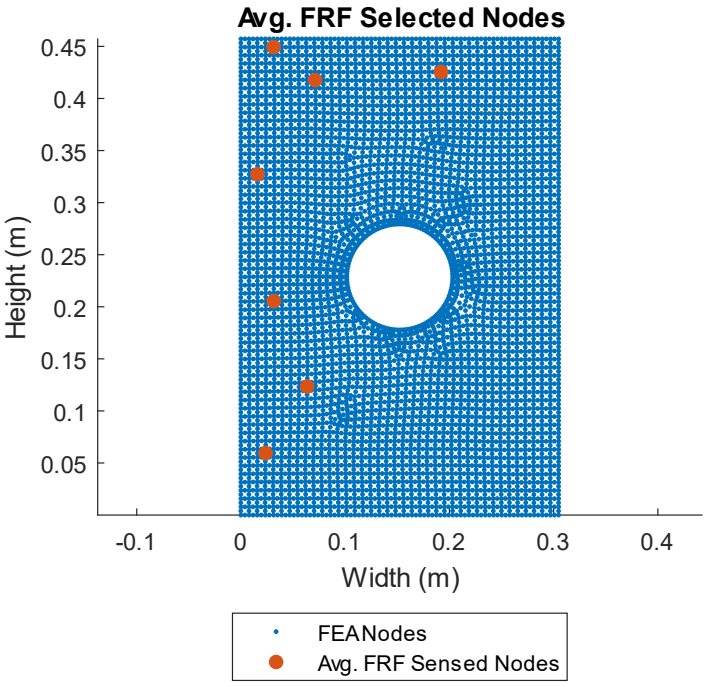

Fig. 6.11 Sensed nodes for average FRF method

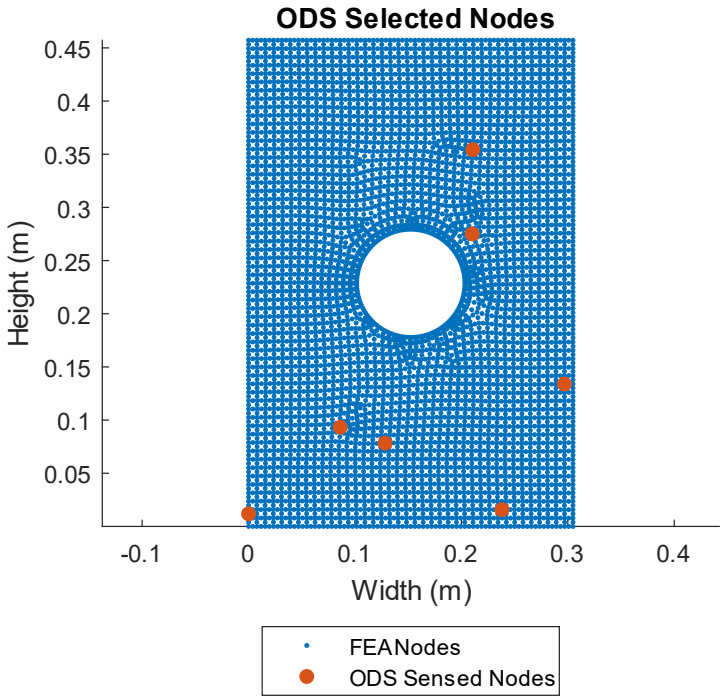

Fig. 6.12 Sensed nodes for ODS method

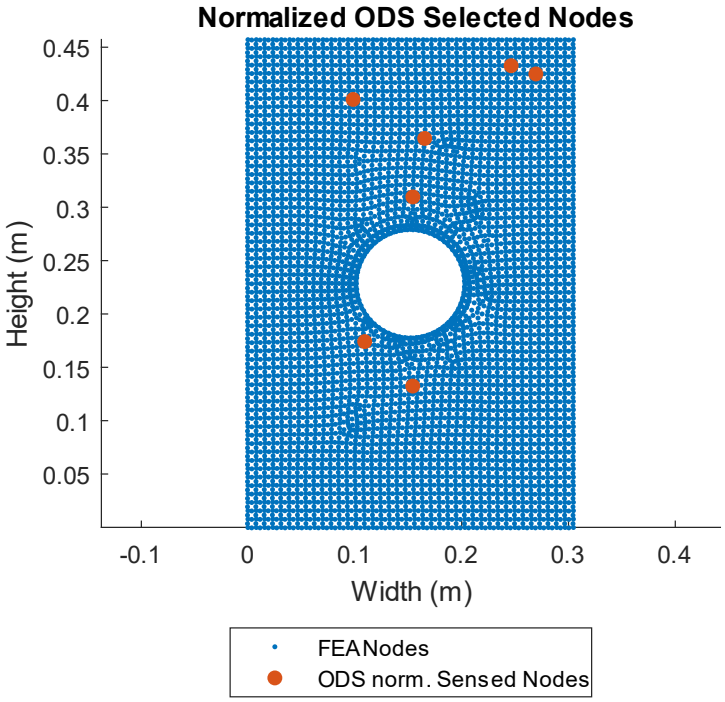

Fig. 6.13 Sensed nodes for normalized ODS method

Visually, the average FRF and EIM method appear the most similar, with accelerometers distributed relatively evenly along the vertical axis, but clustered more towards the top of the

plate. The normalized ODS nodes are—like the first two methods—concentrated towards the top of the plate; however, they align close to the central axis of the plate near the hole. The ODS sensed nodes are clustered around the base of the plate. Intuitively this sensor distribution is the least expected, as acceleration should be lowest near the base of the plate. Lower accelerations imply lower sensitivity to excitation and therefore would be expected to be less capable of measuring the excitation.

6.3.4 Natural Frequency Extraction

An FRF was calculated from the extracted FEA acceleration data for each set of nodes and the FRF and coherence was plotted for the first 4 nodes in each sensor set. Next, the natural frequency for each sensor selection methodology was extracted using the same *lsrf* algorithm in the *modalfit* function. The natural frequencies extracted from the dataset were then compared to the natural frequencies extracted from ANSYS, with particular attention paid to the transverse bending natural frequencies. The code used to generate this data is presented in Appendix E

It is important to recall that the extracted mode shapes from ANSYS are only the mode shapes at the transverse bending modes. As this is the dataset used in the EIM method, the EIM sensors should be expected be positioned to better capture these modes.

Overall coherence is relatively low in the low end of the frequency spectrum, with most sensor locations. Given the short sample time able to be extracted from the FEA model of 9.5E-3 seconds and the high sample rate of 20 kHz, the lowest frequency that could be captured in that data is approximately 104 Hz, while the highest is 10 kHz. These limitations can partially account for the low coherence in the beginning of the FRF, and coherence there is expected to improve if the same sensor locations were selected with a larger dataset.

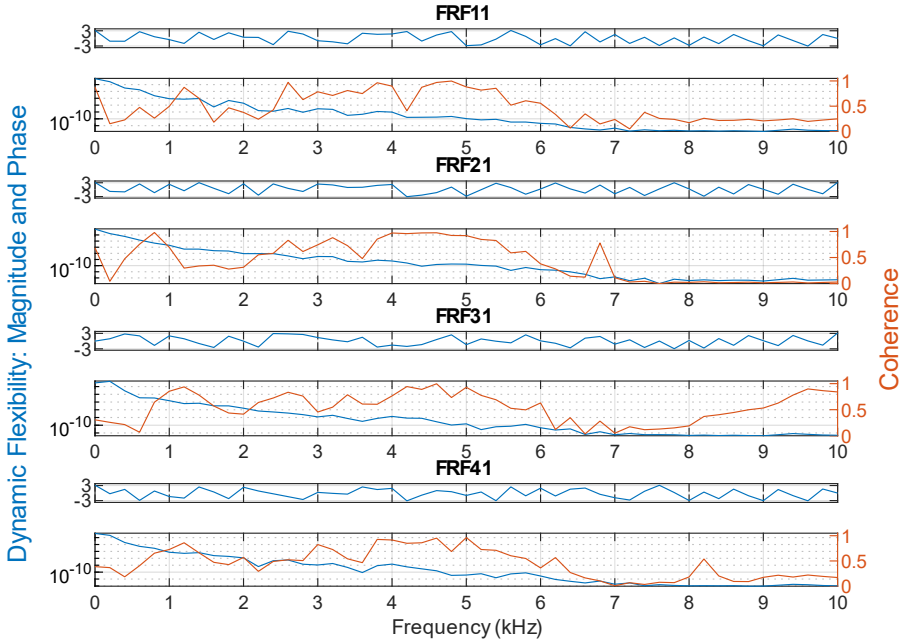

Fig. 6.14 EIM method FRF function (node 1-4)

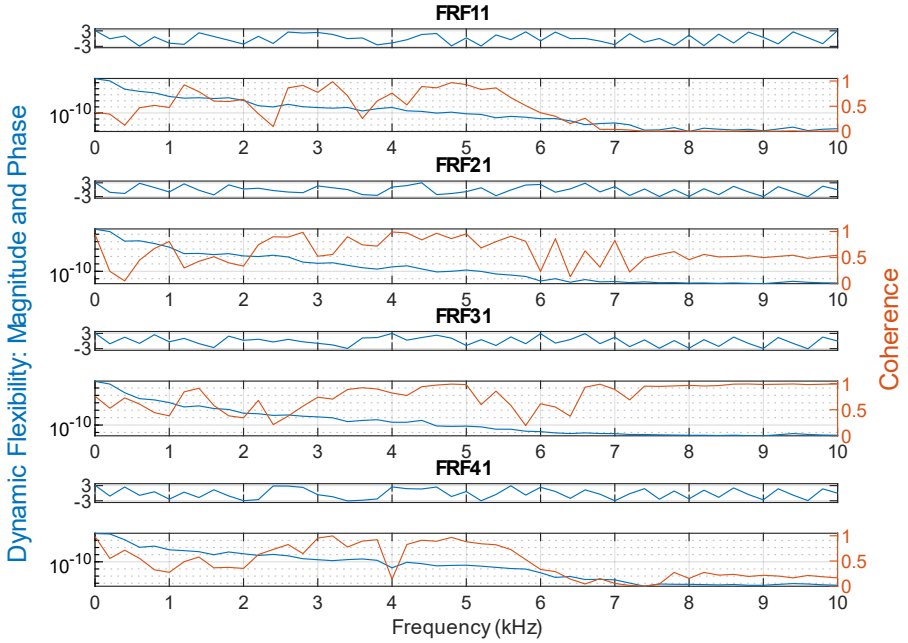

Fig. 6.15 ODS Method FRF function (node 1-4)

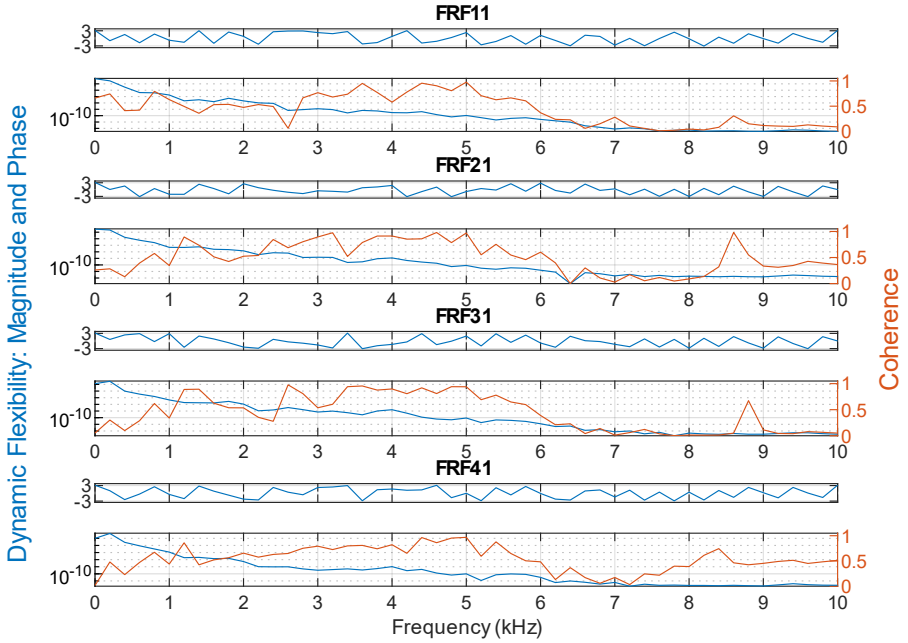

Fig. 6.16 Normalized ODS method FRF function (node 1-4)

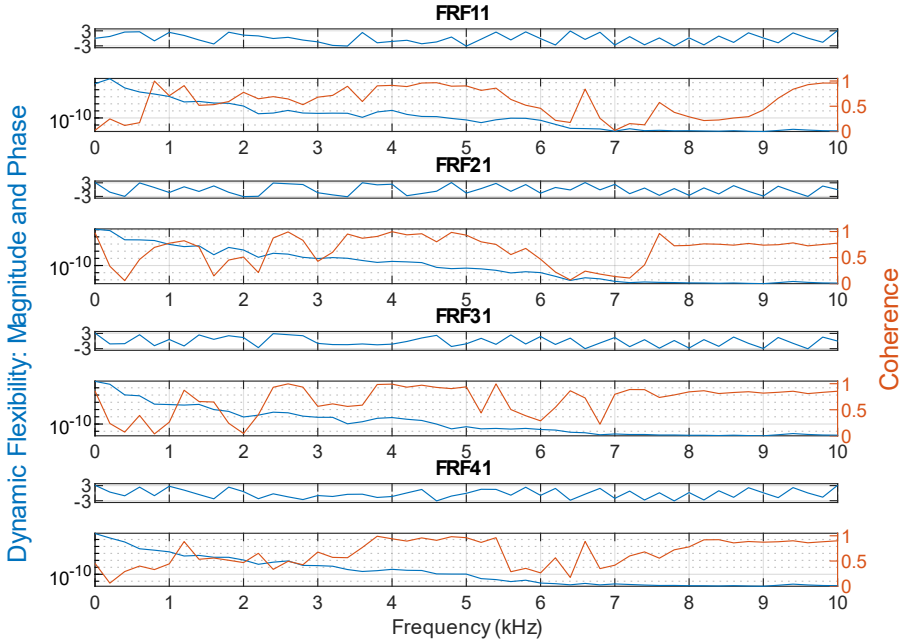

Fig. 6.17 Average FRF method FRF function (node 1-4)

Table 6.6 Natural frequencies of 2D plate. Non bending frequencies indicated with ( )

FEA	EIM		ODS		norm ODS		FRF	
f (Hz)	**f (Hz)**	**% Err**	**f (Hz)**	**% Err**	**f (Hz)**	**% Err**	**f (Hz)**	**% Err**
24.057	–	–	–	–	–	–	–	–
147.39	202.5	37%	–	–	–		201.0	45%
(363.18)	–	–	–	–	–	–	–	–
420.73	406.2	3%	419.0	1%	418.0	1%	425.0	1%
(591.11)	654.5	11%	655.6	11%	–	–	656.6	11%
843.47	916.2	9%	–	–	865.8	3%	–	–
(917.41)	–	–	902.6	2%	–	–	915.7	0%
1343.8	–	–	–	–	–	–	–	–
1433.1	–	–	–	–	–	–	–	–
1518.1	–	–	–	–	–	–	–	–
(1881.4)	–	–	1837.1	2%	1715.2	9%	1818.4	3%
1973.6	–	–	–	–	1924.3	3%	–	–
2231.9	2284.6	2%	–	–	–	–	–	–
2458.8	–	–	–	–	–	–	–	–
2998.9	–	–	–	–	–	–	–	–
(4037.4)	–	–	4063.2	1%	3782.0	6%	–	–

As shown above in Table 6.5, compared with the results of the beam, all the sensor selection methods appear to perform worse. Across all sensor methodologies, only 5 of the first 14 transverse bending natural frequencies are captured with less than 5% error. If we expand the selection criteria to include all natural frequencies, and not just those in bending, then across all methodologies 7 natural frequencies are captured with less than 5% error.

One limitation of the approach used here, as mentioned previously, is the small size of the time domain acceleration dataset that was able to be exported from ANSYS. Given limitations in the *lsrf* modal extraction algorithm that limit the number of frequencies capable of being returned with small window sizes i.e., only 5 frequencies were output despite requesting 7, these results are not sufficient by themselves to assess the effectiveness of the various sensor methodologies. The ODS methodology had the lowest combined error of the returned natural frequencies, while EIM returned the fewest non-bending frequencies. This is expected behavior for EIM as the dataset fed into the selection algorithm included only modes that were primarily transverse bending.

6.4 Discussion

Both the ODS RFR method and the traditional EIM method appear to perform similarly for the 1D beam, yielding similar natural frequencies to each other in both loading conditions as well as having lower percent error than the FRF and normalized ODS method. An advantage of both methodologies over the FRF based RFR method is that they are insensitive to windowing during sensor selection. This reduces the amount of preprocessing that needs to be conducted and removes a step in the process where error could be introduced by choosing an inappropriate window. Additionally, conducting a transient analysis in FEA for large or complex geometries is non-trivial, and can greatly increase the time and computational requirements for sensor selection, a disadvantage the EIM method does not have.

Windowing is sensitive to the excitation signal and therefore accuracy of the extracted modes is sensitive to both the excitation signal and the windowing. For the 1D beam, both the chirp and white noise excitation yielded similar extracted natural frequencies, although the chirp frequency does appear to yield more consistent results with fewer non-predicted frequencies. The chirp excitation yielded a more accurate first natural frequency, although accuracy was still poor with a best-case 28% error. The ODS based sensors appeared to perform slightly better than the EIM based sensors, but this is offset by being unable to predict the 7^{th} natural frequency.

Due to the small dataset used for the 2D plate, it is more difficult to assess the effectiveness of the various methodologies. Although all appear capable of predicting some frequencies with greater than 5% accuracy, the sensed frequencies are scattered. Expanding the time history dataset would likely improve the results.

7 Experimental Validation

7.1 Chapter Summary

A physical plate identical in dimensions and material to the plate discussed in the previous chapter was subjected to modal testing using both traditional and machine learning derived sensor configurations. Six sensor configurations were tested, the four discussed in the prior chapter—effective independence, RFR Average FRF, RFR ODS, and RFR normalized ODS—and two additional configurations where the sensors were placed in a simple grid pattern. The plate was then excited, and data recorded to determine the natural frequencies of the plate. The following chapter outlines the experimental methodology, presents the collected data, and analyzes the effectiveness of the various sensor techniques.

7.2 Methodology

For the experiment, the plate was welded along the boundary condition to C-channel, that was then bolted through the table as indicated in Fig. 7.1. The plate's welded support structure and the modal shaker support structure were both attached to a steel fixturing table with bolts. The modal shaker was mounted to the corresponding trunnion, which in turn was bolted to the support structure.

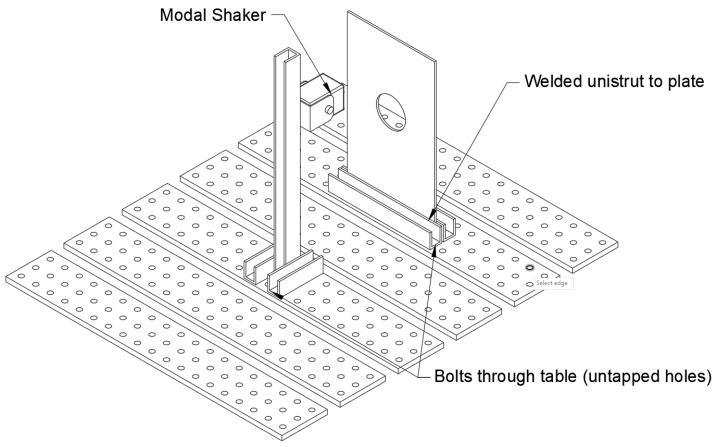

Fig. 7.1 Experimental configuration

The modal shaker was attached to plate via a stinger with a force sensor mounted to the end of the stinger and glued to the plate. Seven accelerometers were mounted to the opposite side of the plate with wax. A total of 6 different sensor configurations were trialed:

- Effective Independence Method (EIM)
- RFR selecting for avg. FRF (RFR-FRF)
- RFR selecting for ODS (RFR-ODS)
- RFR selecting for normalized ODS (RFR-nODS)
- Large distributed grid (LDG)
- Small distributed grid (SDG)

The two grid sensor configurations were chosen as representative of a non-algorithmic sensor placement technique, with accelerometers placed to cover half of the plate (LDG) and a quarter of the plate (SDG). For the LDG sensor placement, the sensors were spaced 100mm from each other. In the SDF configuration, sensors were placed with 50mm spacing. The placement configuration for the other placement methodologies can be found in the prior chapter. Photographs of the six sensor configurations are provided in the following figures:

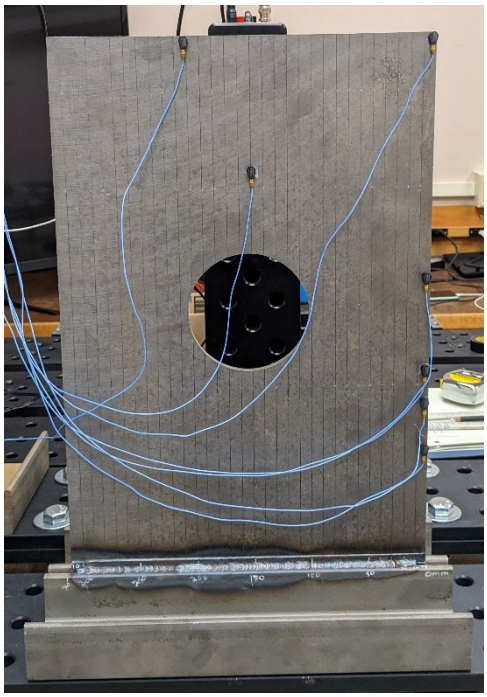

Fig. 7.2 EIM sensor placement

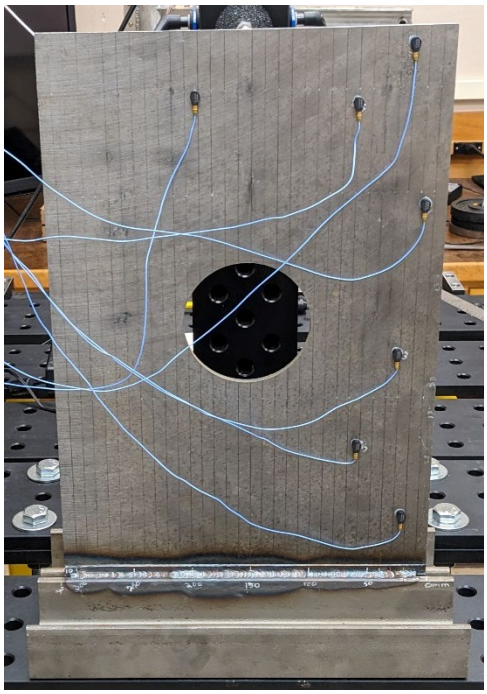

Fig. 7.3 RFR-FRF sensor placement

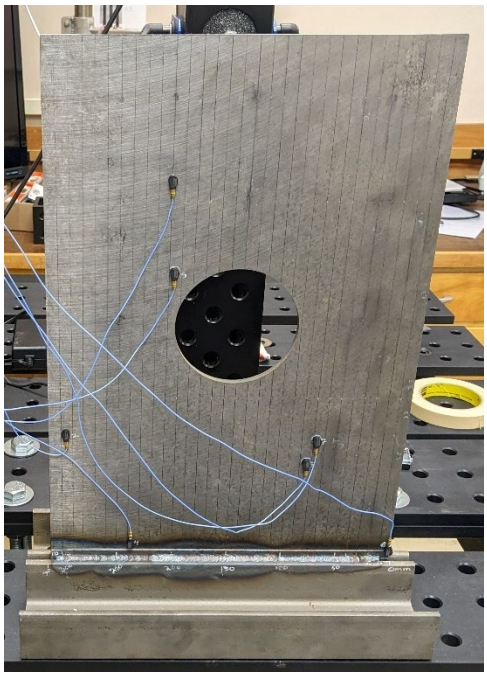

Fig. 7.4 RFR-ODS Sensor Placement

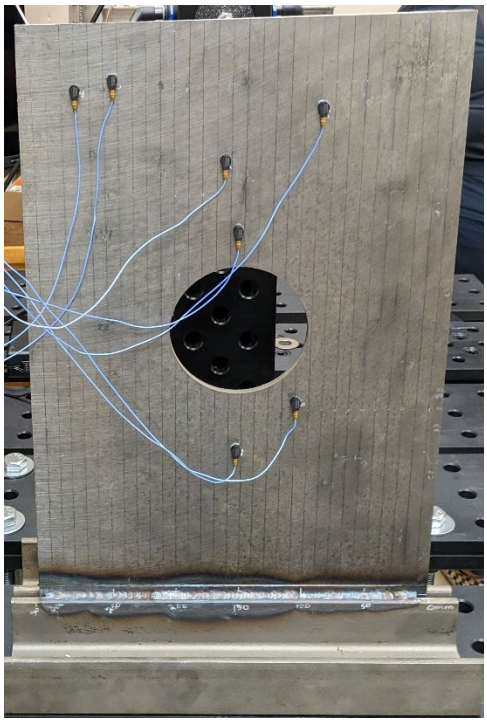

Fig. 7.5 RFR-nODS sensor placement

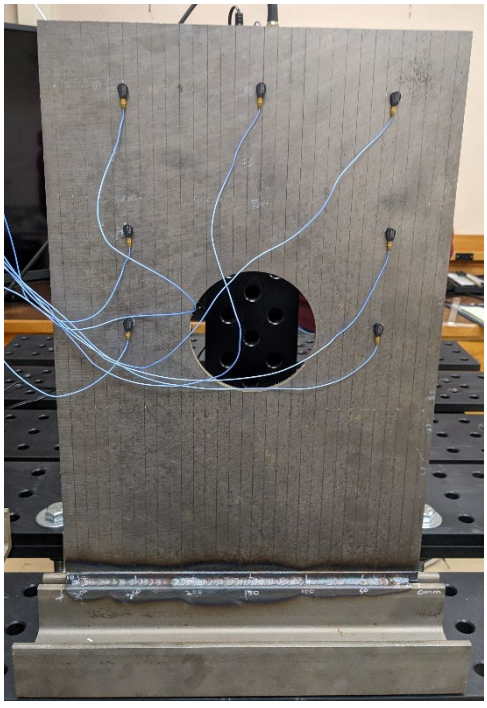

Fig. 7.6 LDG sensor placement

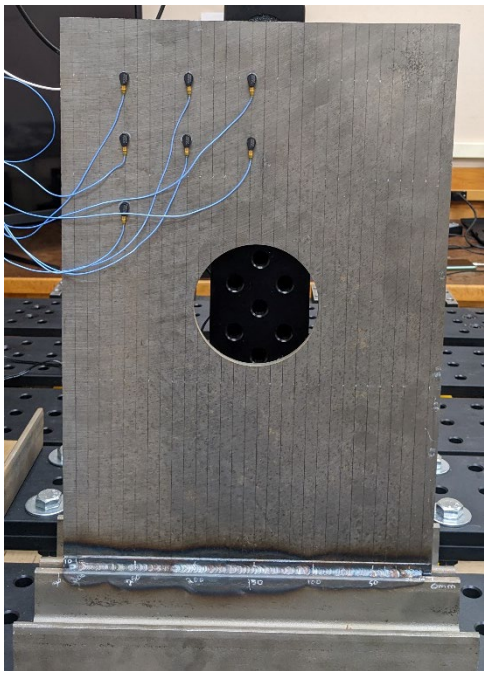

Fig. 7.7 SDG sensor placement

The sensor and shaker were mounted so the force was applied at a height of 441 mm from the fixed plate base and at 152 mm from the edge of the plate, approximately in the center. The point of application for the excitation was chosen to match the FEA plate load condition as closely as possible, though in real world testing it is not possible to apply the excitation to a single node. A photograph of the modal shaker, force sensor, and plate as set up for the experiment is shown in Fig. 7.8.

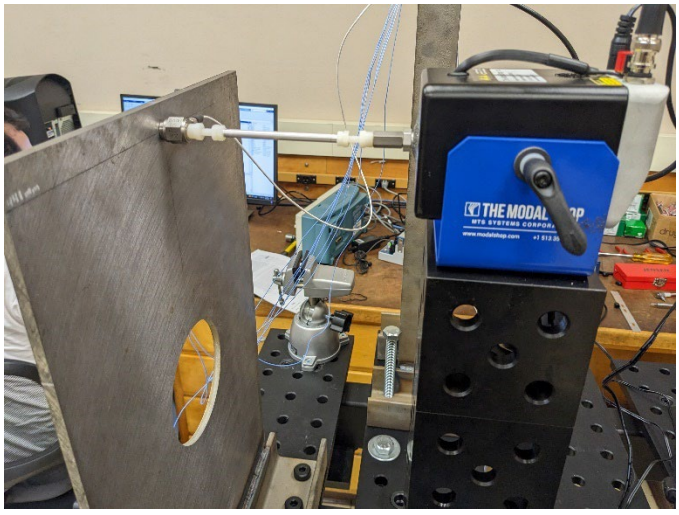

Fig. 7.8 Photograph of modal shaker as set up

7.2.1 Excitation Signal

For each sensor configuration, the beam was excited with a linear chirp function. The chirp function was chosen due to being an easier signal to process with the *modalfit* and *modalfrf* in MATLAB. Time constraints prevented the collection of data from multiple excitation signals. The characteristics of the chirp function and single-sided amplitude spectrum are available in Table 7.1 and Fig. 7.9 respectively. The amplitude of the signal was chosen to ensure that the acquired data was not clipped, and the excitation frequency range was chosen as the primary frequencies that would ideally be observed were all below 2 kHz. Sampling frequency was determined by hardware and software constraints.

Table 7.1 Excitation Signal

Property	**Value**
Duration	4s
Frequency Range	0-2000 Hz
Frequency Sweep	Linear
Magnitude	0.1 N
Sampling Frequency	8533 Hz

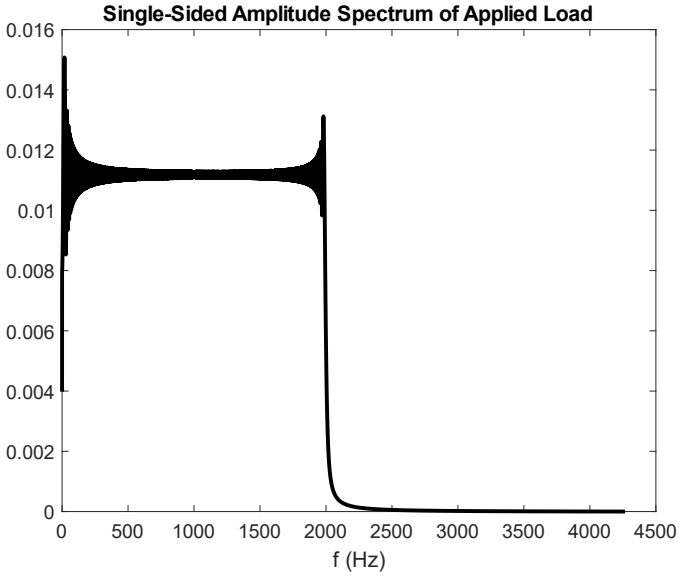

Fig. 7.9 Experimental excitation signal

7.2.2 Data Acquisition

Excitation force data was collected with a PCB Piezotronics 208C01 ICP force sensor. Acceleration data was collected via 7 PCB Piezotronics model 352A24/NC ICP accelerometers. The force sensor and accelerometers were routed through a PCB Piezotronics model 482C15 signal conditioner before plugging into two NI 9234 4-channel analog input module, one channel per signal. The modal shaker used was a K2007E01 Mini SmartShaker from The Modal Shop. The excitation signal was output through a NI9260 analog output module. Both the sensor and signal modules were interfaced with a MATLAB script through a NI CDAQ-9178 over USB. The MATLAB script performed the signal generation and response recording. The analog signal was sampled at a rate of 8533 Hz. The NI9234 provides built in, automatic, anti-aliasing filtering for all recorded excitations.

7.2.3 Postprocessing

For each sensor configuration, 10 samples were acquired. Sensor data was logged for each sample and analyzed for any experimental error. In three cases—RFR-FRF, nODS, and SDG, one channel of the data in one sample was missing a value. This is potentially due to the actual sampling rate of the hardware being 8533.3 recurring and not being fully captured by MATLAB. In these cases, the data from that channel for that sample was discarded.

The remaining data for each channel was then averaged across the 10 samples. A low-pass sharp cutoff filter was applied to the dataset from each sample, with a cutoff frequency of 2000 Hz and a stopband attenuation of 100 db. The full MATLAB code for the postprocessing, FRF calculation, and feature extraction is presented in Appendix I

7.3 FRF

For the FRF calculation, *modalfrf* in MATLAB was again used to calculate the H_1 FRF. A Hamming window of size 4267 with a 90% overlap was used. The window size was chosen to allow at least 10 cycles of the lowest target frequency (24 Hz) to be observed within each window. A high overlap was chosen to improve frequency resolution. The FRF and coherence for each of the sensors for each configuration are shown in the following figures.

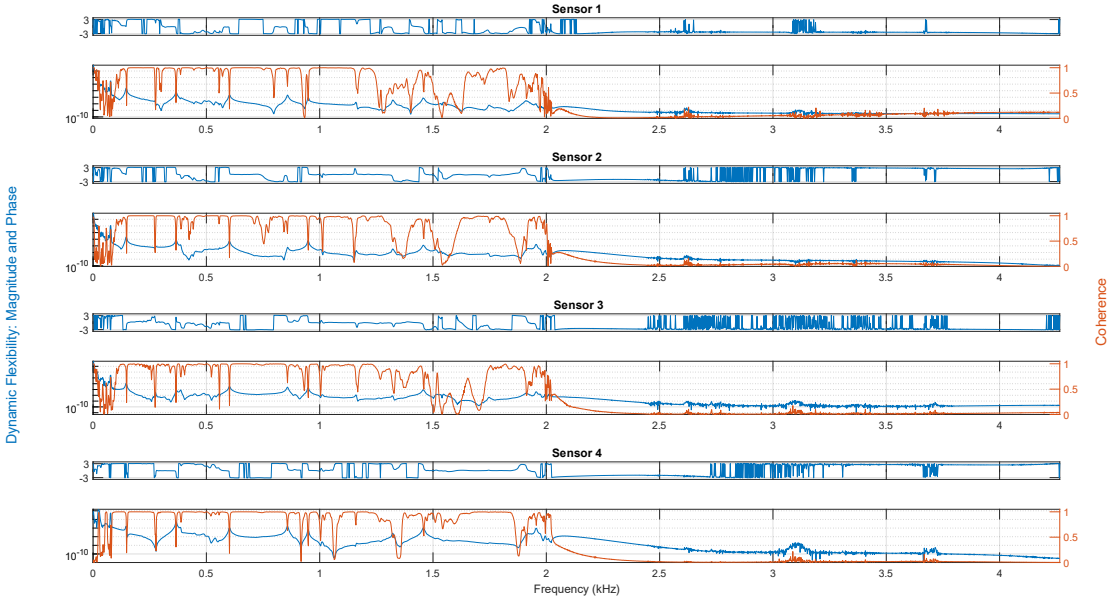

Fig. 7.10 EIM sensor 1 through 4

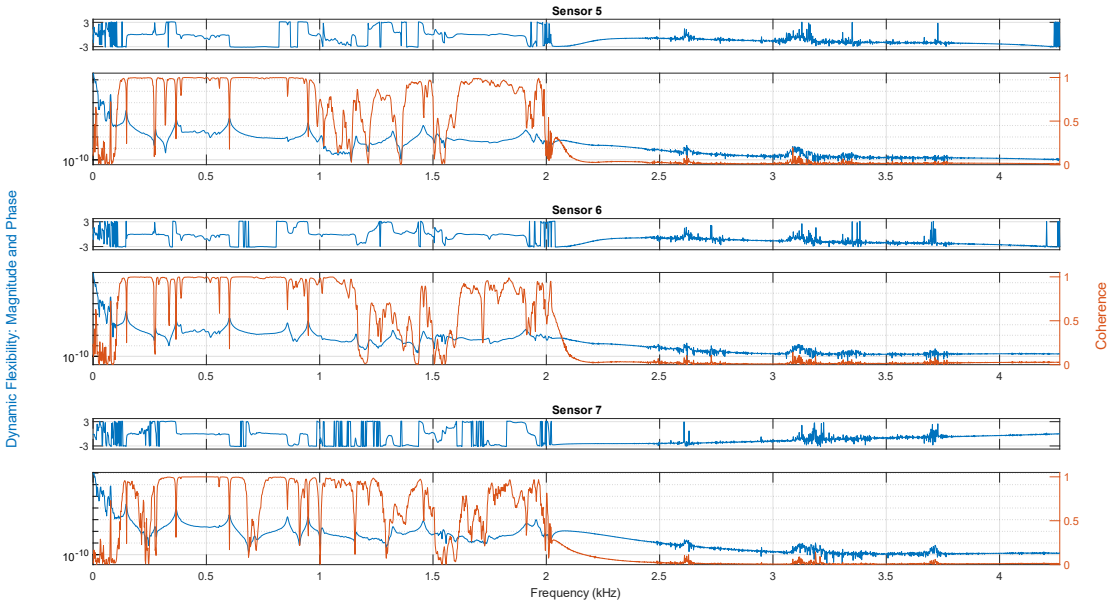

Fig. 7.11 EIM sensor 5 through 7

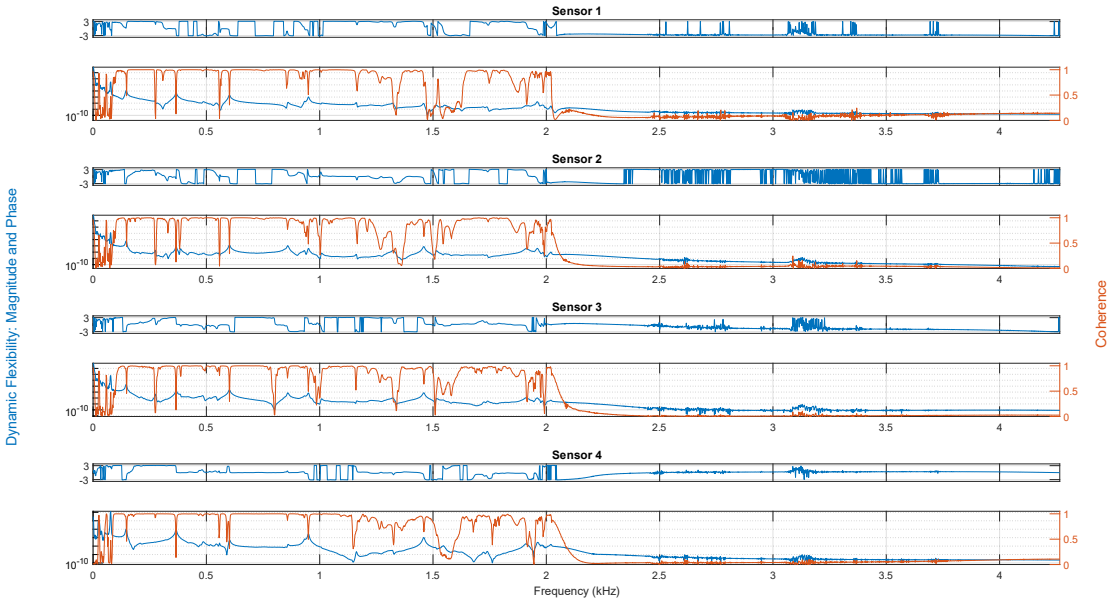

Fig. 7.12 RFR-FRF sensor 1 through 4

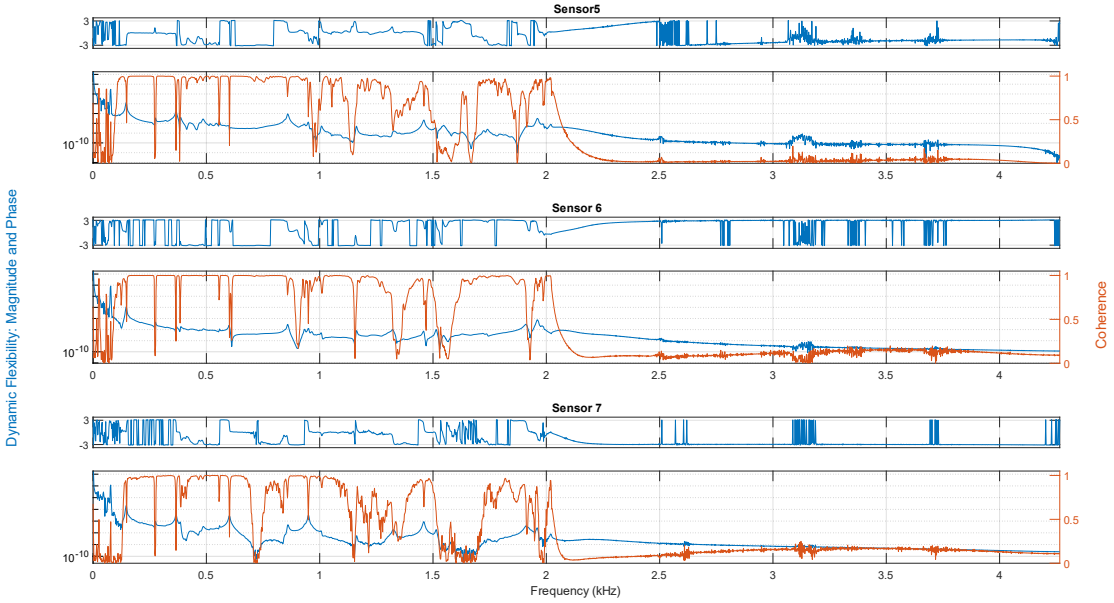

Fig. 7.13 RFR-FRF sensor 5 through 7

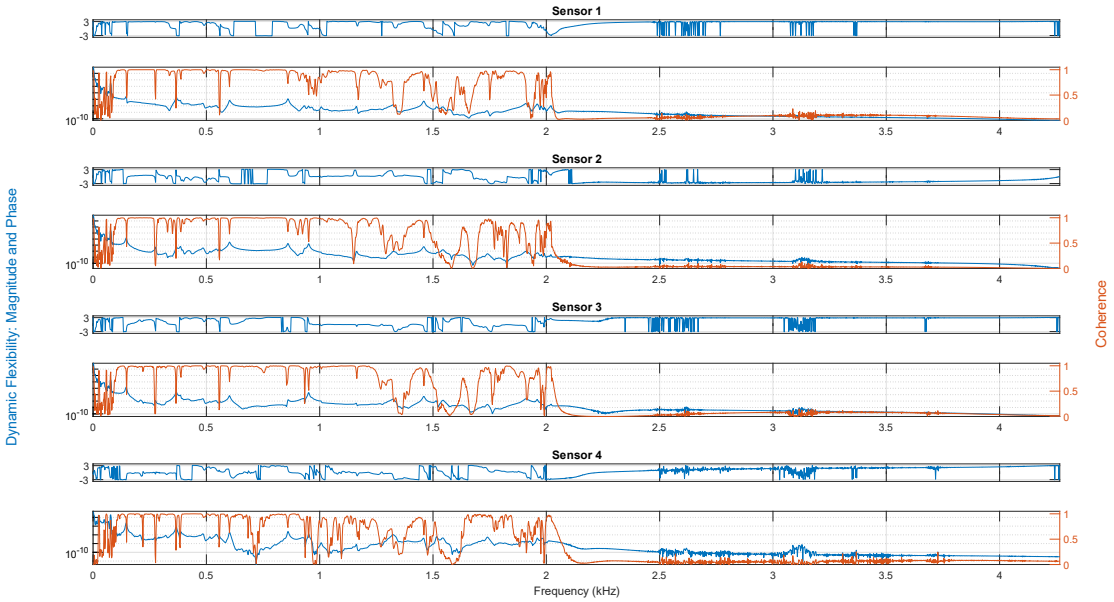

Fig. 7.14 RFR-ODS sensor 1 through 4

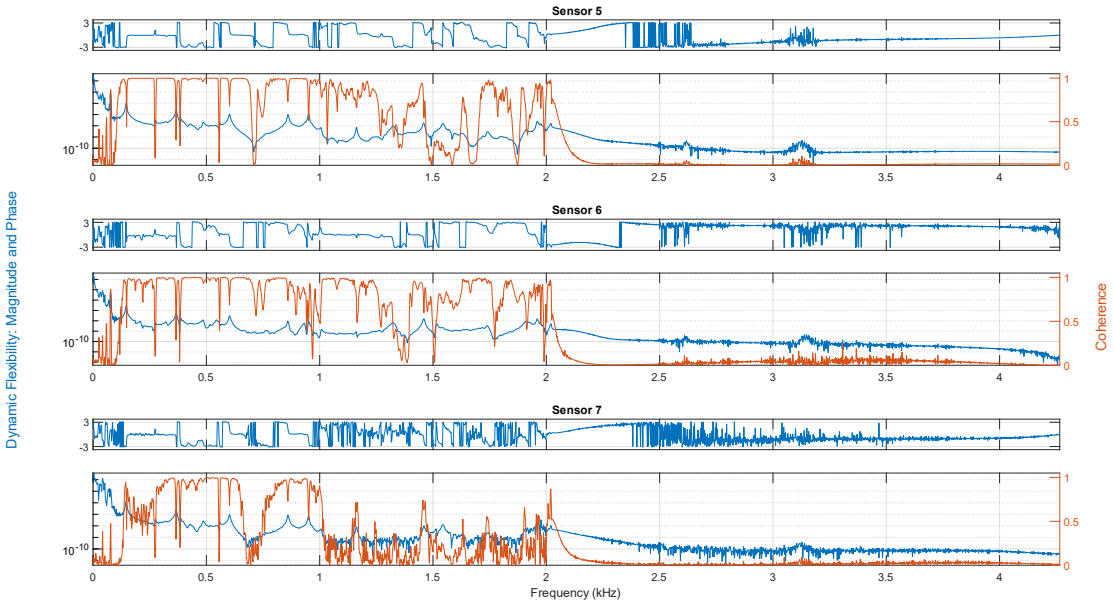

Fig. 7.15 RFR-ODS sensor 5 through 7

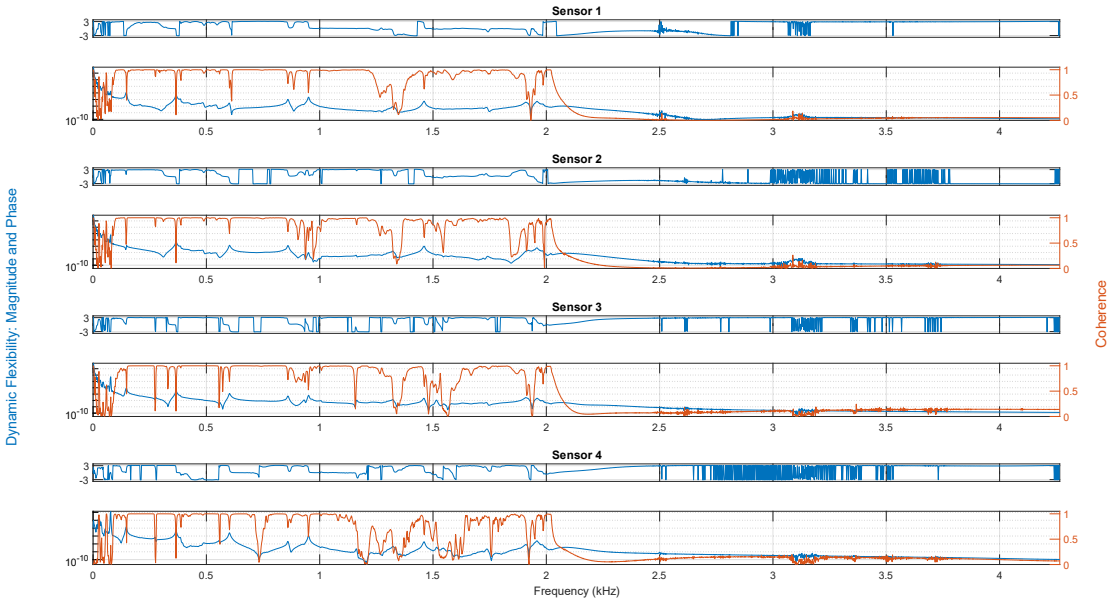

Fig. 7.16 RFR-nODS sensor 1 through 4

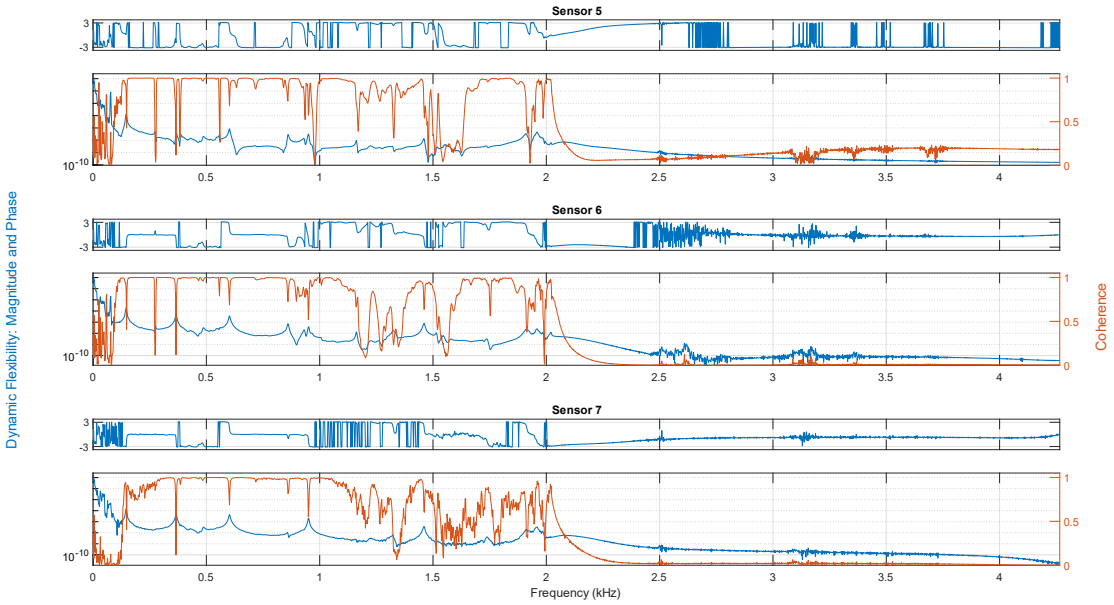

Fig. 7.17 RFR-nODS sensor 5 through 7

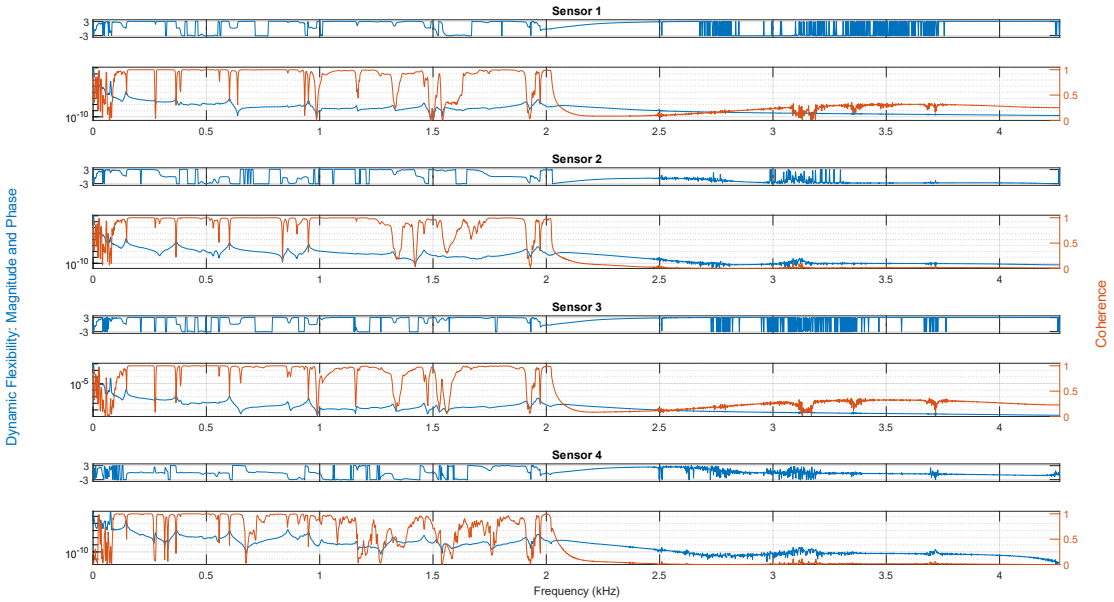

Fig. 7.18 LDG sensor 1 through 4

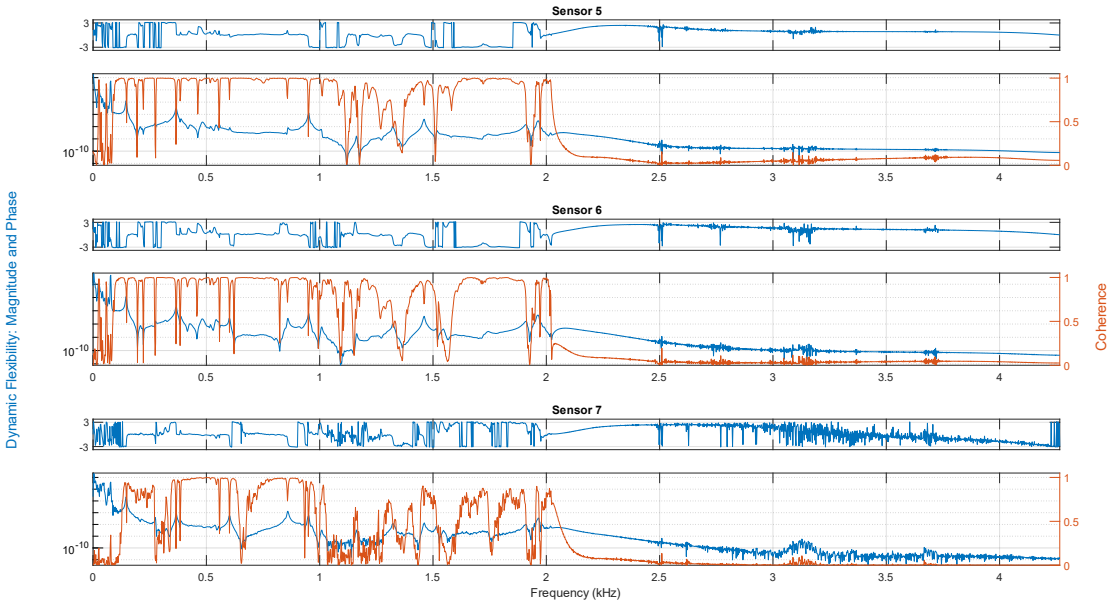

Fig. 7.19 LDG sensor 5 though 7

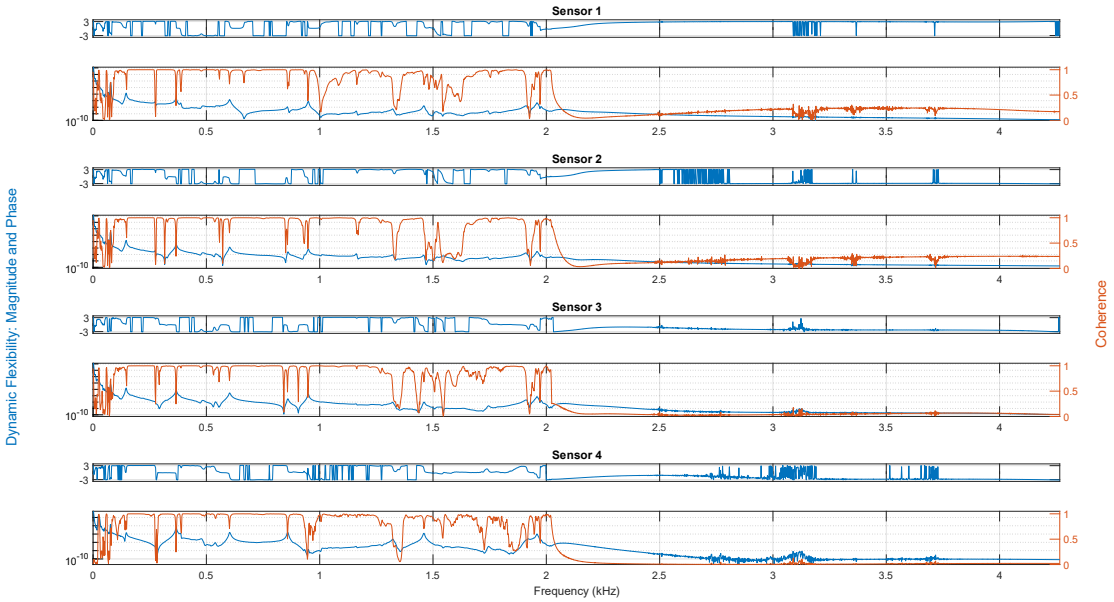

Fig. 7.20 SDG sensor 1 through 4

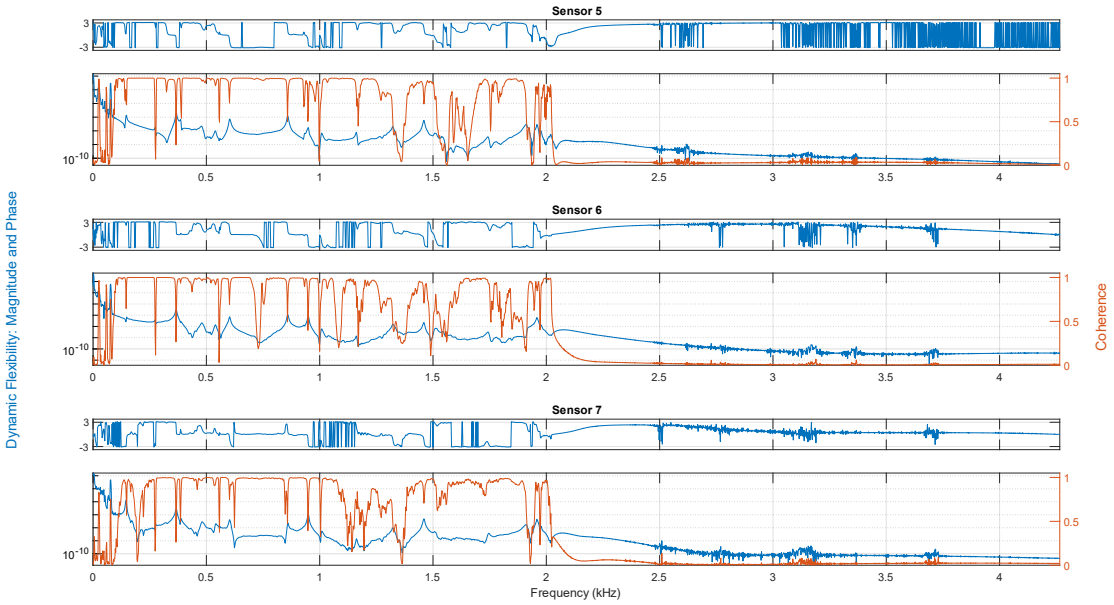

Fig. 7.21 SDG sensor 5 through 7

Some features are shared across all the FRF functions. The first is the expected sharp drop off in coherence after 2 kHz. This is a direct result of the lowpass filter applied to the signals before calculating the FRF. The other feature is the relatively poor coherence between 0 and

approximately 150 Hz. This lack of coherence could be a result of experimental configuration since the shaker may not have sufficiently excited these low frequencies. Additionally, a stuttering behavior was observed when the excitation signal was triggered during testing, which may impact the low frequency response.

In the case of the RFR-ODS sensors (Fig. 7.14, Fig. 7.15), there is a noticeable decrease in coherence as sensor number increases. In the RFR method, sensors are ordered in terms of importance, with sensor 1 being most important and sensor 7 being the least important of the top 7 sensors. As a result, coherence decreases as sensor number increases. This is particularly noticeable in RFR-ODS sensor 7 (Fig. 7.15), where coherence past 1 kHz is lower compared to sensors 1-6 and compared to sensor 7 on other methodologies. The RFR-FRF and RFR-nODS sensor selection both also exhibit this behavior, although to a lesser extent than the RFR-ODS method.

7.4 Feature Extraction

Before extracting the natural frequencies, the stabilization diagrams for each of the sensor configurations were plotted using the *modalsd* MATLAB function. This function was run on the FRF function generated previously, but restricted to between 10 and 2000 Hz, as the frequencies of interest occur between those two values. The function was also generated with the default LSCE algorithm. Using the LSRF algorithm does improve frequency resolution, especially at the low end of the frequency range, but at the cost of greatly increased computation time. The LSCE based stabilization diagram provided a good baseline for determining the model order to use for the *modalfit* function, even if the *modalfit* function was used with the LSRF algorithm. The stabilization diagrams are presented below. Each stabilization diagram presents the averaged response function from all the sensor data, and the frequencies extracted from the averaged FRF. By following the model number across the left Y axis, the model order needed to extract the desired modes can be determined. Higher modes yield more frequencies, but can also identify frequencies that are purely computation, i.e., artifacts from curve fitting.

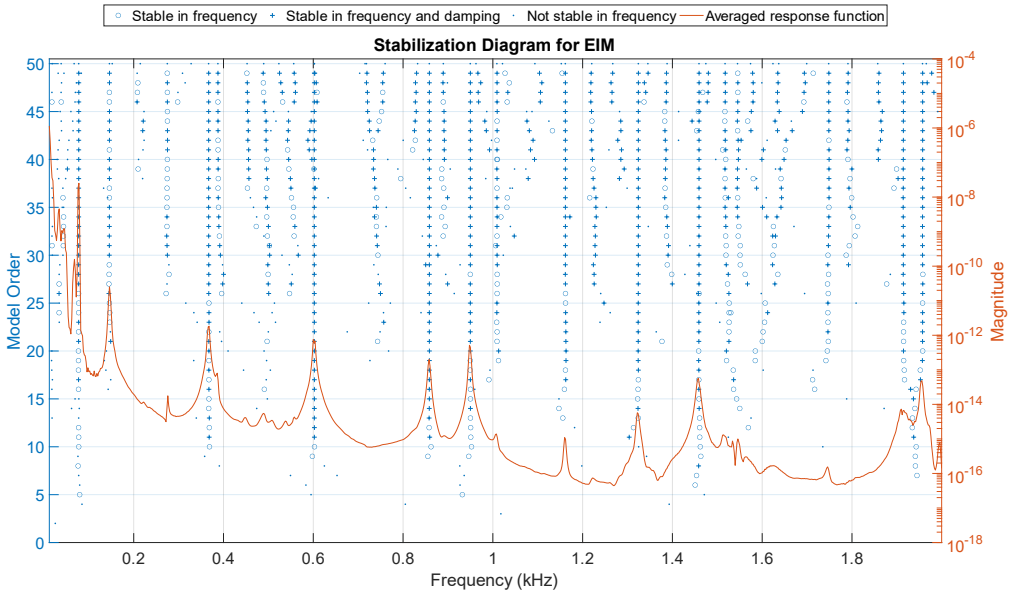

Fig. 7.22 EIM stabilization diagram

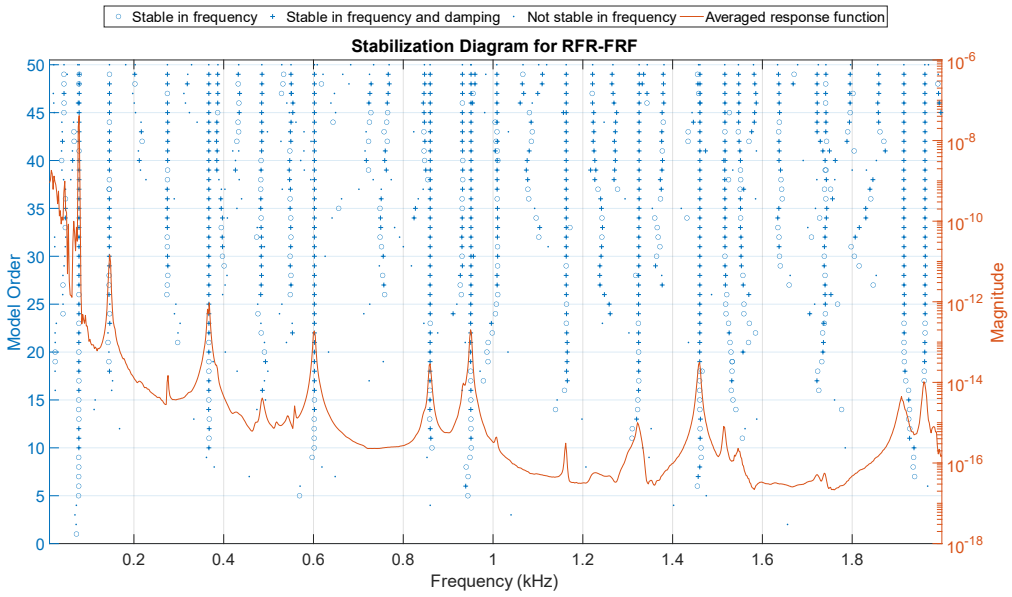

Fig. 7.23 RFR-FRF stabilization diagram

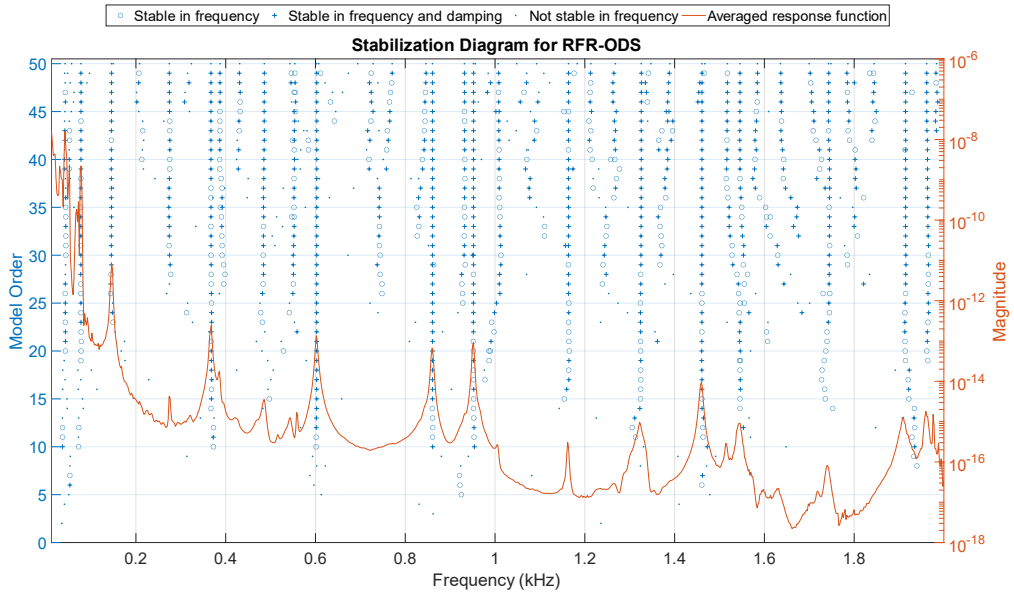

Fig. 7.24 RFR-ODS stabilization diagram

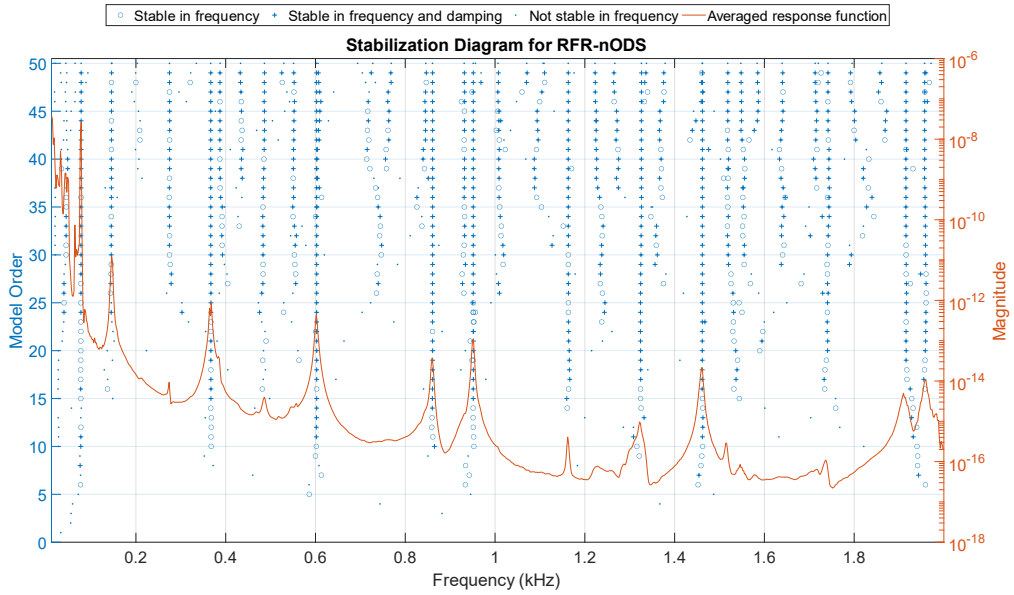

Fig. 7.25 RFR-nODS stabilization diagram

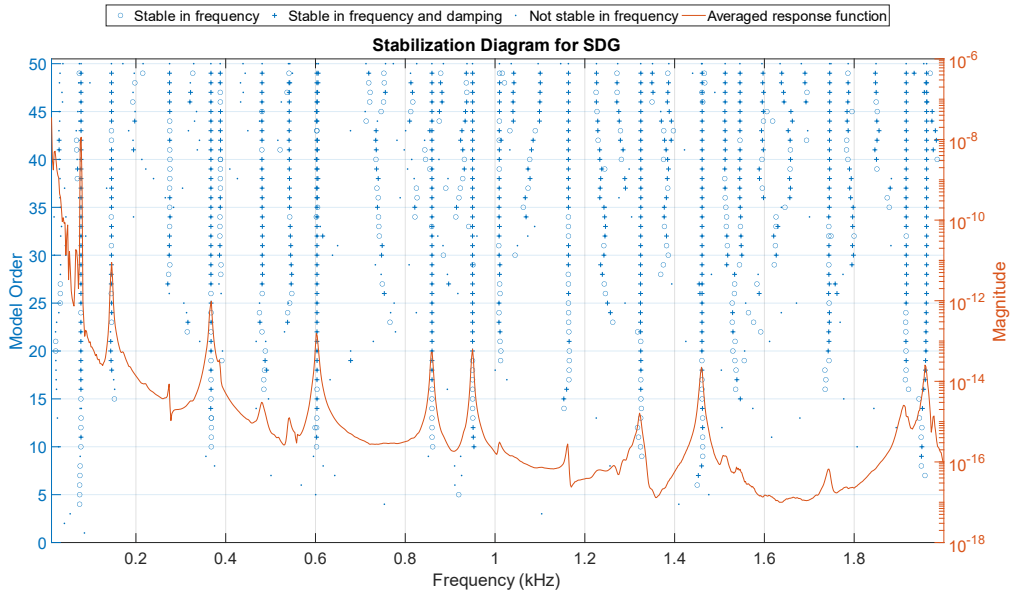

Fig. 7.26 SDG stabilization diagram

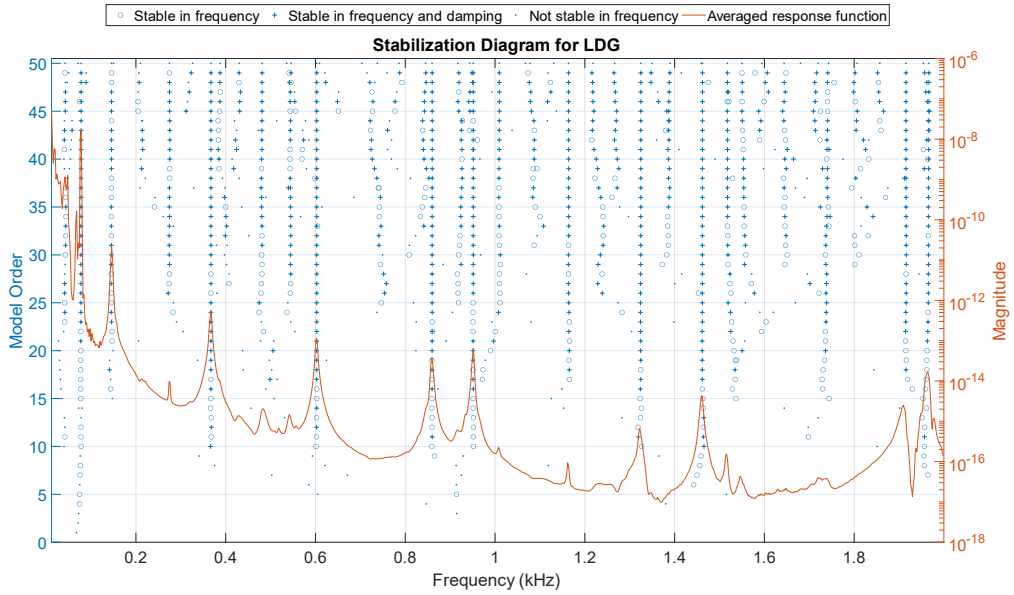

Fig. 7.27 LDG stabilization diagram

7.5 Results

Natural frequencies were extracted from the FRF data using the *modalfit* MATLAB function. In this case, the function was limited to only return frequencies between 10 and 2000 Hz, as this frequency range is the mostly cleanly represented in the FRFs as well as covering the primary area of interest. See Table 7.2 and Table 7.3 for frequencies.

Table 7.2 Natural frequencies (EIM, RFR-FRF, RFR-ODS)

EIM	Closest FEA Freq	% Error	RFR-FRF	Closest FEA Freq	% Error	RFR-ODS	Closest FEA Freq	% Error
13.9			12.7			14.0		
19.7			16.1			17.2		
20.4	24.1	15.1	22.4	24.1	7.0	21.9	24.1	8.9
30.6			27.9			29.7		
33.3			33.2			38.5		
33.7			33.9			42.0		
35.6			39.0			46.3		
38.2			41.7			49.1		
41.5			44.8			52.4		
47.2			49.2			65.1		
67.8			64.2			75.9		
76.1			68.6			80.5	78.4	2.6
80.0	78.4	2.1	76.2			367.8	363.2	1.3
366.5	363.2	0.9	80.2	78.4	2.3	385.5		
386.5			366.2	363.2	0.8	540.5	551.9	2.1
601.7	591.1	1.8	384.7			601.7	591.1	1.8
856.1	843.5	1.5	601.7	591.1	1.8	859.7	843.5	1.9
858.0			859.0	843.5	1.8	927.1	920.8	0.7
948.8	920.8	3.0	930.2	920.8	1.0	947.4	920.8	2.9
950.2			950.3			951.3		
1324.5	1343.8	1.4	1168.1	1139.4	2.5	1325.4	1343.8	1.4
1458.1	1433.1	1.7	1327.1	1343.8	1.2	1329.9	1343.8	1.0
1462.0			1459.6	1433.1	1.9	1427.1	1433.1	0.4
1515.6	1518.1	0.2	1515.9	1518.1	0.1	1464.5	1433.1	2.2
1540.4	1518.1	1.5	1547.2			1515.3	1518.1	0.2
1542.3	1518.1	1.6	1693.3			1539.9	1518.1	1.4
1910.9	1920.3	0.5	1910.6	1920.3	0.5	1554.6	1518.1	2.4
1912.9	1920.3	0.4	1914.6	1920.3	0.3	1736.6		
1935.9	1920.3	0.8	1950.3	1973.6	1.2	1892.8	1881.4	0.6
1948.7	1973.6	1.3	1960.8	1973.6	0.6	1911.0	1920.3	0.5

Table 7.3 Natural Frequencies (RFR-nODS, LDG, SDG

RFR-nODS	Closest FEA Freq	% Error	LDG	Closest FEA Freq	% Error	SDG	Closest FEA Freq	% Error
15.3			14.1			10.9		
17.8			17.6			16.2		
22.4			20.9			18.3		
25.2	24.1	4.6	24.2	24.1	0.6	21.6	24.1	10.1
32.0			30.1			28.9		
36.4			40.0			34.1		
42.8			44.2			39.3		
49.8			54.6			42.5		
50.5			65.0			42.7		
64.0			72.8			48.8		
68.9			75.9			52.2		
77.9	78.4	0.7	79.5	78.4	1.4	69.0		
79.5			144.6	147.4	1.9	76.1		
366.2	363.2	0.8	366.7	363.2	1.0	80.0	78.4	2.0
386.8	363.2	6.5	601.2	591.1	1.7	366.6	363.2	0.9
601.3	591.1	1.7	603.6	591.1	2.1	386.9		
604.0	591.1	2.2	858.9	843.5	1.8	602.6	591.1	1.9
860.0	843.5	2.0	859.2	843.5	1.9	800.3	843.5	5.1
929.6	920.8	1.0	935.3	920.8	1.6	858.6	843.5	1.8
949.4	920.8	3.1	948.5			948.9		
950.8	920.8	3.3	954.3			1323.6	1343.8	1.5
1162.2	1139.4	2.0	1323.2	1343.8	1.5	1331.6	1343.8	0.9
1324.0	1343.8	1.5	1458.9	1433.1	1.8	1459.9	1433.1	1.9
1459.0	1433.1	1.8	1460.7	1433.1	1.9	1461.7	1433.1	2.0
1460.4	1433.1	1.9	1516.5	1518.1	0.1	1507.4	1501.8	0.4
1515.7	1518.1	0.2	1913.1	1920.3	0.4	1509.0	1501.8	0.5
1542.8	1518.1	1.6	1914.5	1920.3	0.3	1805.2		
1911.1	1920.3	0.5	1947.4	1973.6	1.3	1914.8	1920.3	0.3
1913.8	1920.3	0.3	1950.3	1973.6	1.2	1958.9	1973.6	0.7
1921.0	1920.3	0.0	1956.4	1973.6	0.9	1961.2	1973.6	0.6

Both tables were created by extracting the natural frequencies from the FRFs, and then comparing each extracted frequency to the frequencies calculated using FEA. The low frequencies contained many extra modes compared to the FEA data, and unfortunately the FRF is not clear enough in this band to determine which are physical modes. The above two charts

have blanks where the percentage error was large and an existing predicted frequency already exists.

Table 7.4 Percent error and standard deviation from all natural frequencies

	EIM	RFR-FRF	RFR-ODS	RFR-nODS	LDG	SDG
Mean	14.51	16.72	15.14	14.91	9.88	19.53
Sdev	23.82	27.25	27.23	28.82	19.23	28.33

Examining the mean and standard deviation of the complete chart with no eliminated values, the LDG method was found to have both the smallest mean error as well as the lowest standard deviation; however, this only accounts for natural frequencies and does not consider the accuracy of mode shapes. A further study of the data generated is needed to more accurately determine the effectiveness of these techniques; however, as the sensor placement was generated from a comparatively low number of time steps, these results must be taken with caution.

8 Conclusion

Machine learning represents an appealing solution to the issue of sensor selection for modal testing. Current algorithms used for sensor selection are iterative when implemented ideally, and for large geometries and complex models, the computational time can be substantial. The ODS RFR sensor selection methodology appears to compare favorably with the existing EIM sensor selection methodology in the case of a 1D beam.

Unfortunately, the 2D plate analysis was inconclusive due to the smaller FEA dataset used. A drawback to the RFR based sensor selection techniques is that, using the current parameters of average FRF, normalized ODS, and ODS, acceleration data for every node must be extracted from a transient FEA analysis. This is very expensive computationally and does not scale well into models with large node counts. Although the RFR run is faster than the purely iterative EIM, the upfront cost cannot be ignored when comparing these techniques. If the RFR-based techniques can provide a better sensor selection, allowing fewer sensors, this may justify the cost. More experimentation is needed to prove the effectiveness of these techniques. Additionally, this work focused primarily on comparison of natural frequencies and predicted mode shapes. In future work, the Modal Assurance Criterion (MAC) should be calculated at each frequency for each method to allow for better assessment of the technique's effectiveness.

Though natural frequencies matching those of the FEA model were extracted from all of the sensor placements, the low number of points used to generate the sensor placements makes it difficult to confidently draw conclusions on the sensor placement effectiveness. Future studies should rerun the sensor selection method with more exported time steps to allow for better capturing of low frequency modes and provide a larger dataset for training. Depending on the sensor placements determined by adding more datapoints, physical verification should be conducted again. Additionally, implementing the RKE method would allow an additional point of comparison to be made for the RFR based sensor selection methodologies. Alternative selection criteria for the RFR may also be experimented with, ideally to reduce the computational cost of generating the data for the RFR.

References

[1] Ewins, D. J. *Modal Testing: Theory and Practice*. Research Studies Press Ltd., 1984.

[2] Harris, C. M. *Shock and Vibration Handbook*. McGraw-Hill Book Company, New York, 1976.

[3] Kammer, D. C., and Yao, L. "Enhancement of On-Orbit Modal Identification of Large Space Structures through Sensor Placement." *Journal of Sound and Vibration*, Vol. 171, No. 1, 1994, pp. 119–139. https://doi.org/10.1006/jsvi.1994.1107.

[4] Schulze, A., Zierath, J., Rosenow, S. E., Bockhahn, R., Rachholz, R., and Woernle, C. "Optimal Sensor Placement for Modal Testing on Wind Turbines." *Journal of Physics: Conference Series*, Vol. 753, No. 7, 2016. https://doi.org/10.1088/1742-6596/753/7/072031.

[5] Akers, J. C., Otten, K. D., Sills, J. W., and Larsen, C. E. Modern Modal Testing: A Cautionary Tale. In *Conference Proceedings of the Society for Experimental Mechanics Series*, Springer, 2020, pp. 1–8.

[6] Inman, D. J. *Engineering Vibration*. Pearson, Boston, 2020.

[7] Demirlioglu, K., Gonen, S., and Erduran, E. "On the Selection of Mode Shapes Used in Optimal Sensor Placement." *Sensors and Instrumentation, Aircraf/Aerospace and Dynamic Environments Testing, Volume 7. Conference Proceedings of the Society for Experimental Mechanics Series*, 2023, pp. 85–92. https://doi.org/10.1007/978-3-031-05415-0_8.

[8] Stephan, C. "Sensor Placement for Modal Identification." *Mechanical Systems and Signal Processing*, Vol. 27, No. 1, 2012, pp. 461–470. https://doi.org/10.1016/j.ymssp.2011.07.022.

[9] Demirlioglu, K., Gonen, S., and Erduran, E. On the Selection of Mode Shapes Used in Optimal Sensor Placement. 2023.

[10] Papadimitriou, C. "Optimal Sensor Placement Methodology for Parametric Identification of Structural Systems." *Journal of Sound and Vibration*, Vol. 278, Nos. 4–5, 2004, pp. 923–947. https://doi.org/10.1016/j.jsv.2003.10.063.

[11] Coote, J. E., Lieven, N. A. J., and Skingle, G. W. "Sensor Placement Optimisation for Modal Testing of a Helicopter Fuselage." *Conference Proceedings of the Society for Experimental Mechanics Series*, 2005.

[12] Udwadia, F. E. "Methodology for Optimum Sensor Locations for Parameter Identification in Dynamic Systems." *Journal of Engineering Mechanics*, Vol. 120, No. 2, 1994, pp. 368–390. https://doi.org/https://doi-org.libaccess.sjlibrary.org/10.1061/(ASCE)0733-9399(1994)120:2(368).

[13] Lollock, Jeffrey A. ; Cole, T. R. The Effect of Mass-Weighting on the Effective Independence of Mode Shapes. In *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference - Collection of Technical Papers*, 2005, pp. 449–456.

[14] Coppolino, R. N. “Systematic Modal Test Planning.” *The Integrated Test Analysis Process for Structural Dynamic Systems*, Vol. 1, 2020, pp. 37–64. https://doi.org/10.1007/978-3-031-79729-3_3.

[15] Coppolino, R. Aerospace Perspective for Modeling and Validation. In *Handbook of Experimental Structural Dynamics*, Springer New York, New York, NY, 2022, pp. 1–36.

[16] Coppolino, R. N. “Response DOF Selection for Mapping Experimental Normal Modes-2016 Update.” *Conference Proceedings of the Society for Experimental Mechanics Series*, Vol. 4, 2016, pp. 33–46. https://doi.org/10.1007/978-3-319-29763-7_5.

[17] Farrar, C. R., and Worden, K. *Structural Health Monitoring: A Machine Learning Perspective*. 2012.

[18] Yuan, F.-G., Zargar, S. A., Chen, Q., and Wang, S. “Machine Learning for Structural Health Monitoring: Challenges and Opportunities.” Vol. 1137903, No. April, 2020, p. 2. https://doi.org/10.1117/12.2561610.

[19] Badarinath, P. V., Chierichetti, M., and Kakhki, F. D. “A Machine Learning Approach as a Surrogate for a Finite Element Analysis: Status of Research and Application to One Dimensional Systems.” *Sensors*, Vol. 21, No. 5, 2021, pp. 1–18. https://doi.org/10.3390/s21051654.

[20] Liu, D., Tang, Z., Bao, Y., and Li, H. “Machine-Learning-Based Methods for Output-Only Structural Modal Identification.” *Structural Control and Health Monitoring*, Vol. 28, No. 12, 2021. https://doi.org/10.1002/stc.2843.

[21] Mathworks. Modalfrf. *MATLAB R2022b Documentation*. https://www.mathworks.com/help/signal/ref/modalfrf.html.

[22] Haberman, C. M. *Vibration Analysis*. Carles E. Merril Publishing Company, Columbus, 1968.

[23] Mathworks. Modalfit. *MATLAB R2022b Documentation*. https://www.mathworks.com/help/signal/ref/modalfit.html.

[24] Worley, W. J., Ed. *Experimental Techniques in Shock and Vibration*. The American Society of Mechanical Engineers, New York, 1962.

[25] Kammer, D. C. “Sensor Placement for On-Orbit Modal Identification and Correlation of Large Space Structures.” *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 2, 1991, pp. 251–259. https://doi.org/10.2514/3.20635.

[26] Choppala, S., Kelmar, T. W., Chierichetti, M., Davoudi, F., and Huang, D. Optimal Sensor Location and Stress Prediction on a Plate Using Machine Learning. 2023.

[27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Duboug, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. “Skikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

Appendix A: Effective Independence Code

```
function [sensedNodes] = SensorSelector(P,numsensSel,selType,anType,supressPlots)
% This function selects a subset of nodes (sensor locations) for a rfr
% given dataset based on the selected method.
% Inputs:
% - P: matrix containing data for sensor selection
% - numsensSel: number of sensors to be selected
% - selType: method for sensor selection (e.g. 'effindep', 'rfrODS', 'rfrnormODS',
'rfrAvgFRF')
% - anType: 'beam' or 'plate' - changes how data is processed depending on
if nargin == 4
    supressPlots='';
end
% if beam or plate is being considered.
if strcmp(selType,'effindep')
    %EIM Method
    if strcmp(anType,'beam')
    % Get only the vertical mode shapes
    PSel=P(2:3:end,1:15);
    %Add index to Phi for sensor selection
    Pdexed=[(1:size(PSel,1))',PSel];

    %output full EFI table for error checking/Manual verification
    % [Edexed,~]=efIndep(Pdexed);


    Ait=[];
    %loop through EFI runs. At each run, remove the smallest contribution to
    %the EFI matrix and then rerun.
        if strcmp(supressPlots,'unsupressed')
            figure
            hold on
        end
        while(height(Pdexed)>numsensSel-1)

            [Edexed,A]=efIndep(Pdexed); % get EFI table for current Phi Values
            Ait=[Ait,det(A)];

            if height(Edexed) >numsensSel-1
                [~,Idex]=min(Edexed(:,2));
                Pdexed(Idex,:)=[];

            end
            % Plot the sorted effective independence values
            if strcmp(supressPlots,'unsupressed')
                plot(sort(Edexed(:,2),'descend'));
            end
            sensedNodes=sort(Edexed(:,1),'descend');
        end
    % Plot the determinant of the Fisher Information Matrix (A_0) vs iteration
%     figure
%     title('Effective Independence (Sorted)')
%     xlabel('Sensor Locations')
```

```
%     ylabel('Effective Independence')
%     figure
%     plot(Ait)
%     title('Determinant of Fisher Information Matrix (A_0) vs Iteration')
%     xlabel('Iteration')
%     ylabel('Det(A_0)')
    elseif strcmp(anType, 'plate')
        % For a plate, since the Nodes are not simply linearly ordered, the
        % index is passed from the imported ANSYS table since in this case
        % the index corresponds to the node number and the physical
        % location of the sensor. Additionally, the relevant data is
        % already extracted in the ANSYS script,
        Pdexed=P;
    %output full EFI table for error checking/Manual verification
    % [Edexed,~]=efIndep(Pdexed);
    Ait=[];
    %loop through EFI runs. At each run, remove the smallest contribution to
    %the EFI matrix and then rerun.
         if strcmp(supressPlots,'unsupressed')
            figure
            hold on
        end

    while(height(Pdexed)>numsensSel-1)

        [Edexed,A]=efIndep(Pdexed); % get EFI table for current Phi Values
        Ait=[Ait,det(A)];

        if height(Edexed) >numsensSel-1
            [~,Idex]=min(Edexed(:,2));
            Pdexed(Idex,:)=[];

        end
        % Plot the sorted effective independence values
        if strcmp(supressPlots,'unsupressed')
            plot(sort(Edexed(:,2),'descend'));
        end
        [efSorted,I]=sort(Edexed(:,2),'descend');
         sensedNodes=Edexed(I,:);
        %sensedNodes=sort(Edexed(:,2),'descend');
    end
end
    % Plot the determinant of the Fisher Information Matrix (A_0) vs iteration
%     figure
%     title('Effective Independence (Sorted)')
%     xlabel('Sensor Locations')
%     ylabel('Effective Independence')
%     figure
%     plot(Ait)
%     title('Determinant of Fisher Information Matrix (A_0) vs Iteration')
%     xlabel('Iteration')
%     ylabel('Det(A_0)')


elseif strcmp(selType,'rfrODS')
```

```
    % Random Forest Regression (RFR) method based on ODS data
    ODS=importMLResults('FI_ODS.csv');
    sensedNodes=ODS{1:numsensSel,1};

elseif strcmp(selType,'rfrnormODS')
   % RFR method based on normalized ODS data
   ODSnorm=importMLResults('FI_ODSnorm.csv');
   sensedNodes=ODSnorm{1:numsensSel,1};

elseif strcmp(selType,'rfrAvgFRF')
     % RFR method based on average FRF data
    frfML=importMLResults('FI_frfTab.csv');
    sensedNodes=frfML{1:numsensSel,1};

end
% Return the selected subset of nodal sensor locations
end

function [Edexed,A] = efIndep(pdexed,n_sensors)
% This function computes the effective independence (EfI) of each degree of freedom.
% Inputs:
% pdexed-first column is an index, each subsequent column is a mode shape.
% Outputs:
% 'Edexed' contains the index and the corresponding EfI values for each mode shape.
%  'A' : Fisher information Matrix


% Extract the mode shapes
phi=pdexed(:,2:end);

EfI=diag(phi*pinv(phi'*phi)*phi');
A=phi'*phi;
% Create Edexed  by concatenating the index
% of 'pdexed' with the EfI values ('E')
Edexed=[pdexed(:,1),EfI];


end
```

Appendix B: APDL Extraction Code

```
! ANSYS MODE SHAPE EXPORT CODE
! AUTHOR: TODD KELMAR
! VERSION: 1.0
! This code must be inserted into the solution tree after all the deformations are
calculated.
! This code will export the mode shapes (total deflection) into a csv file on the
desktop.
! Modes to be exported must be specified manually
! Currently this code exports the mode shapes from one side of the model (All points
with Y=0.0061)
! The mode shape is a vector sum of the mode shape X, Y, and Z deflection
! Last updated 3/19/2023 by Todd Kelmar

! Switch to Advanced Nodal Post processor
/POST26

!Define Variables
! Select all Nodes to check verify the total number of nodes (For troubleshooting)
! NSLE,ALL
! Select all the nodes on one face of the plate
NSEL,S,LOC,Y, 0.0061

! Get the number of selected nodes and store in NUM_NODES
*GET, NUM_NODES, NODE, 0, COUNT,
! Get lowest node number in the selected set and store it in currn
*GET, currn, NODE, 0, NUM, MIN,
! Set file name for easier reference
file1 ='YNodes_mode_shapes'
! Create an empty array Node_List of dimension NUM_NODES x 4 for storing node number,
X, Y, and Z coord
*DIM, Node_List, ARRAY, NUM_NODES, 4
! Create an empty array MODES of dimension 13 x 1
*DIM, MODES, ARRAY, 13
! Create empty MODE_SHAPES array to store the mode shapes NUM_NODES x 13
*DIM, MODE_SHAPES, ARRAY, NUM_NODES, 13
! Create Natural Freq array of dim 13 x 1
*DIM, NATURAL_FREQUENCIES, ARRAY, 13
! Define the modes of interest (manually ID as transverse bending modes from the
Total Deformation plots)
MODES(1) = 1
MODES(2) = 2
MODES(3) = 6
MODES(4) = 9
MODES(5) = 10
```

```
MODES(6) = 15
MODES(7) = 16
MODES(8) = 18
MODES(9) = 21
MODES(10) = 24
MODES(11) = 25
MODES(12) = 30
MODES(13) = 44

! Select all Nodes to check verify the total number of nodes (For troubleshooting)
! NSLE,ALL
! Select all the nodes on one face of the plate
NSEL,S,LOC,Y, 0.0061

! Get the number of selected nodes and store in NUM_NODES
*GET, NUM_NODES, NODE, 0, COUNT,
! Get lowest node number in the selected set and store it in currn
*GET, currn, NODE, 0, NUM, MIN,

! Loop through each frequency of interest
*DO, i, 1, 13, 1
! Get the natural frequency of the current mode, and store in NATURAL FREQUENCIES
*GET, NATURAL_FREQUENCIES(i), MODE, MODES(i), FREQ,
*ENDDO
! Enter the basic postprocessor
/POST1
! Select lowest node in the subset as the active node
NODE = currn
! LOOP through all nodes in the subset, for each node store the node ID, NODE X< Y< Z
Coord in Node_List
*DO, k,1,NUM_NODES,1
ncx = nx(currn)
Node_List(k,1) = currn
Node_List(k,2) = nx(currn)
Node_List(k,3) = ny(currn)
Node_List(k,4) = nx(currn)
!Loop through all 13 frequencies and extract the modal displacement (vector sum) and
store in mode shapes for the current node
*DO, l,1,13,1
SET,1,MODES(l),
 *GET, MODE_SHAPES(k,l), NODE, currn,U,SUM
*ENDDO
!Increment to next node
*GET, currn, NODE, currn, nxth,
!Select next node
```

```
NODE = currn
*ENDDO

! Write data to CSV file

FILE, file,
/CWD,'C:\Users\Todd\Desktop'
*CFOPEN, file1, csv
*VWRITE, 'Node', 'NODE X','NODE Y','NODE Z', 'Mode 1', 'Mode 2', 'Mode 6', 'Mode 9',
'Mode 10', 'Mode 15', 'Mode 16', 'Mode 18', 'Mode 21', 'Mode 24', 'Mode 25', 'Mode
30', 'Mode 44'
%C, %C,%C,%C,%C, %C, %C, %C, %C, %C, %C, %C, %C, %C, %C, %C, %C
*VWRITE,'Frequency','','','', NATURAL_FREQUENCIES(1), NATURAL_FREQUENCIES(2),
NATURAL_FREQUENCIES(3), NATURAL_FREQUENCIES(4), NATURAL_FREQUENCIES(5),
NATURAL_FREQUENCIES(6), NATURAL_FREQUENCIES(7), NATURAL_FREQUENCIES(8),
NATURAL_FREQUENCIES(9), NATURAL_FREQUENCIES(10), NATURAL_FREQUENCIES(11),
NATURAL_FREQUENCIES(12), NATURAL_FREQUENCIES(13)
%C, %C, %C, %C, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G
*CFCLOS

*CFOPEN, file1,csv, ,append
*VWRITE, Node_List(1,1),Node_List(1,2),Node_List(1,3),Node_List(1,4),
MODE_SHAPES(1,1), MODE_SHAPES(1,2), MODE_SHAPES(1,3), MODE_SHAPES(1,4),
MODE_SHAPES(1,5), MODE_SHAPES(1,6), MODE_SHAPES(1,7), MODE_SHAPES(1,8),
MODE_SHAPES(1,9), MODE_SHAPES(1,10), MODE_SHAPES(1,11), MODE_SHAPES(1,12),
MODE_SHAPES(1,13)
%G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G, %G
*CFCLOS


/EXIT
```

Appendix C: RFR Sensor Selection

```
#Random Forest Regressor for modal sensor selection
#By Todd Kelmar
import time

from sklearn import metrics
start_time=time.time()
import pandas as pd
import numpy as np
import joblib as jl
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
from sklearn.metrics import r2_score
#import seaborn as sns
import matplotlib.pyplot as plt

import os
from scipy.stats import pearsonr
from sklearn.decomposition import PCA
from sklearn.model_selection import cross_val_score,cross_val_predict,KFold
from sklearn.ensemble import RandomForestRegressor

#set up plot font sizes
SMALL_SIZE = 10
MEDIUM_SIZE = 20
BIGGER_SIZE = 30

plt.rc('font', size=MEDIUM_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=BIGGER_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=MEDIUM_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=MEDIUM_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=MEDIUM_SIZE)    # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure title
print('Changing Directory')
#Change directory
os.chdir("X:/Onedrive/.SJSU/AE295 - Masters Project/Code/beam_Matlab_files/Current
Code")

stdf=pd.DataFrame(columns=['Dataset','R2','MSE'])
```

```
#loop through the folder to process each file individually
for filename in os.listdir(r"X:/Onedrive/.SJSU/AE295 - Masters
Project/Code/beam_Matlab_files/Current Code"):
    if filename.endswith(".csv"):
        print('Importing data...')
        rf=pd.read_csv(filename)
        print("loading %s"%(filename))

        set_name=filename[:-4]

        #set_name=['New_Plate_indiv']
        optVal= rf['OPT']
        rf.drop(columns=['OPT', 'Freq'],inplace = True)


        print("rfr running for %s"%(set_name))
        # applying random forest for getting feature importance plots
        model_sens = RandomForestRegressor(n_estimators = 100, n_jobs=16)

        model_sens.fit(rf,optVal)


        print("plotting %s"%(set_name))
        sorted_index_optVal = model_sens.feature_importances_.argsort()
        plt.figure(figsize=(60,30))
        plt.barh(rf.columns[sorted_index_optVal],
model_sens.feature_importances_[sorted_index_optVal])
        plt.xlabel('Feature Importance')
        featDF=pd.DataFrame(rf.columns[sorted_index_optVal])
        featDF.insert(1,'Importance',model_sens.feature_importances_[sorted_index_opt
Val].tolist())
        featDF.to_csv("./impCols/FI_%s.csv"%set_name)
        plt.title(filename)
        print("Saving figure...")
        plt.savefig('./graphs/%s.png'%(set_name))
       # plt.show()
        rf_imp_cols=rf.columns[sorted_index_optVal]



        spred=model_sens.predict(rf)
```

```
        mse = metrics.mean_squared_error(optVal,spred)
        score=model_sens.score(rf,optVal)
        stdf.loc[len(stdf.index)]=[set_name,score,mse]

        new_rf_df = rf[rf_imp_cols]
        new_rf_df.shape
        print("Writing important columns...")
        new_rf_df.to_csv("./outputs/imp_cols_%s.csv"%set_name)
        print("Writing Random Forest Model....")
        jl.dump(rf, "./models/%s.joblib"%(set_name))
        time.sleep(5)
    else:
        continue
stdf.to_csv("./errors/errors_%s.csv"%set_name)
print((time.time()-start_time)/60)
```

Appendix D: Transient Analysis APDL Export Script

```
! ANSYS Transient Analysis Acceleration Export
! Author: Todd Kelmar
! Version: 1.2
! Last modified: 4/25/23
! Changlog:
! Reodered file write operations for speed
! Modified to run in batch mode from APDL terminal, must be placed in same working
directory as ANSYS results file.
! This script Exports only Y acceleration data, Node number, and nodal coordinates
for nodes on the Y=6.1mm plane.
! Get all nodes on the Y =6.1mm plane
RESUME
NSEL,S,LOC,Y, 0.0061
*GET, NUM_NODES, NODE, 0, COUNT,
*GET, NUM_STEPS, ACTIVE, 0, SOLU, NCMSS
!Insert Signal Filename here
filename = 'CHIRP_TEST'
!FILE, filename
*CFOPEN, filename, csv
!Set up file and write headings
*VWRITE, 'T', 'NODE', 'X','Y', 'Z', 'AY'
%C, %C, %C, %C, %C, %C
*CFCLOS
*CFOPEN, filename, csv, , append
! For each time step, loop through the selected nodes and write node number, X, Y, Z
coordinate and Y acceleration to CSV file
*DO, i, 1, NUM_STEPS, 1

/PREP7
NSEL,S,LOC,Y, 0.0061
*GET, currn, NODE, 0, NUM, MIN
NODE = currn

*DO, j, 1, NUM_NODES, 1

NODE_NUM = currn
NODE_X = nx(currn)
NODE_Y = ny(currn)
NODE_Z = nz(currn)
/POST1
SET, 1, i
*GET, NODE_TIME, ACTIVE, 0, SET, TIME
*GET, NODE_AY, NODE, currn, A, Y
! Write Data to CSV File
```

```
*VWRITE, NODE_TIME, NODE_NUM, NODE_X, NODE_Y, NODE_Z, NODE_AY
%e, %i, %e, %e, %e, %e

/PREP7
!Increment to next node
*GET, currn, NODE, currn, nxth,
!Select next node
NODE = currn

*ENDDO

*ENDDO
! Close the CSV File
*CFCLOS

FINISH
```

Appendix E: 2D Plate Natural Frequency Extraction Code

```
%The following script imports data from ANSYS, calculates the EIM method, formats the
plate data for export to python ML script, and then reimports the data to calculate
the FRF and fn for various methods.

ansysExport='YNodes_mode_shapes.dat';
sensorCount=7;
[Ptab,ftab]=importANSYSModeShapes(ansysExport);

%extract Matrix of Mode Shapes
P=table2array(Ptab(:,[1,5:end]));
tic
[sensedNodes] = SensorSelector(P,sensorCount,'effindep','plate');
toc
sensedidex=ismember(Ptab.Node,sensedNodes(:,1));

%% Import Data from FEA (Uncomment if using new FEA Data)
% FEA_DATA_Reformat
load formatted_Plate_Ac
load chirp_signal.mat
%% Plot All Possible Nodes on Plate
figure
hold on
scatter(Ptab{:,"NODEX"},Ptab{:,'NODEZ'},'.');
scatter(Ptab{sensedidex,"NODEX"},Ptab{sensedidex,'NODEZ'},'filled');
legend('FEA Nodes', 'EFI Sensed Nodes');
xlabel('Width (m)')
ylabel('Height (m)');
axis equal
%% Process Plate Data for ML

sampleRate=20e3;
winSize=100;
overlap=50;

sensDat=outputNDB{:,2:end};
input=chirp_signal;

[fullfrf,f,~]=modalfrf(input,sensDat,sampleRate,hamming(winSize),overlap);
modalfrf(input,sensDat,sampleRate,hamming(winSize),overlap);
fullfrf=abs(fullfrf);
% Combine the frequency and FRF data into a matrix for input into the
% python script
MLinput=[f,fullfrf];
% Calculate the Operational Deflection Shape (ODS) and ODS normalization
ODS=zeros([size(f,1),1]);
ODSnorm=zeros([size(f,1),1]);
for i=1:size(fullfrf,1)
ODS(i)=fullfrf(i,:)*fullfrf(1,:)';
ODSnorm(i)=ODS(i)./norm(fullfrf(1,:));
end
% Calculate the average FRF for all nodes
frfAvg=sum(fullfrf,2)./nodeCount;
```

```
% Create headers for the resulting tables

nodeLab=  nodes; %add index to keep track of the node number
header=["Freq",nodeLab'];
% Convert the data to tables and add variable names to the columns
% Create a seperate table for each of the different types of analysis
MLTabODS=array2table([MLinput,ODS], 'VariableNames',[header, "OPT"]);
MLTabODSnorm=array2table([MLinput,ODSnorm],'VariableNames',[header, "OPT"]);
MLTabAvFrf=array2table([MLinput,frfAvg],'VariableNames',[header, "OPT"]);
% Write the tables to separate .csv files
writetable(MLTabAvFrf,'frfTabPlate.csv')
writetable(MLTabODSnorm, 'ODSnormPlate.csv')
writetable(MLTabODS, 'ODSPlate.csv')



%%

numsensSel = 7;

    ODS=importMLResults('FI_ODSPlate.csv');
    sensedNodesODS=ODS{1:numsensSel,1};
odsNodes=getcord(sensedNodesODS)
figure
hold on
scatter(Ptab{:,"NODEX"},Ptab{:,'NODEZ'},'.');
scatter(odsNodes(:,2),odsNodes(:,4),'filled');
legend('FEA Nodes', 'ODS Sensed Nodes');
xlabel('Width (m)')
ylabel('Height (m)');
axis equal
title('ODS Selected Nodes')

   % RFR method based on normalized ODS data
   ODSnorm=importMLResults('FI_ODSnormPlate.csv');
   sensedNodesODSnorm=ODSnorm{1:numsensSel,1};
   odsnormNodes=getcord(sensedNodesODSnorm);

figure
hold on
scatter(Ptab{:,"NODEX"},Ptab{:,'NODEZ'},'.');
scatter(odsnormNodes(:,2),odsnormNodes(:,4),'filled');
legend('FEA Nodes', 'ODS norm. Sensed Nodes');
xlabel('Width (m)')
ylabel('Height (m)');
axis equal
title('Normalized ODS Selected Nodes')
    frfML=importMLResults('FI_frfTabPlate.csv');
    sensedNodesfrf=frfML{1:numsensSel,1};
frfNodes=getcord(sensedNodesfrf)

figure
hold on
scatter(Ptab{:,"NODEX"},Ptab{:,'NODEZ'},'.');
scatter(frfNodes(:,2),frfNodes(:,4),'filled');
```

```
legend('FEA Nodes', 'Avg. FRF Sensed Nodes');
xlabel('Width (m)')
ylabel('Height (m)');
axis equal
title('Avg. FRF Selected Nodes')


%% Plate Mode Extract
% The section manually extracts the nodes used in lab experimentation for
% consistency. It can be rewritten to extract acc data automatically if
% needed.
EIMsensdat=[outputNDB.Node16990_AccY,outputNDB.Node17037_AccY,
outputNDB.Node4540_AccY, outputNDB.Node12798_AccY, outputNDB.Node17029_AccY,
outputNDB.Node4648_AccY,outputNDB.Node4500_AccY];
ODSnormsensdat=[outputNDB.Node14426_AccY,outputNDB.Node13080_AccY,outputNDB.Node14155
_AccY,outputNDB.Node15116_AccY,outputNDB.Node15468_AccY,outputNDB.Node14430_AccY,outp
utNDB.Node14090_AccY];
ODSsensdat=[outputNDB.Node4384_AccY,outputNDB.Node4163_AccY,outputNDB.Node13412_AccY,
outputNDB.Node16034_AccY,outputNDB.Node16766_AccY,outputNDB.Node17072_AccY,outputNDB.
Node13262_AccY];
FRFsensdat=[outputNDB.Node15136_AccY,outputNDB.Node15615_AccY,outputNDB.Node3548_AccY
,outputNDB.Node15896_AccY,outputNDB.Node13877_AccY,outputNDB.Node3358_AccY,outputNDB.
Node4055_AccY];
time=outputNDB.Time;
signal=chirp_signal;
sampleRate=20e3;
winSize=100;
overlap=50;

% EIM
figure
[EIMfrf,EIMf,EIMcoh]=modalfrf(signal,EIMsensdat,fs,hamming(winSize),overlap);
modalfrf(signal,EIMsensdat,fs,hamming(winSize),overlap);
[EIMfn]=modalfit(EIMfrf,EIMf,sampleRate,7,FitMethod="lsrf" );
% ODS
figure
[ODSfrf,ODSf,ODScoh]=modalfrf(signal,ODSsensdat,fs,hamming(winSize),overlap);
modalfrf(signal,ODSsensdat,fs,hamming(winSize),overlap);
[ODSfn]=modalfit(ODSfrf,ODSf,sampleRate,7,FitMethod="lsrf" );
% ODSNORM
figure
[normODSfrf,normODSf,normODScoh]=modalfrf(signal,ODSnormsensdat,fs,hamming(winSize),o
verlap);
modalfrf(signal,ODSnormsensdat,fs,hamming(winSize),overlap);
[normODS]=modalfit(normODSfrf,normODSf,sampleRate,7,FitMethod="lsrf" );
% AVG FRF
figure
[FRFfrf,FRFf,FRFcoh]=modalfrf(signal,FRFsensdat,fs,hamming(winSize),overlap);
modalfrf(signal,FRFsensdat,fs,hamming(winSize),overlap);
[FRFfn]=modalfit(FRFfrf,FRFf,sampleRate,7,FitMethod="lsrf" );


%%
function[selNodeCords]= getcord(selNodes)
% Helper function to associate node names with their respective
```

```
% coordinates. Uses prebuilt Coordinate Table
load CoordTab
selNodeCords=zeros([length(selNodes),4])
for i =1:length(selNodes)
match=ismember(CoordTab(:,1),selNodes(i));
selNodeCords(i,:)=CoordTab(match,:)
end
end
```

Appendix F: Modal Extraction MATLAB Code

```
% %Extract Data and Format for ML from main_beam output table

%  clear all


%load ref data from FEA
%  load T.mat
%  load F.mat
%  load w.mat
markers=["o","+","*","X","square"];
sensedNodes=[Edexed,ODS,ODSnorm,frfML];
figure
hold on
% Loop through different sensed nodes and plot natural frequencies.
for i=1:size(sensedNodes,2)
sensLoc=unique(T.x);
nNodes=size(sensLoc,1);
 sens_pos=sensLoc(sensedNodes(:,i));
% sens_pos=sensLoc(linspace(1,nNodes,nNodes));
for j=1:length(sens_pos)
tempTab=T(ismember(T.x,sens_pos(j)),:);
    sensDat(:,j)=tempTab.acc;
    clear tempTab
end

sigLength=size(sensDat,1);


%
%  figure
%
% plot(F(122,:))
whz=w/2/pi;

%%
 t=unique(T.time);

 %% Windowing for Random
winSize=600;
 overlap=190;
%% Windowing for Chirp
% winSize=1000;
%  overlap=600;

sampleRate=1/mean(diff(unique(T.time)));

[Pxx, freqs] = pwelch(sensDat, hamming(winSize), overlap, [], sampleRate);
% figure
% plot(Pxx,freqs)
%     figure
% modalfrf(F(end-1,:)',sensDat, sampleRate,hamming(winSize),overlap)
```

```
 [frf, f,coh]=modalfrf(F(end-1,:)',sensDat, sampleRate,hamming(winSize),overlap);
%figure
%sdm= modalsd(frf,f,sampleRate);
%modalsd(frf,f,sampleRate)
%figure
% xline(w)
%[fnlsce, drlsce, ms,ofrf]=modalfit(frf,f,sampleRate,10 ...
%   ,FitMethod="lsce");
%modalfit(frf,f,sampleRate,24,FitMethod="lsrf")
%[fnpp, drpp, mspp,ofrfpp]=modalfit(frf,f,sampleRate,10 ...
%    ,FitMethod="pp");

[fnlsrf]=modalfit(frf,f,sampleRate,10 ...
    ,FitMethod="lsrf" );
% [fnlsce]=modalfit(frf,f,sampleRate,7 ...
%     ,FitMethod="lsce" );

scatter(1:length(fnlsrf),fnlsrf,100,markers(i))
end
plot(whz(1:10), "-square")
xlabel('Natural Frequency Number')
ylabel('Freqency (Hz)')
% title("Natural Frequencies Window Size"+ winSize+" Overlap:"+overlap)
title('Natural Frequencies')
legend('f EIM','f ODS', 'f normODS','f FRF', 'f FEA')



%% Plot Ideal FRF From Matricies
%  H=idealFRF(w,M,C,K);

% Plot ideal FRF

% main_modefit
%%
Pvert=P(2:3:end,1:7);



%% Sensor Selctor
sensorNum=8;
tic
Edexed=SensorSelector(P,sensorNum,'effindep','beam');
toc
ODS=SensorSelector(P,sensorNum,'rfrODS','beam');
ODSnorm=SensorSelector(P,sensorNum,'rfrnormODS','beam');
frfML=SensorSelector(P,sensorNum,'rfrAvgFRF','beam');


%% Sensor Selector Plotting
%Plot Sensor Selection output
   figure
hold on
numcharts=7;
```

```
for k=1:numcharts

subplot(numcharts,1,k)
hold on
plot(Pvert(:,k))
title("Mode  \omega = " +wHz(k)+" Hz")
xlabel("Node")
ylabel("Mode Shape")
xlim([1 size(Pvert,1)])
xline(Edexed(:,1), ':b','LineWidth',3.5)
xline(ODS(:,1),'--r','LineWidth',2)
xline(ODSnorm(:,1),'g','LineWidth',2)
xline(frfML(:,1),'-.m', 'LineWidth',2)
end
figure
hold on
numcharts=7;
% for m=1:numcharts
% subplot(numcharts,1,m)
% hold on
% plot(Pvert(:,m))
% title("Mode  f = " +wHz(m)+" Hz")
% xlabel("Node")
% ylabel("Mode Shape")
% xlim([1 size(Pvert,1)])
% plot(Edexed(:,1),Pvert(Edexed(:,1),m), '-s','LineWidth',2)
% plot(sort(ODS(:,1)),Pvert(sort(ODSnorm(:,1)),m),'--r','LineWidth',2)
% plot(sort(ODSnorm(:,1)),Pvert(sort(ODSnorm(:,1)),m),'-x','LineWidth',2)
% plot(sort(frfML(:,1)),Pvert(sort(frfML(:,1)),m),'-o', 'LineWidth',2)
% legend('FEA Mode Shape', 'EFI Sensor Mode Shape', 'ODS Sensor Mode Shape',
'Normalized ODS Sensor Mode Shape', 'FRF Sensor Mode Shape',
'Location','eastoutside')
% end
ODSnormplot=sort(ODSnorm(:,1));
for m=1:7
subplot(4,7,m)
hold on
plot(Pvert(:,m))
title("Mode  f = " +wHz(m)+" Hz")
xlabel("Node")
ylabel("Mode Shape")
xlim([1 size(Pvert,1)])
scatter(Edexed(:,1),Pvert(Edexed(:,1),m))

subplot(4,7,m+7)
hold on
plot(Pvert(:,m))
title("Mode  f = " +wHz(m)+" Hz")
xlabel("Node")
ylabel("Mode Shape")
xlim([1 size(Pvert,1)])
scatter(sort(ODS(:,1)),Pvert(sort(ODS(:,1)),m))

subplot(4,7,m+14)
```

```
hold on
plot(Pvert(:,m))
title("Mode  f = " +wHz(m)+" Hz")
xlabel("Node")
ylabel("Mode Shape")
xlim([1 size(Pvert,1)])
scatter(sort(ODSnorm(:,1)),Pvert(sort(ODSnorm(:,1)),m))

subplot(4,7,m+21)
hold on
plot(Pvert(:,m))
title("Mode  f = " +wHz(m)+" Hz")
xlabel("Node")
ylabel("Mode Shape")
xlim([1 size(Pvert,1)])
scatter(sort(frfML(:,1)),Pvert(sort(frfML(:,1)),m))


% legend('FEA Mode Shape', 'EFI Sensor Mode Shape', 'ODS Sensor Mode Shape', ...
%     'Normalized ODS Sensor Mode Shape', 'FRF Sensor Mode Shape',
'Location','eastoutside')
end
```

Appendix G: Ideal FRF MATLAB Code

```
function [H] = idealFRF(w,M,C,K)
% Calculates the frequency response function (FRF) for a system
% with natural frequencies w, mass matrix M, damping matrix C,
% and stiffness matrix K.

% Set up frequency vector
% f = logspace(0,4, 10000); % 1000 logarithmically spaced frequencies from 1 to
10,000 H
f = linspace(1,10000,100000);
w = 2 * pi * f;
% Initializing an ideal force vector with zeros
% except for the second to last element which is set to 1
idealF=zeros([length(M),1]);
idealF(end-1)=1;
% Calculate num degrees of freedom
n_dof = size(M, 1);
%Initialize FRF Matrix H
H = zeros(n_dof, length(w));

%For each frequency: Calculate the A matrix and then the FRF matrix
for i = 1:length(w)
    A=(K-w(i)^2*M+1i*w(i)*C);
     H(:,i)=A\idealF;
end

% Plot FRF
figure
hold on
colors = lines(n_dof);
% Loop through each dof and plot it's FRF
%for i = 1:n_dof

i=n_dof;
plot(f, db(abs(H(i, :))), 'LineWidth', 1, 'Color', colors(i, :));

%end


xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('Frequency Response Function (FRF)')
legend(arrayfun(@(i) sprintf('DOF %d', i), 1:n_dof, 'UniformOutput', false),
'Location','eastoutside')
grid on
end
```

Appendix H: MATLAB Import/Export Functions

```
% This function imports the data from the RFR Python output
% and removes the first column since it is just an index
function [fileTab] = importMLResults(filename)

% Read the data from the file and assign it to fileTab
fileTab=readtable(filename, 'ReadVariableNames',true);

% Remove the first column from the table since it is assumed to be a row index
fileTab(:,1)=[];

% Sort the rows by feature importance.
fileTab=sortrows(fileTab,2,'descend');

% Return the sorted table to the calling function
end

function [modeShape,modeFreq]=importANSYSModeShapes(filename)
% import data from CSV generated by ANSYS APDL script.
% Reads table and outputs two tables, one of nodes with their corresponding
% numbers, locations, and mode shapes for the exported frequencies.
% The second table ModeFreq contains the natural frequencies for the
% exported mode shapes and is stored seperately for ease of retreival.

%Suppress warning about table header names. Does not impact data
warning('off','MATLAB:table:ModifiedAndSavedVarnames')

% Read CSV file
imptab=readtable(filename,'ReadVariableNames',true);

%Split table by removing frequencies to a seperate table.
modeFreq=imptab(1,5:end);
modeShape=removerows(imptab,1);
end



function FEA_DATA_Reformat()
load CHIRP.MAT

inputNDB=CHIRP;
nodes=unique(CHIRP.NODE);
nodeCount=length(nodes);
tNodeStep=unique(CHIRP.T);
tstepcount=length(tNodeStep);
%
% outputNDB=zeros(tNodeStep,nodeCount+1);

outputNDB=array2table(tNodeStep, 'VariableNames', {'Time'});



for node = nodes'
    nodeData=inputNDB(inputNDB{:,2} == node, :);
```

```
    nodeAcc=nodeData{:,6};
     if length(nodeAcc) ~= tstepcount
        error('Data is not uniformly sampled or missing for some nodes');
     end

     outputNDB{:,end+1}=nodeAcc;
     outputNDB.Properties.VariableNames{end} = sprintf('Node%d_AccY',node);


end

save formatted_Plate_Ac outputNDB

%%
CoordTab=CHIRP{1:nodeCount,2:5};
end
```

Appendix I: MATLAB Plate Experimental Data Processing

```
clear
clc
load TestData.mat

%% Import and format Data for processing.
[tfrf,frfsig]=expDat(avgFRFdata);
[tlgrid,lgridsig]=expDat(largeGriddata);
[tsgrid,sgridsig]=expDat(smallGriddata);
[tods,odssig]=expDat(ODSdata);
[tnods,nodssig]=expDat(normODSdata);
[teim,eimsig]=expDat(EIMdata);
fs=8533;

%% PreProcess
% Apply lowpass filter to signals
frfsigp=preprocessPlate(frfsig,fs);
lgridsigp=preprocessPlate(lgridsig,fs);
sgridsigp=preprocessPlate(sgridsig,fs);
odssigp=preprocessPlate(odssig,fs);
nodssigp=preprocessPlate(nodssig,fs);
eimsigp=preprocessPlate(eimsig,fs);
%%
fs=8533; %sample rate in hz
winsize=round((fs/20)*10,0);

%%
%Generate FRFs
[fn(:,2),~,~,~,~]=frfPlate(frfsigp,fs,winsize,'RFR-FRF');
[fn(:,5),~,~,~,~]=frfPlate(lgridsigp,fs,winsize,'LDG');
[fn(:,6),~,~,~,~]=frfPlate(sgridsigp,fs,winsize,'SDG');
[fn(:,3),~,~,~,~]=frfPlate(odssigp,fs,winsize,'RFR-ODS');
[fn(:,4),~,~,~,~]=frfPlate(nodssigp,fs,winsize,'RFR-nODS');
[fn(:,1),~,~,~,~]=frfPlate(eimsigp,fs,winsize,'EIM');



function [fn,dr,ms,frf, f] = frfPlate(sig,fs,winsize, selectionname)
%Helper function to computer frf function using a hann window, 90% overlap.

[frf, f] = modalfrf(sig(:,1),sig(:,2:end), fs, hann(winsize),round(winsize*.9,0));
figure
modalfrf(sig(:,1),sig(:,2:end), fs, hann(winsize),round(winsize*.9,0));
% sgtitle(['FRF for sensor 1, 2, 3, 4 of ' selectionname])
set(gcf,'position',[0,0,1920,1080])
fontsize(gcf,scale=1.5)
saveas(gcf, [selectionname '1.fig'])
saveas(gcf,[selectionname '1.svg'])
figure
modalfrf(sig(:,1),sig(:,6:end), fs, hann(winsize),round(winsize*.9,0));
% sgtitle(['FRF for first four sensors of ' selectionname])
set(gcf,'position',[0,0,1920,1080])
fontsize(gcf,scale=1.5)
```

```
saveas(gcf, [selectionname '2.fig'])
saveas(gcf,[selectionname '2.svg'])
figure
modalsd(frf,f,fs,"FreqRange",[10,2000])
title(['Stabilization Diagram for ' selectionname])
set(gcf,'position',[0,0,1920,1080])
fontsize(gcf,scale=2)
legend('Location','northoutside','Orientation','horizontal')
saveas(gcf, [selectionname 'stab.fig'])
saveas(gcf,[selectionname 'stab.svg'])
figure
[fn,dr,ms,ofrf]=modalfit(frf,f,fs,50,'FitMethod','lsrf','FreqRange',[10,2000]);

end



function y = preprocessPlate(x,Fs)
%  Preprocess input x
%    This function expects an input vector x and a vector of time values
%    tx. Fs is the sample rate in Hz
%    Applies lowpass filter with stopband of 2000Hz, cutoff steepness of .9999
%    and stopband attenuation of 100db

% Generated by MATLAB(R) 9.13 and Signal Processing Toolbox 9.1.
% Generated on: 01-May-2023 12:51:03

y = lowpass(x,2000,Fs,'Steepness',0.9999,'StopbandAttenuation',100);
end

%% Import Lab Data
function [t, avgSig] = expDat(sigdat)

%Extract Averaged Data from 10 sets

t=sigdat.Time;
channels=fieldnames(sigdat);

for i = 2:numel(channels)

  avgSig(:,i-1) = meanFromStruct(sigdat.(channels{i}));

end

end
%
function [Average] = meanFromStruct(channel)
% Extract Signals from the given Channel
%Get the channel Subfield names. This assuems the last four fields are the
%Average, Avg Mean, std dev, and stdev mean, which can be ignored.

    fields=fieldnames(channel);

    %Loop through the signals and store in sigTab, then compute the mean of the
    %rows of the signal.
```

```
    for i = 1:numel(fields)-4

        sigTab(:,i)=channel.(fields{i});
    end
Average=mean(sigTab,2);
end
```