Structural identification of bridges: development of an integrated system for continuous dynamic monitoring.

Identificazione strutturale di ponti: sviluppo di un sistema integrato per il monitoraggio dinamico in tempo reale.

Laureando:	Alberto Lorenzon
Relatore:	Prof. Ing. Claudio Modena
Correlatore:	Dr. Ing. Kleidi Islami
Correlatore:	Ing. Mauro Caldon

Dipartimento di Ingegneria Civile, Edile ed Ambientale

Università degli Studi di Padova

Anno Accademico 2012-2013

Abstract

This thesis focuses on dynamic structural identification of existing structures by means of output-only modal analysis (OMA). First an improved implementation, using Matlab® environment, of a particular Operational Modal Analysis method was developed, enabling the extraction of damping ratio. Subsequently the development of a program with a graphical user interface was performed in order to allow structural dynamic identification and automatic monitoring using two OMA procedures. Finally, two applications of the program are presented in detail and used in order to observe and validate the results in the context of real structures.

Questa tesi si focalizza sull'identificazione strutturale dinamica di strutture esistenti per mezzo di metodi di Analisi Modale Operazionale (OMA). Inizialmente é stata sviluppata un'implementazione di un particolare metodo di OMA, che permette di estrarre il parametro di damping ratio, usando l'ambiente Matlab®. Successivamente é stato sviluppato un programma con interfaccia grafica, che permette di svolgere operazioni di identificazione dinamica e monitoraggio dinamico automatico di strutture esistenti utilizzando due metodi di analisi modale output-only. Infine sono state considerate nel dettaglio due applicazioni del programma al fine di osservarne e validarne i risultati relativamente a due strutture reali.

Table of Contents

Chapter 1

Introduction

1.1 Preface

The increasingly demanding safety requirements introduced in the regulatory framework related to structural engineering field and the newly available technologies have created, in the last few decades, new challenges for the scientific understanding of existing structures.
In the Italian context, characterized by a high seismic risk, it is necessary not only to design new earthquake-resistant structures, but above all to be able to correctly identify the dynamic behaviour of existing structures and monitor them over time in order to ensure structural safety, since existing structures built before the seismic regulations represent almost the totality. Particular attention must be placed on structures of historical and strategic nature. Current design and assessment procedures do not have much quantitative linkage with the characteristics of actual constructed systems; such gaps are filled by qualitative observation, anecdotal experiences, or laboratory studies reduced to a minimum because of the need to contain costs. For centuries, the engineers were forced to rely on extremely simple and idealistic models of constructed systems for analysis and design. As the profession moves towards more performance-based design approaches, and begins to more seriously consider durability, maintenance, and serviceability limit states as well as struggle with contemporary challenges associated with preservation and renewal, such simple modeling approaches are becoming inadequate.

Today more sophisticated modeling approaches are readily available, which have the ability to simulate both the three-dimensional local and global behaviors of constructed systems, but it is proven that, to ensure that they are a truthful representation of the real structure, a calibration is required using experimental response data from the actual structure.
Activities of structural identification can aim the observation of many characteristics of the structure, of both static and dynamic character. Static structural identification is essential to conduct activities of Structural Health Monitoring, as it allows to observe, for example, how the geometry of the elements changes over time (opening of the cracks, deformation, etc..). Dynamic structural identification has as its objective the extraction of the dynamic parameters of the real structure and allows, in addition to allowing monitoring the health of the structure from the observation of any changes in dynamic characteristics, to calibrate mathematical models based on real data.
During the last few decades modal analysis has become an increasingly important resource to conduct identification, monitoring and optimization of the dynamic characteristics of civil structures.
Modal analysis provides as results the modal parameters of the structure, namely *natural frequencies, mode shapes and damping ratios.* A particular type of Modal Analysis is considered within this thesis: the *Operational Modal Analysis (OMA).* In the last few years OMA has acquired a broad application in structural and mechanical engineering fields because it measures the response of the structure without interrupting its functionality. The structure, in a OMA analysis, is in fact subject to its natural excitations, which may be, in case of a bridge, due to wind, vehicular traffic or micro-seismic vibrations. The fact that OMA does not need to perform shaker tests or impact tests, allows the analysis to be performed over a long time span without compromising the structure's operability and therefore a it is actually possible to perform a monitoring of the structure over time. Structural monitoring offers a series of applications in the field of civil engineering, such as design, damage evaluation, maintenance and reinforcement of the existing structures.
Of high interest for this thesis is the Frequency Domain Decomposition method. It is a quite popular method of modal analysis operational that operates in

the frequency domain. In the context of the University of Padova, the FDD method has been studied and used for some years in order to conduct identification and monitoring of many bridges and structures of historical interest. Another OMA method that will be considered is Stochastic Subspace Identification.
In this thesis, on the basis of a first implementation of the FDD method conducted by Piovesan D. in [36] and of the SSI method conducted by Islami K. in [24], the implementation of the first method was improved, using Matlab® environment, and both methods were considered with two major goals:

- To develop an algorithm for Damping Ratio estimation using the Enhanced Frequency Domain Decomposition;
- To develop a Graphical User Interface for EFDD and SSI algorithms in order to allow automatic continuous monitoring of the structure.

Thus, by means of the methods and tools discussed later in this thesis, the development a procedure of structural identification that provides a scientific and objective description of some of the parameters of the existing structure was carried out. Two study cases are considered, in order to show how the program works and to validate its results with other methods.

1.2 Overview of chapters

Chapter 2 offers an overview of structural identification and monitoring methods. General purposes and applicability are discussed. Structural Health Monitoring is introduced in order to set a context for the analysis. Modal models are introduced, focusing on Operational Modal Analysis, which is the core method considered in this thesis. An historical description of the available methods is carried out.
Chapter 3 contains all of the theoretical basis required for the description of the methods used. First the dynamic description of simple mechanical system is presented. Then, signal analysis and random data analysis are considered. Finally the methods implemented in the thesis (AutoEFDD and SSI) are ex-

plained.
Chapter 4 consists of the user manual for the computer program developed for this thesis (**SIM/AtOMA**). In this chapter the functions of the program are explained by a user perspective and an application of the program is given for the case of a bridge in Peschiera sul Garda (Ponte sul Mincio).
Chapter 5 provides a detailed description of the code behind the program. The main functions of the interface are explained with a focus on the way data is passed between functions and between objects. In particular, an in-depth description of the algorithm for the determination of the damping ratio with Enhanced Frequency Domain Decomposition method, which was developed during the present thesis, is provided.
Chapter 6 shows an application of the monitoring systems detailed in this thesis for a bridge in Verona, namely *Ponte Nuovo del Popolo*. In this application, the results of a previous identification analysis conducted using a commercial software are compared to those provided by **SIM/AtOMA**.

Chapter 2

Overview of Structural Identification and Monitoring

2.1 Introduction

Every structure, whether it is a bridge or a building, is subjected to degradation. The materials accumulate damage, which in the long run can affect the operability of the structure. The excitations that involve the structure may vary in intensity and characteristics, causing unexpected behaviours, and the boundary conditions and the system constraints may change over time due to an inadequate maintenance.
All these phenomena make it difficult, if not impossible, to understand the actual behaviour of the structure on the only basis of physics-based modeling, i.e. using Finite Elements method, because the knowledge of the structure is very limited. Original designs of the structure, and inspection and testing of the materials in laboratory environment, hardly provide enough information in order to develop a model of the structure that actually describes its global characteristics.

The answer to these needs for *real* information on the *actual* structure lies in **Structural Identification**.

Structural identification (St-Id) is a paradigm that allows the understanding of a certain set of parameters of the structure on the basis of measured data.

The applications of St-Id are far wider than the only model calibration. St-Id allows in fact to perform decisions based on the real status of the structure, as well as many other tasks, as it will be addressed further on this chapter.

Mathematical models are a convenient way of describing the characteristics of a dynamic system. In this thesis the interest was put in modal analysis methods, due to their ability to provide the dynamic characteristic of the structure through modal parameters.
In this chapter the purposes of system identification (Sys-Id), including Structural Health Monitoring and model calibration, will be introduced, and specified for the structural context (St-Id). Then the different types of modal analysis methods will be described, with a focus on Operational Modal Analysis (OMA).
Finally, an overview of available OMA methods will be carried out.

2.2 System Identification

System Identification (Sys-ID) deals with the problem of building mathematical models of dynamical system based on observed data from the systems [30]. A *system* is, in general terms, an object in which variables of different kinds interact and produce observable signals, usually called *outputs*. The system is also affected by external stimuli. Those that can be manipulated by the observer are called *inputs*, others are called *disturbances*. Disturbances can be divided into those that are directly measured (for example, temperature, humidity) and those that are only observed through their effect on the output signal.
It might also be that the actual input is unknown and therefore uncontrollable in some applications. Thus, the output will be a mixture of dynamic response of the system and characteristics of the input and disturbance as well.
The way the system variables relate to each other constitutes a *model* of the system. For structural applications, it is necessary to use models that describe the relationships among the system variables in terms of mathematical expres-

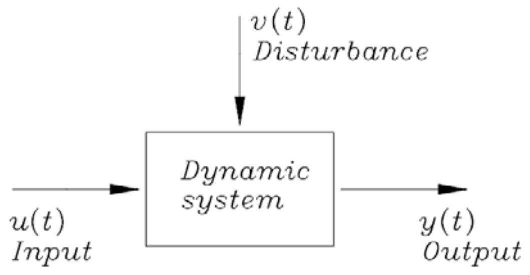


Figure 2.1: A dynamic system with input $u(t)$, output $y(t)$, and disturbance $v(t)$.

sions. There are two fundamental ways for building a model. A first way, known as *modeling*, splits up the system into subsystems, whose properties are well understood from previous experience. These subsystems are then assembled mathematically and a model of the whole system is obtained. This is how, for example, FEM models are built.
The other way is directly based on experimentation. Input and output signals from the system are recorded and subjected to data analysis to infer a model. Therefore *system identification* aims to develop a model of a dynamic system based on experimental data. System identification is not a substitute of physical modeling, since identification can be based on model structures that have physical origin, but physical modeling needs to be linked with parametric models of a structure.
To provide a mathematical definition of Sys-ID, consider a dynamic mechanical system. The Sys-ID process begins by hypothesizing a mathematical representation of the system, such as a state-space model [33]:

$$x(t) = F(x(\tau), \quad \tau < t; \qquad u(\tau), \quad \tau < t) \tag{2.2.1}$$

$$y(t) = G(x(\tau), \quad \tau < t; \qquad v(\tau), \quad \tau < t) \tag{2.2.2}$$

where $x(t)$ represents the various states of the system, $u(t)$ denotes the external inputs to the system, $y(t)$ denotes the outputs of the system, and $v(t)$ denotes observation uncertainty (all at time instant t), $F(\cdot)$ is the functional relation of state transition, and $G(\cdot)$ is the functional relation of outputs and states. In general, the states which define the system are not directly observable. As a results Sys-ID aims to infer the states $x(t)$ from the output $y(t)$, and in some cases the input $u(t)$ and measurement uncertainty $v(t)$, by finding an appropriate estimator $H(\cdot)$, such that:

$$\hat{x}(t) = H(y(\tau), \quad \tau \leq t; \qquad u(\tau), \quad \tau \leq t; \qquad v(\tau), \quad \tau \leq t) \tag{2.2.3}$$

2.3 Structural Identification

2.3.1 Introduction

Structural identification (St-ID) is a subset of Sys-ID which has been adapted to mechanical and civil structural systems, and may be defined as *"the parametric correlation of structural response characteristics predicted by a mathematical model with analogous quantities derived from experimental measurements* [33].

St-ID aims to fill the gap between the model and the real system by developing reliable estimates of the performance of structural system through simulations.

Historical development The origins of St-ID are soundly founded on the scientific method of observation (experiment), hypothesis (model) and validation. This approach is rooted in the argument that the establishment of reality requires that mathematical models would be hypothesized and validated based on observations of the physical world. This concept dates back to Plato, who claimed that "*the reality which scientific thought is seeking must be expressible in mathematical terms, mathematics being the most precise and definite kind of thinking of which we are capable*". These ideas embody the scientific method and also provide the foundation for the concept of St-ID. In its most general form, St-ID aims to establish the relationship between the Physical, Mental and Platonic-Mathematical worlds. Up to modern times, in 1907 Robert Maillart argued that designers should be encouraged to check their analytical assumption through load tests that establish deflections of the completed structure [4]. The modern applications of St-ID have their origins in systems engineering, which began to grow during the late 1950's. The advent of computers permitted extensive simulation and evaluation of systems, subsystems, and components. St-Id is a transformation and application of Sys-Id to mechanical (manufactured) and civil (constructed) structural systems. The paradigm was first introduced to engineering mechanics researchers by Hart and Yao (1977) and to Civil-structural engineering researchers by Liu and Yao (1978). Over 30 years later St-Id remains an active research area in both mechanical and civil-structural engineering.

Purpose of St-ID The use of FE models in Civil Engineering in recent years has seen a huge spread, linked to the widespread diffusion of computational tools and the steady increase in computing power. However, regardless of the degree of precision of the FE analysis, the description of existing systems is affected by an uncertainty that can not be filled except with calibration and validation based on actual observations and experimental measurements.
Structural identification focuses on creating a model of a dynamic system based on its measured response.

Reliability of St-ID Although St-ID is principally motivated by the need to quantify and reduce uncertainty, its reliable implementation is also affected by uncertainties, which must be properly addressed and quantified to provide accurate estimates of the reliability of the identified model. One of the most widely investigated and reported sources of uncertainty in making measurements on constructed systems involves their observed non-stationary behavior caused by environmental inputs such as temperature and humidity. This type of uncertainty can be effectively reduced by using ARX models to distinguish the effects of temperature fluctuations from real damage events on the parameters of the structure.
The difficulty in proving the reliability of a St-Id method is, however, still a major challenge for St-Id. Most reported examples of St-Id have been in the realm of research, and were considered successful once an agreement between the measured and simulated properties was established. Unless this definition of success is expanded to explicitly include the ability of the application to influence the decision-making process, the use of St-Id won't be as widespread

as it could be. Therefore, the validation of a St-Id model is a critical step, in the perspective of using the model to perform decisions of economic nature regarding the structure.

Application Scenarios: There are several scenarios for which the introduction of a model of structural identification would be advisable. Examples include [4]:

1. Design verification and construction planning in case of challenging new designs.
2. A means of measurement-based delivery of a design-built contract in a performance -based approach.
3. Documentation of as-is structural characteristics to serve as a baseline for assessing any future changes due to aging and deterioration, following hazards, etc.
4. Load-capacity rating for inventory, operation or special permits.
5. Evaluation of possible causes and mitigation of deterioration damage and/or other types of performance deficiencies.
6. Structural intervention, modification, retrofit or hardening due to changes in use-modes, codes, aging, and/or for increasing system-reliability to more desiderable levels,
7. Health and performance monitoring for operational and maintenance management of large systems.
8. Asset management of a population of constructed systems.
9. Advancing our knowledge regarding how actual structural systems are loaded, how they deform and how they transfer their forces through the members to foundations and to soil.

Steps for Structural-Identification of Constructed Systems St-ID requires the integration of *analytical, experimental* and *information* technologies. While all aspects of the St-ID process are critical to the overall success of the effort, it is important to recognize that analytical modeling is perhaps the most significant step. St-ID is usually involved with six basic steps:

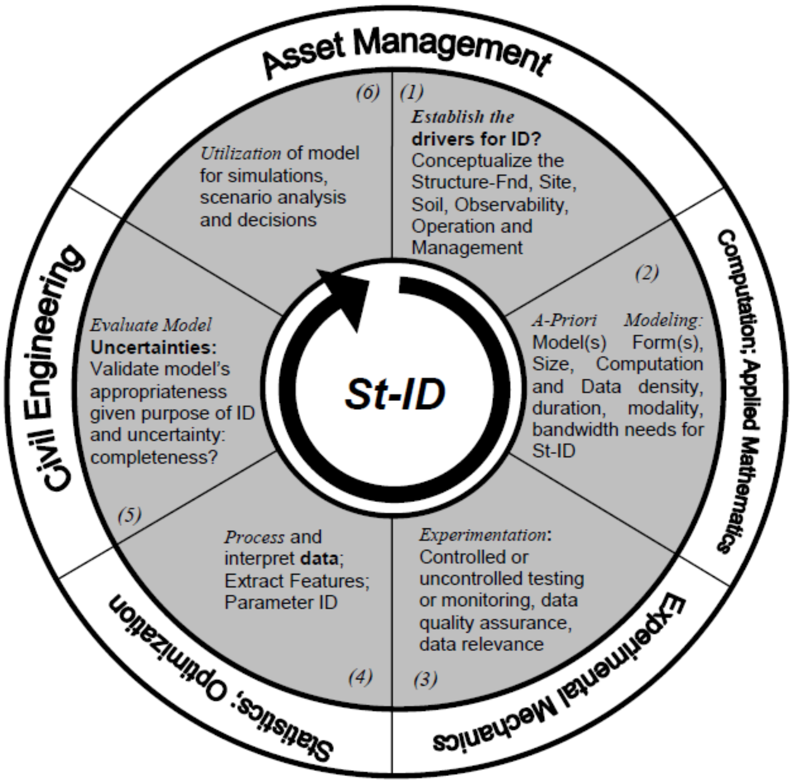

Asset Management
Computation; Applied Mathematics
Experimental Mechanics
Statistics; Optimization
Civil Engineering
(1) Establish the drivers for ID? Conceptualize the Structure-Fnd, Site, Soil, Observability, Operation and Management
(2) A-Priori Modeling: Model(s) Form(s), Size, Computation and Data density, duration, modality, bandwidth needs for St-ID
(3) Experimentation: Controlled or uncontrolled testing or monitoring, data quality assurance, data relevance
(4) Process and interpret data; Extract Features; Parameter ID
(5) Evaluate Model Uncertainties: Validate model's appropriateness given purpose of ID and uncertainty: completeness?
(6) Utilization of model for simulations, scenario analysis and decisions
St-ID

Figure 2.2: St-ID stages [4].

1. Observation and Conceptualization: definition of the investigation objectives.

2. A-Priori Modeling: a-priori models can be created on the basis of original drawings, in order to provide analytical prediction, that is often useful to determine optimal experimentation parameters such as critical sensor locations and frequency band of interest.

3. Measurement, Monitoring, Controlled Experimentation.

4. Integration, Processing, and Interpretation of Data. Data processing

activities aim to make the acquired data more appropriate for interpretation.

5. Evaluating Modeling Errors, Model Calibration & Check Model Completeness. This step involves the selection and calibration of physics-based models. These models are formulated to explicitly recognize the underlying physics of the constructed system. If direct data interpretation is employed in the St-Id application, this step may be optional. However, this step is crucial to informing decisions.

6. Utilization of Model for Simulation, Scenario Analysis. The ability to utilize the mode developed and calibrated (PB) or trained (NPB) through the St-Id process for decision-making is essential if the application is to be justified from an economic standpoint.

Today, it is not possible for any one individual to claim expertise throughout the entire spectrum of critical knowledge pertinent to the St-Id of constructed systems.
The 4th step (data processing) is the one mainly considered within this thesis, as in case of dynamic structural identification, it leads to the extraction of the modal parameters of the structure.

Analytical modeling forms for structural identification Generally existing analytical methods can be broadly classified as two categories, physics-based models (PB) and non-physics-based models (NPB), as shown in Table. Physics-based models are often preferred because most of the associated model parameters always have a clear physical meaning behind them.
Currently the the most commonly employed PB St-ID approach relies on modal analysis algorithms and linear(ized) finite element (FE) models [33]. Depending on how severely the structure being identified violates the implicit assumptions of this approach (i.e., linearity, stationarity, and observability), significant errors can occur. NPB techniques have the advantage that they are data-driven, so the construction of NPB models is dependent entirely on the data provided, but they may not be effective in the absence of PB models and heuristics.

Physics-based models	**Non-physics-based models**
1. Laws of Mechanics (a) Newton's Laws of Motion (b) Hooke's Law 2. Continua Models (a) Theory of Elasticity (b) Idealized Differential Equations (e.g. Beam theories of Bernoulli, Timoshenko, Vlasov) 3. Discrete Geometric (a) Idealized macro or element level models (e.g. idealized grillage models) (b) FEM for solids and field problems (c) Modal models i. Modal parameters (i.e. natural frequency, mode shape, damping) ii. Ritz Vectors	1. Semantic Models (a) Ontologies (b) Semiotic Models 2. Meta Models (a) Input-Output Models (b) Rule-based Meta Models (c) Mathematical (Ramberg-Osgood, etc.) 3. Numerical Models (a) Probabilistic Models i. Historigrams to Frequency Distribution ii. Standard Prob. Distributions iii. Independent events iv. Event-based v. Time-based vi. Symptom-based (b) Agents (c) Statistical (data-based) i. ARMA, ANN, others ii. Signal/Pattern Analysis, Wavelet, EMD, others

Table 2.1: PB and NPB models

2.3.2 Modal Analysis

Once the model form has been chosen, an appropriate technique must be selected in order to identify the parameters of that model. This selection depends on many factors, including the details of the instrumented degrees of freedom, the availability of measured input excitation, and the nature of the excitation of the system.
In the context of the St-Id methods considered in this thesis, we intend to introduce the concept of modal analysis.

What is modal analysis? As defined by He & Fu [22], modal analysis is *the process of determining the inherent dynamic characteristics of a system in forms of natural frequencies, damping factors and mode shapes, and using them to formulate a mathematical model for its dynamic behaviour.*
A modal model expresses the dynamical behaviour of the structure as a linear combination of different resonant modes. Each resonance mode is described in terms of its modal parameters: natural frequency, damping ratio, mode shape and participation vector.
Modal analysis embraces both theoretical and experimental techniques. In order to describe the dynamic behaviour of the system, a theoretical modal analysis uses a physical model of the system, comprising its mass, stiffness and damping properties, in form of partial differential equations. Finite elements analysis is perhaps the most used physical modal analysis technique.
On the other hand, modal analysis based on experimental data, namely modal testing, leads to the derivation of the modal model by developing mathematical relationships between the measured response of the structure and the measured excitations. This relationship is known as Frequency Response Function (FRF). The excitation can be of a selected frequency band, stepped sinusoid, transient, random or white noise. The last case is that of greatest interest in the context of this thesis as it provides a particular type of modal analysis, which will be seen in detail later, called Operational Modal Analysis.
An experimental modal model does not contain specific information about the structural connectivity or the geometric distribution of mass, structural damping, and stiffness. However, this kind of modal model is very useful since the

modal parameters are directly analogous to the eigen-solution of the structural mass and stiffness matrix, so it's well suited for the process of model correlation. Also, the structural frequency response function can be written in canonical form in terms of the modal parameters.

Modal Testing Modal testing is an experimental technique used to derive the modal model of a linear time-invariant vibratory system by defining the Frequency Response Function that relates the measured dynamic response of the structure with the applied excitation. Modal testing has been widely applied in vibration trouble shooting structural dynamics modification, analytical model updating, optimal dynamic design, vibration control, as well as vibration-based structural health monitoring in aerospace, mechanical and civil engineering [41]. Traditional *Experimental Modal Analysis* (EMA) makes use of input (excitation) and output (response) measurements to estimate modal parameters, consisting of modal frequencies, damping ratios, mode shapes and modal participation factors. In the last four decades numerous modal identification algorithms have been developed, in Time Domain (TD), Frequency Domain (FD) and Spatial Domain (SD):

- Single-Input/Single-Output (SISO)
- Single-Input/Multi-Output (SIMO)
- Multi-Input/Single-Output (MISO)
- Multi-Input/Multi-Output (MIMO)

Experimental Modal Analysis involves three constituent phases: test preparation, frequency response measurements and modal parameter identification. Traditional EMA has some major limitations. It requires artificial excitation in order to measure Frequency Response Functions (FRFs), or Impulse Response Functions (IRFs); it's normally conducted in lab environment, and it is difficult to use to determine FRF or IRF for large structures; to use it in a lab environment it is required to simulate reasonable boundary condition.

The main problem associated with forced vibration tests on bridges, buildings, or dams stems from the difficulty in exciting the most significant modes of vibration in a low range of frequencies with sufficient energy and in controlled manner [17]. In very large, flexible structures like cable-stayed or suspension bridges, the forced excitation requires extremely heavy and expensive equipment usually not available in most dynamic labs. Fortunately, recent technological developments in transducers and A/D converters have made it possible to accurately measure the very low levels of dynamic response induced by ambient excitations like wind or traffic. This has stimulated the development of output-only modal identification methods, also called *Operational Modal Analysis methods.*

Applications of Modal Models For engineers, the modal model is often not the final goal, but only an intermediate result that can be used for a wide range of applications [15]:

- *Model updating*: the initial values chosen for the material properties, geometrical properties and boundary condition of a FE model of a constructed structure can be updated using modal testing, to more realistic values.
- *Structural health monitoring and damage detection*: given a reference model of a healthy undamaged structure a modal model can provide a tool to perform a decision on the structural integrity of the structure by comparing newly estimated models to the reference one.

Other applications include:

- Response prediction;
- Sensitivity analysis and structural modification;
- Sub-structuring;
- Load identification;
- Vibro-acoustics.

2.3.3 Operational Modal Analysis

Operational modal analysis (OMA) has attracted great attention since early 1990's in the civil engineering community. OMA, also named as *natural-excitation* or *output-only* modal analysis, utilizes only response measurements of the structures in operational condition, subjected to ambient or natural excitation, in order to identify modal characteristics.
The main difference between OMA and EMA is therefore the source of applied forces acting on the structures. Ambient excitation usually provides multiple inputs and a wide-band frequency content thus stimulating a significant number of vibration modes. The identification algorithm used for OMA is therefore MIMO-type. For simplicity, output-only methods assume that the excitation input is a zero-mean Gaussian white noise. So, the real excitation can be expressed as the output of a suitable filter excited with white noise input.

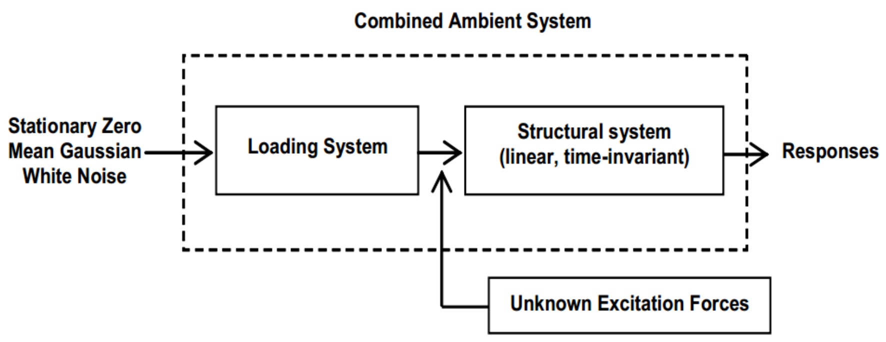

Combined Ambient System
Stationary Zero Mean Gaussian White Noise
Loading System
Structural system (linear, time-invariant)
Responses
Unknown Excitation Forces

Figure 2.3: Operational Modal Model [20].

As previously introduced, in output-only modal testing, the testing is normally done by just measuring the responses under the natural (ambient or operational) conditions. This avoids the need to interrupt the ordinary use of the structure in analysis. This means, for instance, that if a bridge is going to be tested, the bridge traffic and normal operation need not to be interrupted during the test and will be used as the excitation source. The response of the structure to the loading, as well as to natural excitations acting on it, will be measured and used to perform an output-only modal identification.

OMA has become particularly popular over the past two decades due to some of its advantages compared to traditional experimental modal testing and analysis:

- It is cheap and fast to conduct, since it doesn't require any excitation equipment or boundary condition simulation;
- OMA provides a dynamic characterization of the complete system in its actual environment;
- Owing to the use of real random forces applied to different points of structure, a linear model is obtained around the operating conditions rather than experimental condition;
- OMA is an appropriate method for complex and complicated structures, since in this method the analysis is MIMO-type and close modes can be easily identified;
- OMA is also suitable for conducting a monitoring of the structure and damage detection, as it does not interfere with the functionality of the structure[32]: .

Review of Operational Modal Analysis methods

There are two main groups of OMA methods: non-parametric methods, essentially developed in frequency domain, and parametric methods in the time domain. Five major categories of OMA methods can be identified:

1. *NExT*: Natural Excitation Technique;
2. *SSI*: Stochastic Subspace Identification;
3. *FDD*: Frequency Domain Decomposition;
4. *ARMA*: Auto-Regression Moving Average;
5. *Stochastic Realization.*

A brief description of these five categories will be provided. Then, the methods implemented in the system identification and monitoring procedures presented in this thesis (namely the FDD and SSI), will be analyzed from a theoretical point of view.

NExT-type procedures Natural Excitation Technique (NExT) was first introduced by James et al. [27] in 1992 and it is a time domain OMA method. This technique considers the correlation function (COR), given by the random response of a structure due to ambient excitation, as a summation of decaying sinusoids. Each decaying sinusoid is characterized by its damped natural frequency, damping ratio and mode shape coefficients, which are related to the parameters of a structural modes. Therefore the first step in NExT methods consists on obtaining a Time Response Function (TRF) and the second step leads to the extraction of the modal parameters by one of the common methods in Time Domain. In order to obtain the TRF, many methods have been proposed:

- Inverse FFT ot the FRF estimation to obtain IRF for time domain EMA;
- Free decay response (FDR) can be measured from transient excitation or sudden termination of broad band random excitation;
- Correlation function can be estimated from stochastic response via correlogram, or from power spectral density via Inverse Fourier Transform;
- Random decrement provides an estimate of the Correlation function.

Major multi-input/multi-output (MIMO) TD modal identification procedures developed in traditional EMA can be adopted for OMA:

- Improved Polyreference Complex Exponential (IPRCE);
- Eigensystem Realization Algorithm (ERA);
- Extended Ibrahim TD (EITD).

It is important to notice that NExT-type modal identification procedures adopted for EMA are all developed in the deterministic framework, but in OMA the data utilized for modal parameter estimation are of random response, and belong to a stochastic process.

SSI procedures Stochastic subspace base methods are one of the modal identification methods in time domain used in OMA.
The Stochastic Subspace Identification (SSI) derives from systems and control engineering and uses a subspace-based method to identify the system's state-space matrices.
SSI was first applied to measured random response of structures by Van Overchee and De Moor in 1996 in [39].
The Stochastic Subspace Identification Method is based on the state-space description of the system. The main steps necessary for a SSI procedure are:

- Computation of the projection of the *future* outputs on the basis of *past* outputs via numerical techniques;
- Estimation of Kalman filter state via SVD of the projection matrix;
- Estimation of the discrete-time system matrices via LS techniques;
- Calculation of modal parameters.

SSI methods are preferable to methods such as NeXT since they do not require the computation of the correlation functions. Since SSI makes direct use of stochastic response data to identify modal parameters, it is also called data-driven SSI.
An in-depth theoretical description of SSI method is provided in chapter **3.6.2**.

FDD-type procedures Frequency Domain Decomposition (FDD) is a frequency domain OMA method, which is based on the MIMO input-output relationship for a random process [6]. Frequency Domain techniques provide several advantages over TD techniques including the speed and simplicity, and the fact that FD methods have no bother with computational modes. However, FD methods also have some disadvantages, which will be presented later.
FD OMA methods are based on the assumption that input Power Spectral Density function is considered as a constant, so the PSD of the response of the structure can be considered, by means of modal decomposition, as the superposition of the effects due to the different modes of vibration of the structure,

which are identifiable by the peaks of the PSD.
The simplest Frequency Domain method consists in *Peak-Picking*, where the peaks of the Power Spectral Density calculated from measured data represent a structural mode, and lead to the extraction of an estimate of the modal parameters. PP method gives reasonable modal estimates if the modes are well separated [40], but it may not be very accurate in case of complex structures due to the dependency of the result to the resolution of the PSD spectrum; furthermore, operational deflection shapes is obtained instead of real modes shapes, and damping ratio estimation via half-power point is inaccurate or impossible.
Brinker et al. [13] proposed in 2000 a new OMA method in frequency domain, called *Frequency Domain Decomposition* (FDD). FDD requires the calculation of the Singular Value Decomposition (SVD) of the output Power Spectral Density, estimated at discrete frequencies, so that, when only a mode is dominating at a corresponding modal frequency, the PSD matrix approximates to a rank one matrix and the SV function can be utilized as modal indication functions, while modal frequencies can be located by the peaks of the SV plots.
Further developments [12] allowed the calculation of damping ratios by taking back to time domain the SDoF auto-spectral functions and subsequently evaluating by linear regression the logarithmic decrement of the envelope of maximum values of the SDoF free decay function. This technique is called *Enhanced Frequency Domain Decomposition*. Recently, a new generation of FDD, *Frequency-Spatial Domain Decomposition* (FSDD), has been developed [42] in order to improve the PSD through the use of spatial measurements.
Also, combined methods which use a frequency domain curve-fitting version of the EFDD technique have been proposed by Jacobsen [26] in order to extract structural modes even when several harmonic components are present and even when the harmonic components are located exactly at the natural frequencies of the structural modes.

ARMA-type procedures *Auto-Regression Moving Average* (ARMA) methods can be employed for Operational Modal Identification. Since in Operational Modal Analysis the input source consists in multiple stochastic excita-

tion, a multidimensional ARMA model (ARMAV) can be applied. The use of Prediction-Error Method (PEM) for ARMAV models was proposed in 1997 by Andersen [2]. In 1999, Brinker and Andersen [3] used ARMA models definition in state space as a new way for estimating ARMA model. In 2006, Moore et al. [35] presented theory and application of the ARMAX method to consider the cases characterized by high noise level and periodic and random input excitations.
PEM-ARMAV-type OMA procedures have, however, two major drawbacks [41]: they are very computational intensive and they require an initial "guess" in order to identify the parameters. These disadvantages make ARMAV-type procedures rather difficult to apply, especially for large dimension structures.

Stochastic Realization method System Realization, i.e. recovering or identification of system matrices, is a concept developed in the '60 by Ho & Kalman. The key feature of the stochastic system realization is the system decomposition of the covariance matrix instead of IRF matrix in deterministic system realization [41]. Stochastic Realization methods are often called as Covariance-driven Stochastic Subspace Identification (SSI) methods.

2.4 Applications

Of the many applications that were previously listed, two are of particular interest for the current thesis since they regard the two study cases that will be presented: *Structural Health Monitoring* and *Model Calibration*. A brief introduction of these two applications is provided below.

2.4.1 Structural Health Monitoring

Structural Health Monitoring (SHM) has been defined in the literature as the *"acquisition, validation and analysis of technical data to facilitate life-cycle management decisions"* [21]. More generally, SHM denotes a system with the ability to detect and interpret adverse "changes" in a structure in order to improve reliability and reduce life-cycle costs. The major challenge in designing a SHM system is knowing what "changes" to look for and how to identify them [28]. The damage signature will dictate the type of sensors that are required, which in-turn determines the requirements for the rest of the components in the system.

It is very important to define what is meant for *damage.* In most general terms, damage can be defined as changes introduced into a system that adversely affects its current of future performances [23]. Damage is only meaningful when two states of the system are compared. In structural and mechanical engineering, the definition of damage is limited to changes to the material and/or geometric properties of the systems, including changes to the boundary conditions and system connectivity that negatively affect the performance of the systems.

In terms of *length scale*, all damage begins at the material level and then under appropriate loading scenarios expands to component and system level damage. In terms of *time scale*, damage can accumulate incrementally over time, as for example, due to fatigue.

The basic premise of most damage detection methods is that damage alters the stiffness, mass, or energy dissipation properties of a system, which in turn alter the measured dynamic response of the system.

Damage identification using frequency changes The observation that changes in structural properties cause changes in vibration frequencies was the origin for using modal methods for damage identification and health monitoring [18]. Many methods use a *forward approach*, which consists of calculating frequency shifts from a known type of damage, which is modeled using FEM models. The measured frequencies are then compared to the predicted frequencies to determine the damage. Among others, an example of this type of application was presented by Brinker in [11], which used a statistical analysis method to detect damage in two concrete beams with different reinforcement ratios using changes in the measured vibration frequencies.

The major challenge in monitoring the damage of the structure by analyzing its dynamic response consists on the fact that damage is typically a local phenomenon which may not alter the lower-frequency global response of a structure that is normally measured using modal testing. Therefore it is not easy to state that shifts in the modal frequencies can be used to identify more than the mere existence of damage, since this parameter is a *global* property of the structure, and therefore, to conduct SHM operations, it is necessary to combine the dynamic monitoring system to a static monitoring system.

2.4.2 Model calibration

The calibration of a FE model on the basis of a Modal Testing can be performed by means of the following steps:

- Comparison of frequencies (FE) of the modal analysis with the frequencies obtained from the experimental data;
- Use of the MAC values (Modal Assurance Criterion), which provides an indication that the modal vectors experimentally measured and numerically calculated are consistent or not. Values of MAC near unity indicate a quasi-perfect correlation between reciprocal mode shapes (numerical vs. experimental).

The comparison between experimental and numerical frequencies is performed by means of the calculation of an average percentage error (ϵ):

$$\epsilon = \frac{f_{FEM} - f_S}{f_{FEM}} \cdot 100 \tag{2.4.1}$$

where:

- f_{FEM} is the numerical frequency;
- f_S is the experimental frequency.

The MAC index is defined as:

$$MAC(X,A) = \frac{|\sum_{j=1}^{n} \Phi_{X,j}\Phi_{A,j}|^2}{|\sum_{j=1}^{n} \Phi_{X,j}\Phi_{X,j}||\sum_{j=1}^{n} \Phi_{A,j}\Phi_{A,j}|} \tag{2.4.2}$$

where:

- $\Phi_{X,j}$ is the experimental mode shape vector;
- $\Phi_{A,j}$ is the numerical mode shape vector.

In this case a manual calibration can be applied in order to achieve a progressive improvement of the model, through successive changes and corrections on the mass and stiffness matrices.
Parameters related to the elastic properties of the materials, mass density and boundary conditions must iteratively be changed until the average percentage error assumed values less than 5% for each couple of numerical and experimental frequencies.
Experimental vs. numerical frequencies and mode shapes are compared during the calibration process, trying to minimize the average error in term of frequency and maximize the MAC index for each calculated/extracted mode shape. Various steps compose the calibration process: generally speaking an increase of the Young's modulus corresponds to an increase of the stiffness of the structure and therefore of frequencies, vice versa an increase of mass density and hence of the total mass of the structure corresponds to a decrease of frequencies.

Chapter 3

Theoretical Basis

3.1 Introduction

The purpose of this chapter is to provide the theoretical basis necessary for a proper understanding of the dynamic identification system considered and developed within this thesis. The dynamic identification methods used in Operational Modal Analysis make use of sets of measuring accelerations experienced by the structure in response to ambient excitation without exerting any controlled excitation. The methods defined in the frequency domain work by calculating the frequency response functions (FRF) from which the modal parameters of the structure are derived. The FRF are typically obtained by application of the Fast Fourier Transform (FFT). This chapter is developed following a logical path that first introduces the fundamental concepts (such as frequency response functions and impulse response functions) in the context of classical structural mechanics (initially for simple SDoF systems, then for MDoF systems), then considers them in the context of signal analysis. It is therefore important to understand how the random signals are acquired and processed in digital format. It is then indicated how the relations (functions of frequency response) between the input data and output data in the system are obtained. The considered system, in respect to the assumptions of operational modal analysis, is a system of the multi-input/multi-output type. Finally, the theoretical basis of the two methods in the dynamic identification system considered in the thesis are addressed: the FDD and SSI methods.

3.2 Transform Relationships

The theory from the vibrations point of view requires the comprehension of how the structural parameters of mass damping, and stiffness relate to the impulse response function (time domain), the frequency response function (Fourier, or frequency domain), and the transfer function (Laplace domain) for single and multiple degree of freedom systems.
The relationships between time, frequency, and Laplace domains, with respect to a structural system, are the integral transforms (Fourier and Laplace) that reflect the information contained by the governing differential equations transformed to each domain.
Adopting a digital approach to the measurement data leads to consider the discrete case, for which the relationships are represented by discrete transforms (Discrete Fourier Transform, Z Transform).
Both in the continuous case that in the discrete case, the transform process involves a change of independent variable which represents the change from one domain to another. The transformed variable must be the variable of interest, and the transform process must be computationally simple but unique, and it should not lead to loss of data.

3.2.1 SDOF Systems

Single degree of freedom systems, evaluated in the time, frequency, and Laplace domains serve as the basis for many of the models that are used in modal parameter estimation. In fact, the multiple degree of freedom case can be viewed as a linear superposition of single degree of freedom systems [1].

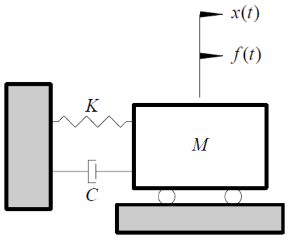


Figure 3.1: Single degree of freedom system.

The general mathematical representation of a SDOF system is expressed as follows:

$$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = f(t) \tag{3.2.1}$$

where:

- M = Mass constant
- C = Damping constant
- K = Stiffness constant

By setting $f(t) = 0$, the homogeneous form of the general equation can be solved:

$$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = 0 \tag{3.2.2}$$

The solution can be assumed to be on the form $x(t) = Xe^{st}$, where s is a complex constant to be determined. Substituting in the previous equation:

$$(Ms^2 + Cs + K)Xe^{st} = 0 \tag{3.2.3}$$

Thus, for a non-trivial solution:

$$Ms^2 + Cs + K = 0 \tag{3.2.4}$$

where:

- s = Complex-valued frequency variable (Laplace variable)

The previous equation is the characteristic equation of the system, whose roots λ_1 and λ_2 are:

$$\lambda_{1,2} = \frac{-C}{2M} \pm \left\{ \left(\frac{C}{2M} \right)^2 - \frac{K}{M} \right\}^{\frac{1}{2}} \tag{3.2.5}$$

Thus the general solution is:

$$x(t) = Ae^{\lambda_1 t} + Be^{\lambda_2 t} \tag{3.2.6}$$

A and B are constants determined from the initial conditions imposed on the system at time $t = 0$.

For most real structures the damping ratio is rarely greater than 10% and all

further discussion is restricted to undampered systems ($\zeta < 1$). In this case, the two roots, $\lambda_{1,2}$, are always complex conjugates and the two coefficients (A and B) are complex conjugates of one another ($B = A^*$).
For an undampered system, the roots of the characteristic equation can be written as:

$$\lambda_1 = \sigma_1 + j\omega_1 \qquad {\lambda_1}^* = \sigma_1 - j\omega_1 \tag{3.2.7}$$

where:

- σ_1 = Damping factor.
- ω_1 = Damped natural frequency.

The root of the characteristic Equation can also be written as:

$$\lambda_1, {\lambda_1}^* = -\zeta_1\Omega_1 \pm j\Omega_1\sqrt{1 - {\zeta_1}^2} \tag{3.2.8}$$

The *damping factor* σ_1 is defined as the real part of a root of the characteristic equation and it describes the exponential decay or growth of the oscillation.

Time Domain: Impulse Response Function (IRF)

The impulse response function of the SDOF system can be determined from the general solution, assuming:

- The initial conditions are zero;
- the system excitation, $f(t)$, is a unit impulse.

The response of the system, $x(t)$, to such a unit impulse is known as the impulse response function (IRF), $h(t)$, of the system. Therefore:

$$h(t) = Ae^{\lambda_1 t} + A^* e^{{\lambda_1}^* t} \tag{3.2.9}$$

$$h(t) = e^{\sigma_1 t}\left[Ae^{+j\omega_1 t} + A^* e^{-j\omega_1 t}\right] \tag{3.2.10}$$

The residue A controls the amplitude of the impulse response, the real part of the pole is the decay rate and the imaginary part of the pole is the frequency of oscillation.

Frequency Domain: Frequency Response Function

An equivalent equation of motion is determined for the Fourier or frequency (ω) domain. The advantage of this representation is that it converts a differential equation to an algebraic equation.
Fourier series is a representation of a periodic function. For a periodic time domain function $x(t)$ with period T, we have:

$$x(t) = x(t + nT) \tag{3.2.11}$$

Mathematically, $x(t)$ consists of a number of sinusoids with frequencies multiple to a fundamental frequency. The fundamental frequency f is set by the period such that $f = \frac{1}{T}$. The amplitude of the kth sinusoid can be determined by:

$$X(f_k) = \frac{1}{T} \int_{-T/2}^{T/2} x(t) e^{\frac{-j2\pi kt}{T}} dt \tag{3.2.12}$$

where:

- f_k is the kth frequency
- $e^{\frac{-j2\pi kt}{T}}$ represents a unit vector rotating at a frequency of $-\frac{k}{T}$

Thus, the equation of the general solution becomes:

$$\left[-M\omega^2 + jC\omega + K\right] X(\omega) = F(\omega) \tag{3.2.13}$$

Restating the above equation:

$$B(\omega)X(\omega) = F(\omega) \tag{3.2.14}$$

where:

- $B(\omega) = -M\omega^2 + jC\omega + K$

The system response $X(\omega)$ is directly related to the system forcing function $F(\omega)$ through the quantity $B(\omega)$ known as the *impedance function.*
If The system forcing function $F(\omega)$ and its response $X(\omega)$ are known, $B(\omega)$ can be calculated:

$$B(\omega) = \frac{F(\omega)}{X(\omega)} \tag{3.2.15}$$

Introducing the frequency response function $H(\omega)$ we can rewrite the equation as follows:

$$X(\omega) = H(\omega)F(\omega) \tag{3.2.16}$$

where:

- $H(\omega) = \frac{1}{-M\omega^2+jC\omega+K}$

The FRF relates the Fourier transform of the system input to the Fourier transform of the system response. The FRF can be defined as:

$$H(\omega) = \frac{X(\omega)}{F(\omega)} \tag{3.2.17}$$

The frequency response function can be written:

$$H(\omega) = \frac{1}{-M\omega^2 + jC\omega + K} = \frac{1/M}{-\omega^2 + j\left(\frac{C}{M}\right)\omega + \left(\frac{K}{M}\right)} \tag{3.2.18}$$

The denominator is know as the characteristic equation of the system.
The *characteristic values* of this equation are known as the *complex roots* of the characteristic equation or the *complex poles* of the system, and they are also called the *modal frequencies.*

The frequency response function $H(\omega)$ can also be written as a function of the complex poles as follows:

$$H(\omega) = \frac{1/M}{(j\omega - \lambda_1)(j\omega - {\lambda_1}^*)} = \frac{A}{(j\omega - \lambda_1)} + \frac{A^*}{(j\omega - {\lambda_1}^*)} \tag{3.2.19}$$

where:

- $\lambda_1 =$ Complex pole
- $\lambda_1 = \sigma + j\omega_1$
- ${\lambda_1}^* = \sigma - j\omega_1$

The frequency response function is a complex valued function of a real valued independent variable (ω).

Frequency response functions are of particular importance for the identification techniques in the frequency domain.

Types of Frequency Response Functions When a FRF is referred to without response parameter specification, it is usually denoted as $H(\omega)$. When a response parameter is specified, individual FRFs have their own denotation. In complex notation, displacement can be expressed as a function of time as:

$$x(t) = Xe^{i\omega t} \tag{3.2.20}$$

By derivation, the expressions for velocity and acceleration in complex notation are obtained:

$$v(t) = \dot{x}(t) = i\omega X e^{i\omega t} \tag{3.2.21}$$

$$a(t) = \ddot{x}(t) = -\omega^2 X e^{i\omega t} \tag{3.2.22}$$

FRF of type *receptance* with displacement as a response parameter is defined:

$$\alpha(\omega) = \frac{X}{F} \tag{3.2.23}$$

Other types of FRF can be obtained, using the derived forms of the displacement. When the response parameter is velocity, the FRF is called *mobility*, and when it's acceleration the FRF is called *inertance*.

Laplace transform and transfer function

The *Laplace transform* is a systematic technique for finding the solutions of a differential equation. For a time domain function $f(t)$, the transform is defined and denoted by [22]:

$$F(s) = \mathcal{L}(f(t)) = \lim_{n\to\infty} \int_0^n f(t)e^{-st}dt \tag{3.2.24}$$

In the context of vibration and modal analysis, this limit *exists* and is a function of s that is called the Laplace variable. In fact, the integration constitutes a transformation from the *time domain* signal $f(t)$ to the s domain.
An important use of Laplace transform in modal analysis is to convert a differential equation into an algebraic equation. For a SDoF system.
Assuming all initial conditions to zero, can apply the Laplace transform on the equation of motion, yielding an algebraic equation:

$$Ms^2X(s) + CsX(s) + KX(s) = P(s) \tag{3.2.25}$$

The response of the system becomes:

$$X(s) = \frac{1}{Ms^2 + Cs + K} P(s) \tag{3.2.26}$$

The transfer function of the system, which defines the relationship between the force input and the response in displacement, can be found as:

$$G(s) = \frac{X(s)}{P(s)} = \frac{1}{Ms^2 + Cs + K} \tag{3.2.27}$$

This transfer function becomes the frequency response function when only the imaginary part of the s operator is considered:

$$G(j\omega) = \frac{X(j\omega)}{P(j\omega)} = \frac{1}{-M\omega^2 + Cj\omega + K} \tag{3.2.28}$$

3.2.2 MDoF Systems

The real application of modal analysis concepts considers a multiple degree-of-freedom system, for a continuous non-homogeneous structure described as a lumped mass. The modal parameters (natural frequencies, damping ratios and mode shapes), can be found by estimating the mass, damping and stiffness matrices or measuring the associated frequency response functions [1].
Since there is no difference in concept between systems with two or more degrees of freedom, a simple two degree of freedom system is considered as the most basic example of a MDoF system.

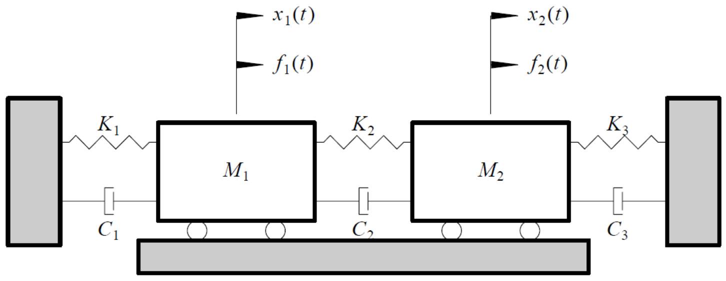


Figure 3.2: Multiple degree of freedom system.

The system has as many natural frequencies as the degrees of freedom. A mode of vibration is associated with each natural frequency. Since the equations of motion are coupled, the motion of the masses are the combination of the motions of the individual modes. If the choice of coordinates allows the uncoupling of the equations, each mode can be examined as an independent SDoF system.

Assuming viscous damping, the equations of motion are derived by applying Newton's second law to each of the masses. The equations of motion for the 2-DoF system in Figure, using matrix notation, are:

$$\begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \cdot \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} + \begin{bmatrix} (C_1 + C_2) & -C_2 \\ -C_2 & (C_2 + C_3) \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} + \\ + \begin{bmatrix} (K_1 + K_2) & -K_2 \\ -K_2 & (K_2 + K_3) \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1(t) \\ f_2(t) \end{bmatrix} \tag{3.2.29}$$

or

$$M\left\{\ddot{x}\right\} + C\left\{\dot{x}\right\} + K\left\{x\right\} = \left\{\ddot{f}(t)\right\} \tag{3.2.30}$$

where $f_1(t)$ and $f_2(t)$ are the excitation forces applied to the respective masses. For an undamped system, free of excitations and damping, the equations of motion are reduced to:

$$\begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \cdot \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{bmatrix} + \begin{bmatrix} (K_1 + K_2) & -K_2 \\ -K_2 & (K_2 + K_3) \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.2.31}$$

The equations are linear and homogeneous and are in the form of:

$$x_1 = B_1 e^{st} \tag{3.2.32}$$

$$x_2 = B_2 e^{st} \tag{3.2.33}$$

where B_1, B_2 and s are constants. since the system is undamped, the values of s are imaginary, $s = \pm jw$, and by Euler's formula the solutions must be harmonic:

$$x_1 = A_1 sin(\omega t + \phi) \tag{3.2.34}$$

$$x_2 = A_2 sin(\omega t + \phi) \tag{3.2.35}$$

where A_1, A_2 and ϕ are constants and ω is a natural frequency of the system. Therefore the equations of motion can be rearranged as:

$$(K_1 + K_2 - \omega^2 M_1)A_1 - K_1 A_2 = 0 \tag{3.2.36}$$

$$-K_1 A_1 + (K_1 + K_3 - \omega^2 M_2) A_2 = 0 \tag{3.2.37}$$

which are homogeneous linear algebraic equations in A_1 and A_2. By setting the determinant $\Delta(\omega)$ of the coefficients of A_1 and A_2 to zero, we obtain the frequency equation of the system from which the values of ω are found, that is:

$$\Delta(\omega) = \begin{vmatrix} K_1 + K_2 - \omega^2 M_1 & -K_1 \\ -K_1 & K_1 + K_3 - \omega^2 M_2 \end{vmatrix} = 0 \tag{3.2.38}$$

This leads to two real and positive values for ω^2, namely ω_1^2 and ω_2^2, whose roots in modulus are the natural frequencies of the 2 degree-of-freedom system.

Damping mechanism In order to evaluate real MDoF systems, the effect of damping on the complex frequencies and modal vectors must be considered. Some classical types of physical mechanism used to describe all of the possible forms of damping that may be present in a particular structure are:

1. Structural damping;
2. Viscous damping;
3. Coulomb damping.

Most structures exhibit damping characteristics that result from a combination of all the above. The model that is usually adopted is only concerned with the resultant mathematical form and represent a hypothetical form of damping , that is proportional to the system mass or stiffness matrix:

$$\{C\} = \alpha \{M\} + \beta \{K\} \tag{3.2.39}$$

Under this assumption, *proportional damping* is the case in which the equivalent damping matrix is equal to a linear combination of the mass and stiffness matrices. In this case, the coordinate transformation that diagonalizes the system mass and stiffness matrices, also diagonalizes the system damping matrix and the system of coupled equations of motion can be transformed to a system of equations that represent an uncoupled system of SDoF systems.

Response functions The development of the response functions for the MDoF case is analogous to the SDoF case. The mass, damping, and stiffness matrices are related to a transfer function model involving multiple degrees of freedom. The frequency response functions between any input and response degree of freedom can be represented as a linear superposition of the single degree of freedom models derived previously:

- *Impulse Response Function*:

$$\{h(t)\} = \sum_{r=1}^{N} \{A_r\} e^{\lambda_r t} + \{A_r^*\} e^{\lambda_r^* t} = \sum_{r=1}^{2N} \{A_r\} e^{\lambda_r t} \tag{3.2.40}$$

- *Frequency Response Function*:

$$\{H(\omega)\} = \sum_{r=1}^{N} \frac{\{A_r\}}{j\omega - \lambda_r} + \frac{\{A_r^*\}}{j\omega - \lambda_r^*} = \sum_{r=1}^{2N} \frac{\{A_r\}}{j\omega - \lambda_r} \tag{3.2.41}$$

- *Transfer Function*:

$$\{H(s)\} = \sum_{r=1}^{N} \frac{\{A_r\}}{s - \lambda_r} + \frac{\{A_r^*\}}{s - \lambda_r^*} = \sum_{r=1}^{2N} \frac{\{A_r\}}{s - \lambda_r} \tag{3.2.42}$$

where:

- t = Time variable
- ω = Frequency variable
- s = Laplace variable
- r = Modal vector number
- $A_r = Q_r \phi_{pr} \phi_{qr}$ Residue
- Q_r = Modal scaling factor
- ϕ_{pr} = Modal coefficient
- q = Measured degree of freedom (input)
- p = Measured degree of freedom (output)
- λ_r = System pole
- N = Number of positive modal frequencies

3.2.3 State-Space concept

The state space system model is frequently used in control and has also found applications in modal analysis. It can be applied in both the frequency and time domains. The main advantage of the state space form is that it reduces the second order differential equation that describes the system into two first order equations.

A state-space model is a different representation of the input-output relationship compared to the transfer or frequency response function approach [22]. The *state* of a dynamic system is the smallest set of variables which, together with the future inputs to the system, can determine the dynamic behaviour of the system, so the state at a time t is uniquely determined by the state at time t_0 and the inputs at time $t > t_0$.

Considering the SDoF system, governed by:

$$M\ddot{y} + C\dot{y} + Ky = f(t) \tag{3.2.43}$$

and applying to it a Laplace transform, the system can be derived as:

$$G(s) = \frac{Y(s)}{F(s)} = \frac{1}{Ms^2 + Cs + K} \tag{3.2.44}$$

Introducing two new variables:

$$x_1(t) = y(t) \qquad x_2(t) = \dot{x}_1(t) \tag{3.2.45}$$

The SDoF governing equation becomes:

$$\dot{x}_2(t) = \frac{1}{M}f(t) - \frac{C}{M}x_2(t) - \frac{K}{M}x_1(t) \tag{3.2.46}$$

It is now possible to define the *state equations*:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{K}{M} & -\frac{C}{M} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} f(t) \tag{3.2.47}$$

The state equations, which use only the first derivatives, can be used to represent the dynamics of the system.

For a MIMO system we can define n state variables $x_1(t), x_2(t), ..., x_n(t)$ such that each variable is the integrator of the one after it in the time domain, r

inputs $u_1(t), u_2(t), ..., u_r(t)$ and p outputs $y_1(t), y_2(t), ..., y_p(t)$. The state-space equations can be written in the following shape:

$$\{\dot{x}(t)\} = [A]\{x(t)\} + [B]\{u(t)\} \tag{3.2.48}$$

$$\{y(t)\} = [C]\{x(t)\} + [D]\{u(t)\} \tag{3.2.49}$$

The first equation is known as *the state equation*, while the second one is known as the *output equation.*
A is known as the system matrix or state transition matrix, B as the input matrix, C as the output matrix, D is the direct transmission matrix. For the SDoF system, the differential equation of the system can be recast as:

$$\{y\} = \left[-C_a M^{-1} K - C_a M^{-1} C_2\right]\{x\} + C_a M^{-1} B_2 \{u\} \tag{3.2.50}$$

$$y = Cx + Du \tag{3.2.51}$$

where C_a is the output location matrix for acceleration measurements (displacement and velocity can also be included), and B_2 relates the inputs to their physical location on the structure.

The discrete time state space model is more applicable in modal analysis where measurement data is sampled, and may be expressed as:

$$x_{k+1} = A_d x_k + B_d u_k \qquad y_k = C_d x_k + D_d u_k \tag{3.2.52}$$

where the system matrices are related to their continuous time counterparts by $A_d = e^{A_c \Delta t}, \quad B_d = (A_d - I)A_c^{-1} B_c, \quad C_d = C_c, \quad D_d = D_c$. An eigenvalue decomposition of A_d yelds:

$$A_d = \Psi \begin{pmatrix} \ddots & & \\ & \mu_i & \\ & & \ddots \end{pmatrix} \Psi^{-1} \tag{3.2.53}$$

where the eigenvectors are identical to the continuous time case, and the eingenvalues are related to the continuous time eigenvalues by $\mu = e^{\lambda_1 \Delta t}$. The modal participation matrix and observed mode shapes are expressed as for the continuous case.

Of greater relevance to OMA is the stochastic state space model which includes noise terms:

$$x_{k+1} = A_d x_k + B_d u_k + w_k \qquad y_k = C_d x_k + D_d u_k + v_k \tag{3.2.54}$$

where w_k and v_k are both unmeasurable, zero mean, white noise. The relevance of this model to OMA will be discussed in the SSI section.

3.3 Signal analysis

The concepts of frequency response function and impulse response function were earlier introduced in relation to the physical mechanical system.
In this section, the same concepts are addressed from the point of view of the signal analysis. The physical characteristics of the system, previously needed to define the response of the system, do not appear in the analysis. In fact, the signal analysis considers the observable signals of the system (the output), the signals that can be manipulated (the input), and those that cannot be manipulated but affect the system (noise). Signal analysis aims to build a model that reconstructs, by inference, the behaviour of the system.

3.3.1 Response Characteristics of Ideal Linear Systems

An ideal system is one that has *constant parameters* and is *linear* between the input and the output (or response). The dynamic characteristics of a constant-parameter linear system can be described by an *impulse response function* $h(\tau)$, also called the *weighting function.* For any arbitrary input $x(t)$, the system output $y(t)$ is given by:

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \tag{3.3.1}$$

The system must respond only to past inputs, so:

$$h(\tau) = 0 \qquad for \quad \tau < 0 \tag{3.3.2}$$

3.3.2 Frequency Response Functions

The dynamic characteristics of a constant-parameter linear system can be described by a *frequency response function* $H(\omega)$, which is defined as the Fourier transform of the impulse response function $h(\tau)$:

$$H(\omega) = \int_0^\infty h(\tau) e^{-j2\pi\omega\tau} d\tau \tag{3.3.3}$$

The frequency response function is a special case of the *transfer function* defined by the Laplace transform, where the complex variable has the real part equal to zero. Taking the Fourier transform of the input $X(\omega)$, and the Fourier transform of the output $Y(\omega)$, it follows that:

$$Y(\omega) = H(\omega) \cdot X(\omega) \tag{3.3.4}$$

The FRF is generally a complex-valued quantity that can be thought in terms of a magnitude and an associated phase angle, by writing $H(\omega)$ in complex polar notation as:

$$H(\omega) = |H(\omega)| e^{-j\phi(\omega)} \tag{3.3.5}$$

where:

- $|H(\omega)|$ is called the system *gain factor*
- $\phi(\omega)$ is called the system *phase factor*

The frequency response function $H(\omega)$ of constant-parameter linear system is a function of only frequency and is not function of either time or the system excitation.

3.3.3 Autospectrum

Given an input signal $a(t)$, the autocorrelation function is defined as:

$$R_{aa}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T a(t) a(t+\tau) dt \tag{3.3.6}$$

The Fourier transform of the autocorrelation function is the autospectrum, also named *Power Spectral Density* (PSD):

$$S_{AA}(\omega) = \int_{-\infty}^{\infty} R_{aa}(\tau) e^{-j2\pi\omega\tau} d\tau \tag{3.3.7}$$

The autospectrum is a function computed from the instantaneous Fourier spectrum as:

$$S_{AA}(\omega) = E\left[A(\omega)A(\omega)^*\right] = \mathcal{F}E\left[a(t)^* a(-t)\right] = \mathcal{F}\left[R_{aa}(\tau)\right] \tag{3.3.8}$$

where E denotes the *expected value*, $\mathcal{F}$ denotes the Fourier transform,

$$A(\omega) = |A(\omega)|e^{j\phi_A(\omega)} \tag{3.3.9}$$

$$A^*(\omega) = |A(\omega)|e^{-j\phi_A(\omega)} \tag{3.3.10}$$

and $R_{aa}(\tau)$ is the autocorrelation function. The definition of autospectrum

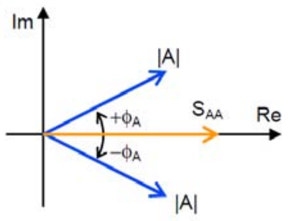


Figure 3.3: Auto-spectrum.

can be recast as:

$$S_{AA}(\omega) = E\left[|A(\omega)| \cdot |A^*(\omega)|e^{j0}\right] = E\left[|A|^2(\omega)\right] \tag{3.3.11}$$

3.3.4 Cross-spectrum

Given two signals, $a(t)$ and $b(t)$, the cross-correlation function is defined as:

$$R_{ab}(\tau) = \lim_{T\to\infty} \frac{1}{T}\int_0^T a(t)b(t+\tau)dt \tag{3.3.12}$$

The cross-correlation function is a measure of the correlation of two different signals.

Its Fourier transform is the Cross spectrum S_{AB} (from A to B):

$$S_{AB}(\omega) = \int_{-\infty}^{\infty} R_{ab}(\tau)e^{-j2\pi\omega\tau}d\tau \tag{3.3.13}$$

The cross-spectrum can be written in function of the Fourier transform of each complex instantaneous spectra $A(\omega)$ and $B(\omega)$:

$$S_{AB}(\omega) = E\left[A^*(\omega)B(\omega)\right] = \mathcal{F}E\left[a(-t)^* b(t)\right] = \mathcal{F}\left[R_{ab}(\tau)\right] \tag{3.3.14}$$

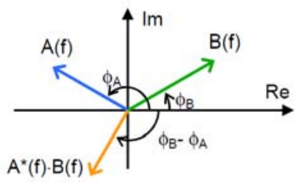


Figure 3.4: Cross-spectrum.

where:

$$A(\omega) = |A(\omega)| \cdot e^{j\phi_A(\omega)} \tag{3.3.15}$$

$$B(\omega) = |B(\omega)| \cdot e^{j\phi_B(\omega)} \tag{3.3.16}$$

Therefore, the cross-spectrum can be recast as:

$$S_{AB}(\omega) = E\left[|A(\omega)| \cdot |B(\omega)| \cdot e^{j(\phi_B(\omega)-\phi_A(\omega))}\right] \tag{3.3.17}$$

The amplitude of the cross-spectrum S_{AB} is the product of the amplitudes of $A(\omega)$ and $B(\omega)$, and it is a measure of the correlation between the functions of the frequency of the two signals.
The phase of the cross-spectrum is the difference of both phases (from A to B). It measures the phase shift between the function of frequency of the two signals.

The advantage of the cross-spectrum is that influence of noise can be reduced by averaging because the phase angle of the noise spectrum takes random values so that the sum of those several random spectra tends to zero. So while the measured autospectrum is a sum of the true autospectrum and the autospectrum of noise, the measured cross-spectrum is equal to the true cross-spectrum.

3.3.5 Coherence

Coherence function indicates the degree of linear relationship between two signals as function of frequency [7]. It is defined by two autospectra and a cross spectrum as:

$$\gamma^2(\omega) = \frac{|G_{AB}(\omega)|^2}{G_{AA}(\omega) \cdot G_{BB}(\omega)} \tag{3.3.18}$$

Coherence can be considered as a squared correlation coefficient at each frequency, where the magnitudes of autospectra correspond to variances of the two variables and the magnitude of cross-spectrum corresponds to covariance.

Coherence value varies from zero to one. A coherence value of zero indicates that there is no correlation between the two variables, whilst a coherence value of one means a perfectly linear relationship

$$0 \leq \gamma^2(\omega) \leq 1 \tag{3.3.19}$$

Coherence function provides useful information only when spectra $G_{AA}(\omega)$, $G_{BB}(\omega)$ and $G_{AB}(\omega)$ are estimates, i.e. spectra averaged from more records. In case of no averaging, coherence is always equal to 1.

Effect of Noise on the FRF

In reality, FRF measurement cannot be noise free and usually a measured signal, i.e. a force $\hat{F}(\omega)$ is a combined signal of the genuine force $F(\omega)$ and the noise from the input $M(\omega)$. These two are normally inseparable in the time domain but they are not correlated to each other [22]. Therefore, we have $S_{MF}(\omega) = 0$. This property is useful in deriving FRF estimators to combat noise. The same happens at the output end. The measured response $\hat{X}(\omega)$ encompasses both the true response $X(\omega)$ and the noise from the output $N(\omega)$. They are not correlated so that $S_{NX}(\omega) = 0$.
A dual channel analyzer allows three alternative estimates, defined using the auto-spectrum and the cross-spectrum:

$$H_1(\omega) = \frac{G_{AB}(\omega)}{G_{AA}(\omega)} \tag{3.3.20}$$

$$H_2(\omega) = \frac{G_{BB}(\omega)}{G_{BA}(\omega)} \tag{3.3.21}$$

$$H_3(\omega) = \sqrt{\frac{G_{BB}(\omega)}{G_{AA}(\omega)}} \cdot \frac{G_{AB}(\omega)}{|G_{AB}(\omega)|} = \sqrt{H_1(\omega) \cdot H_2(\omega)} \tag{3.3.22}$$

Coherence function could be defined as:

$$\gamma^2(\omega) = \frac{|G_{AB}(\omega)|^2}{G_{AA}(\omega) \cdot G_{BB}(\omega)} = \frac{H_1(\omega)}{H_2(\omega)} \tag{3.3.23}$$

The choice of the best estimate depends on whether there is noise on the input or output.

Influence of noise at the output A typical case of noise-influenced output is when FRF is measured using impact excitation: the input signal is clean, without noise, whilst the output signal is modified by system response and deteriorated by noise.

$$H(\omega) = \frac{V(\omega)}{A(\omega)} \tag{3.3.24}$$

$$H_1(\omega) = \frac{G_{AB}}{G_{AA}} = \frac{G_{AV}}{G_{AA}} = H \tag{3.3.25}$$

In case of noise-influenced output H_1 is an exact estimator of the FRF.

Influence of Noise at the Input. When FRF is measured using dynamic exciter, the input signal is deteriorated by noise in the vicinity of resonances, particularly for slightly damped structures. The structure behaves as short circuit in the vicinity of resonances and the input power spectrum has lower values even if the signal entering the exciter is white noise, while the output signal is relatively clean.

$$H(\omega) = \frac{B(\omega)}{U(\omega)} \tag{3.3.26}$$

$$H_2(\omega) = \frac{G_{BB}}{G_{BA}} = \frac{G_{BB}}{G_{BU}} = H \tag{3.3.27}$$

In case of noise-influenced output H_2 is an exact estimator of the FRF.

Influence of Noise at Both Input and Output. In the common case, both input ad output signals are noise-influenced. The exact estimation of the FRF stands between H_1 and H_2.

3.4 Analysis of Random Data

An observed data representing a physical phenomenon can be broadly classified as being either deterministic or non-deterministic. Non-deterministic data are those that can't be described by an explicit mathematical relationship, for which there is no way to predict an exact value at a future instant of time. These data are random in character and must be described in terms of probability statements and statistical averages rather than by explicit equations [6].

A single time history representing a random phenomenon is called a *sample function* (or a *sample record* when observed over a finite time interval). The collection of all possible sample functions that the random phenomenon might have produced is called a *random (or stochastic) process.* A sample record of data for a random physical phenomenon may be thought of as one physical realization of a random process.

Classification of random data

Operational Modal analysis deals with random data input, given by the natural excitement acting on the system. It is therefore essential to identify input/output relations in case of random data.
When a physical phenomenon is considered in terms of random process, its properties can be described at any instant of time by computing average values over the collection of sample functions that describe the random process. The *mean value* of the random process at some t_1 can be computed by taking

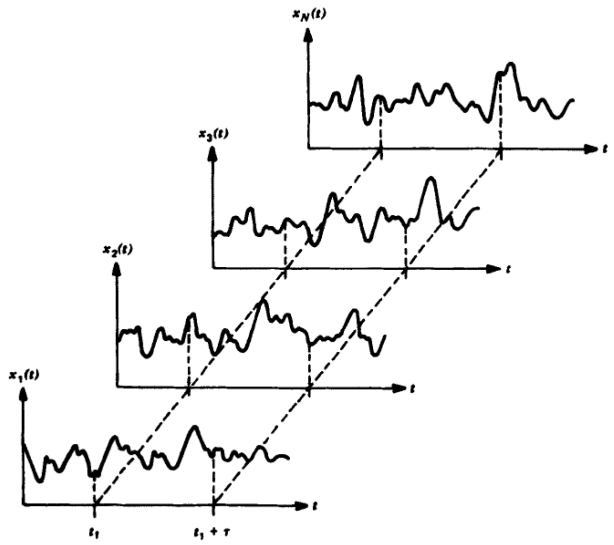


Figure 3.5: Ensemble of time history records defining a random process.

the instantaneous value of each sample function of the ensemble at time t_1, summing the values, and dividing by the number of sample functions. Similarly, a correlation between the values of the random process at two different times (called the *autocorrelation function*) can be computed by taking the ensemble average of the product of instantaneous values at two times, t_1 and $t_1 + \tau$. For the random process $x(t)$, composed of N sample functions, the mean value and the autocorrelation function are given by:

$$\mu_x(t_1) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} x_k(t_1) \tag{3.4.1}$$

$$R_{xx}(t_1, t_1 + \tau) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} x_k(t_1) \cdot x_k(t_1 + \tau) \tag{3.4.2}$$

For the general case where $\mu_x(t_1)$ and $R_{xx}(t_1, t_1 + \tau)$ vary as time t_1 varies, the random process is said to be *non-stationary.* If the mean and the autocorrelation function do not vary with time, the random process is said to be *weakly stationary.*

For the special case where all possible means and autocorrelation functions are time invariant, the random process is said to be *strongly stationary.*

In most cases it is possible to describe the properties of a stationary random process by computing time averages over specific sample functions in the ensemble. The mean value and the autocorrelation function of the kth sample function are given by:

$$\mu_x(k) = \lim_{N\to\infty} \frac{1}{T} \int_0^T x_k(t)dt \tag{3.4.3}$$

$$R_{xx}(t, k) = \lim_{N\to\infty} \frac{1}{T} \int_0^T x_k(t) \cdot x_k(t + \tau)dt \tag{3.4.4}$$

If the random process is stationary and $\mu_x(t)$ and $R_{xx}(t, k)$ do not differ when computed over different sample functions, the random process is said to be *ergodic*, therefore all properties of ergodic random processes can be determined by performing time averages over a single sample function

3.4.1 Input/Output Relations

Input/output cases of common interest can usually be considered as combinations of one or more of the following linear system models:

1. Single-input/single-output model

2. Single-input/multiple-output model
3. Multiple-input/single-output model
4. Multiple-input/multiple-output model

As previously introduced, Operational Modal Analysis uses MIMO-type models. MIMO models are an extension of simpler cases. The MIMO problem can be broken down into MISO problems and solved using MISO techniques: this allows a better understanding of the physical relationship between each input and output. However, since in OMA it is not possible to distinguish the various inputs, the MIMO model is solved using matrix techniques.
Let X be a column vector representing the Fourier transforms of the q input records $X_i = X_i(f)$, $i = 1, 2, \cdots, q$, and Y be a column vector representing the Fourier transform of the q output records $Y_k = Y_k(f)$, $k = 1, 2, \cdots, q$:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_q \end{bmatrix} \qquad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_q \end{bmatrix} \tag{3.4.5}$$

X^*,Y^*= complex conjugate (column) vectors of X, Y.
X',Y'= transpose (row) vectors of X, Y.

$$G_{xx} = \frac{2}{T} E\left\{X^* X'\right\} \tag{3.4.6}$$

$$G_{yy} = \frac{2}{T} E\left\{Y^* Y'\right\} \tag{3.4.7}$$

$$G_{xy} = \frac{2}{T} E\left\{X^* Y'\right\} \tag{3.4.8}$$

where G_{xx} is the *input spectral density* matrix, G_{yy} is the *output spectral density* matrix and G_{xy} is the *input/output cross-spectral density* matrix.
The basic matrix terms are defined as follows:

$$G_{ij} = \frac{2}{T} E\left[X_i^* X_j\right] \tag{3.4.9}$$

$$G_{y_i y_j} = \frac{2}{T} E\left[Y_i^* Y_j\right] \tag{3.4.10}$$

$$G_{x_i y_j} = \frac{2}{T} E\left[X_i^* Y_j\right] \tag{3.4.11}$$

$$\begin{aligned} G_{xx} = \frac{2}{T} E \left\{ \begin{bmatrix} X_1^* \\ X_2^* \\ \vdots \\ X_q^* \end{bmatrix} \begin{bmatrix} X_1 & X_2 & \cdots & X_q \end{bmatrix} \right\} \\ = \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1q} \\ G_{21} & G_{22} & \cdots & G_{2q} \\ \vdots & \vdots & & \vdots \\ G_{q1} & G_{q2} & \cdots & G_{qq} \end{bmatrix} \end{aligned} \tag{3.4.12}$$

$$\begin{aligned} G_{yy} = \frac{2}{T} E \left\{ \begin{bmatrix} Y_1^* \\ Y_2^* \\ \vdots \\ Y_q^* \end{bmatrix} \begin{bmatrix} Y_1 & Y_2 & \cdots & Y_q \end{bmatrix} \right\} \\ = \begin{bmatrix} G_{y_1y_1} & G_{y_1y_2} & \cdots & G_{y_1y_q} \\ G_{y_2y_1} & G_{y_2y_2} & \cdots & G_{y_2y_q} \\ \vdots & \vdots & & \vdots \\ G_{y_qy_1} & G_{y_qy_2} & \cdots & G_{y_qy_q} \end{bmatrix} \end{aligned} \tag{3.4.13}$$

G_{xx} and G_{yy} are Hermitian matrices, namely, $G_{ij} = G_{ji}^*$ for all i and j. For those Hermitian matrices, $G_{xx}^* = G_{xx}'$ and $G_{yy}^* = G_{yy}'$.
The system matrix between X and Y is defined by $H_{xy} = H_{xy}(f)$ where the input always precedes output. The matrix terms $H_{iy_k} = H_{x_iy_k}$. Then:

$$H_{xy} = \begin{bmatrix} H_{1y_1} & H_{1y_2} & \cdots & H_{1y_q} \\ H_{2y_1} & H_{2y_2} & \cdots & H_{2y_q} \\ \vdots & \vdots & & \vdots \\ H_{qy_1} & H_{qy_2} & \cdots & H_{qy_q} \end{bmatrix} \tag{3.4.14}$$

From this definition, it follows that:

$$Y = H_{xy}' X \tag{3.4.15}$$

where H'_{xy} is the transpose matrix to H_{xy}. Thus:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_q \end{bmatrix} = \begin{bmatrix} H_{1y_1} & H_{2y_1} & \cdots & H_{qy_1} \\ H_{1y_2} & H_{2y_2} & \cdots & H_{qy_2} \\ \vdots & \vdots & & \vdots \\ H_{1y_q} & H_{2y_q} & \cdots & H_{qy_q} \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_q \end{bmatrix} \tag{3.4.16}$$

This way any Y_k is related to the inputs X_i. Initially it is assumed that the number of outputs is the same as the number of inputs, and that all inverse inverse matrix operations can be performed.

$$Y = H'_{xy}X \tag{3.4.17}$$

$$Y' = (H'_{xy}X)' = X'H'_{xy} \tag{3.4.18}$$

$$X^*Y' = X^*X'H_{xy} \tag{3.4.19}$$

Taking expected values of both sides and multiplying by $\frac{2}{T}$ gives:

$$G_{xy} = G_{xx}H_{xy} \tag{3.4.20}$$

Multiplying both sides by G_{xx}^{-1}:

$$G_{xx}^{-1}G_{xy} = G_{xx}^{-1}(G_{xx}H_{xy}) \Rightarrow H_{xy} = G_{xx}^{-1}G_{xy} \tag{3.4.21}$$

Matrices G_{xx} and G_{yy} are then related by the FRF:

$$Y = H'_{xy}X \tag{3.4.22}$$

$$Y' = (H'_{xy}X)' = X'H_{xy} \tag{3.4.23}$$

$$Y^* = (H'_{xy}X)^* = H'^*_{xy}X^* \tag{3.4.24}$$

$$Y^*Y' = (H'^*_{xy}X^*)(X'H_{xy}) \tag{3.4.25}$$

Taking expected values of both sides and multiplying by $\frac{2}{T}$ gives:

$$G_{yy} = H'^*_{xy}G_{xx}H_{xy} \tag{3.4.26}$$

This final result represents the starting point for the FDD method, which will be illustrated afterwards.

3.5 Data Acquisition and Processing

The measured random data that represent the response of the system, are actually the expression of a continuous physical phenomenon, where specific data acquisition and processing procedures are required before an analysis on the data can be accomplished. In this section the focus will be put on the procedures required for data acquisition and digital signal processing.
Digital processing of the data is a very important step in structural testing. In order to determine modal parameters, the measured input (excitation) and response data must be processed and put into a form that is compatible with the test and modal parameter estimation methods [1]. The conversion of the data from the time domain into the frequency and Laplace domains is important both in the measurement process and subsequently in the parameter estimation process.

3.5.1 Data Acquisition

A data acquisition system usually involves a transducer with signal conditioning, transmission of the conditioned signal to an analog-to-digital converter (ADC), and a calibration of the data acquisition system (*standardization*) to establish the relationship between the physical phenomena being measured arid the conditioned signal transmitted to the analog-to-digital converter. Each element of the data acquisition system must be selected to provide the frequency range and dynamic range needed for the final engineering application of the data.

$$\begin{tikzcd}\boxed{\text{Transducer}} \arrow[r] & \boxed{\begin{array}{c}\text{Signal}\\\text{conditioning}\end{array}} \arrow[r] & \boxed{\text{Transmission}} \arrow[r] & \boxed{\text{Calibration}}\end{tikzcd}$$

The **transducer** is the primary element in a data acquisition system. Any device that translates power from one form to another is, in general terms, a transducer. In structural identification context, this means the translation of a physical phenomenon (i.e. a mechanical quantity, as the acceleration) into an electrical signal, in a quantity proportional to the magnitude of the phenomenon.

The generated electrical quantity is then converted by a **signal conditioner** into a voltage signal with a low source impedance (generally less than 100Ω) and the desired magnitude and frequency range for transmission to the remainder of the data acquisition system.
Piezoelectric or strain-sensitive materials are employed for sensing elements for force, pressure, and motion transducers. A piezoelectric material produces an

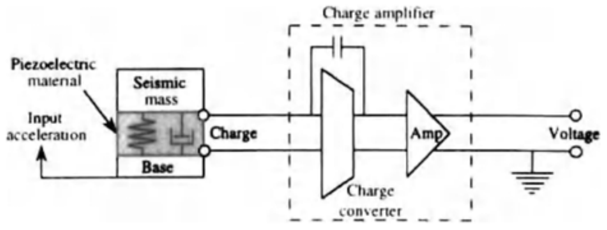


Figure 3.6: Schematic diagrm of piezoelectric accelerometer.

electrical charge when it is deformed. In case of a piezoelectric accelerometer, the mechanical conversion is achieved through a seismic mass supported by a flexible piezoelectric material, where an input acceleration at the base of the accelerometer is converted to a relative displacement of the mass.
When transducers are obtained from commercial sources, supporting literature is normally provided that specifies the limitations on their use.
The signal data from the signal conditioner is then calibrated and transferred to the remainder of the data acquisition system. During the **calibration** process the *dynamic range* of the system is determined. The dynamic range of the system is defined as the ratio of the maximum to minimum data values the system can acquire without significant distortion and is commonly assessed in terms of a maximum allowable *signal-to-noise ratio* (SNR) defined as:

$$SNR = \frac{\psi_s^2}{\psi_n^2} \tag{3.5.1}$$

or in decibels (dB) as:

$$(S/N) = 10 \cdot log_{10} \frac{\psi_s^2}{\psi_n^2} \tag{3.5.2}$$

where ψ_s is the maximum *rms* value of the signal that can be acquired without significant distortion and ψ_n is the rms value of the data acquisition system noise floor (since the system noise floor is commonly assumed to have a zero mean value, $\psi_n = \sigma_n$).

3.5.2 Data conversion

The conditioned signal must be converted into a digital format for it to be processable. This operation is accomplished by an *analog-to-digital converter* (ADC). The ADC is a device that translates a continuous analog signal, which represents an uncountable set, into a series of discrete values (a time series) which represents a countable set.

$\rightarrow \boxed{\text{Analog-to-digital conversion}} \rightarrow \boxed{\text{Sampling errors}} \rightarrow \boxed{\text{Storage}}$

The most common type of ADC in current use is that referred to as the sigma-delta ($\Sigma\Delta$) converter, which is schematically illustrated in the following diagram: Basically, the $\Sigma\Delta$ ADCs consist of an oversampling modulator followed

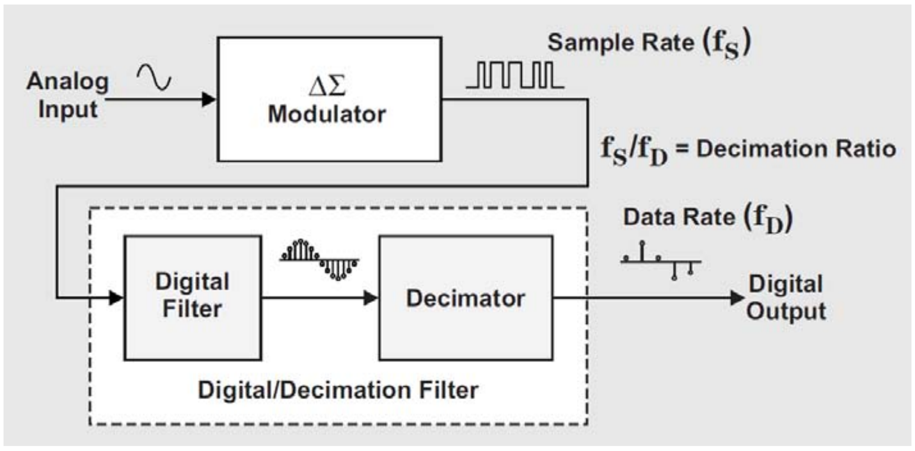


Figure 3.7: Schematic diagram of a sigma-delta converter.

by a digital/decimation filter that together produce a high resolution data-stream output [5]. The rudimentary $\Sigma\Delta$ converter is a 1-bit sampling system. An analog signal applied to the input of the converter needs to be relatively slow so that the converter can sample it multiple times, a technique known as *oversampling*. The sampling rate is hundreds of times faster than the digital results at the output ports. Each sample is accumulated over time and *averaged* with the other input-signal samples through the digital/decimation filter. While most converters have one sample rate, the $\Sigma\Delta$ converter has two, the input sampling rate f_s and the output data rate f_D.

The $\Sigma\Delta$ modulator is the heart of the $\Sigma\Delta$ ADC. It is responsible for digitizing the time-varying analog input signal and reducing noise at lower frequencies. Low-frequency noise is pushed up to higher frequencies (outside of the

band of interest) by the *noise shaping function* of the integrator module.
The $\Sigma\Delta$ modulator acquires many samples of the input signal to produce a stream of 1-bit codes. The system clock implements the sampling speed, f_s, in conjunction with the modulator's 1-bit comparator. The ratio of the number of ones and zeros represents the input analog voltage.
The digital-filter function implements a low-pass filter by first sampling the modulator stream of the 1-bit code. The digital/decimation filter throws away the high frequency noise that was shaped by the modulator stge and reduces the data-output rate of the device to a usable frequency. The output-data is constituted by frequencies from 0 to f_D in the signal band.
The process of oversampling followed by low-pass digital filtering and decimation can be interpreted as increasing the effective resolution of the digital output by suppressing the spectral density of the digital noise in the output.
The use of an ADC requires some important general considerations:

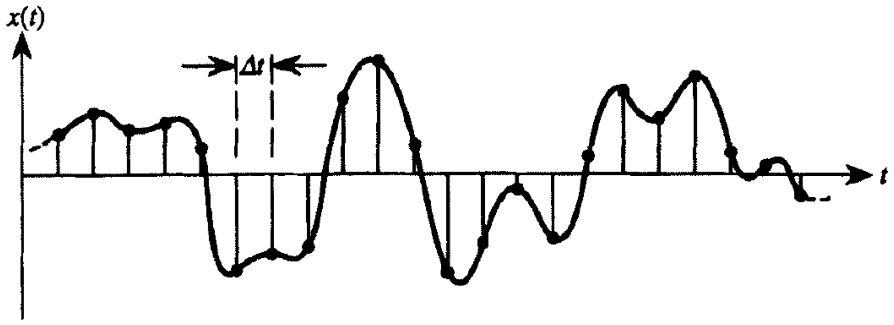


Figure 3.8: Sampling of analog signal

1. *Format.* The output of the ADC consists of the natural binary output of the converter. When the ADC is integrated into a general structural identification equipment it might be necessary to convert the output data in ASCII format in order to be able to process them.

2. *Resolution.* There is a round-off error introduced by the conversion, due to the size of each digital word that define the magnitude of the input signal.

3. *Sampling Interval.* In most cases, the ADC samples the input analog signal with a fixed sampling interval Δt.

4. *Sampling rate.* At least two sample values per cycle are required to define the highest frequency in an analog signal.

3.5.3 Sampling Theorems for Random Records

A sample random time history record $x(t)$ from a random process $\{x_k(t)\}$ is considered and exists only for the time interval from 0 to T seconds, and is zero elsewhere. Its Fourier transform can be expressed as:

$$X(\omega) = \int_0^T x(t)e^{-j2\pi\omega t}dt \tag{3.5.3}$$

If $x(t)$ is continually repeated, a periodic time function with a period of T seconds is obtained and the fundamental frequency increments is $f = 1/T$. By applying a Fourier series expansion, the $x(t)$ can be re-written as:

$$x(t) = \sum_{-\infty}^{\infty} A_n e^{j2\pi nt/T} \tag{3.5.4}$$

where A_n is computed from $x(t)$ as:

$$A_n = \frac{1}{T}\int_0^T x(t)e^{j2\pi nt/T}dt \tag{3.5.5}$$

Thus, $X(n/T)$ determines A_n and, therefore, $x(t)$ at all t:

$$X\left(\frac{n}{T}\right) = \int_0^T x(t)e^{j2\pi nt/T}dt = TA_n \tag{3.5.6}$$

This result determines $X(\omega)$ for all ω, and it is the *sampling theorem in the frequency domain.* The fundamental frequency increment $1/T$ is called a *Nyquist co-interval.*

In a similar way, a Fourier transform $X(\omega)$ of some sample random time history record $x(t)$ is considered. $X(\omega)$ exists only over a frequency interval from $-B$ to B H and is zero at all other frequencies. The actual realizable frequency band ranges from 0 to B Hz. The inverse Fourier transform leads to:

$$x(t) = \int_{-B}^{B} X(\omega)e^{j2\pi\omega t}d\omega \tag{3.5.7}$$

$X(\omega)$ is considered to be repeated in frequency in order to obtain a periodic frequency function with a period of $2B$ Hz, and the fundamental time increment is $t = 1/(2B)$. The Fourier transform $X(\omega)$ can be written as a Fourier series:

$$X(\omega) = \sum_{-\infty}^{\infty} C_n e^{-j\pi n\omega/B} \tag{3.5.8}$$

where C_n can be computed from $X(\omega)$ as:

$$C_n = \frac{1}{2B}\int_{-B}^{B} X(\omega) e^{-j\pi n\omega/B} d\omega \tag{3.5.9}$$

Thus $x\left[\frac{n}{2B}\right])$ determines C_n and, therefore, $X(\omega)$ at all ω.

$$x\left[\frac{n}{2B}\right]) = \int_{-B}^{B} X(\omega) e^{-j\pi n\omega/B} d\omega = 2BC_n \tag{3.5.10}$$

Thus, this determines $x(t)$ for all t:

$$x(t) = \sum_{-\infty}^{\infty} x\left[n/(2B)\right] \frac{sin\pi(2Bt - n)}{\pi(2Bt - n)} \tag{3.5.11}$$

The last equation shows how $x(t)$ is reconstructed from the sample values taken $1/(2B)$ seconds apart. This result is the *sampling theorem in the time domain* and the fundamental time increment $1/(2B)$ is called a Nyquist interval.

By sampling $X(\omega)$ at Nyquist co-interval points $1/T$ apart on the frequency scale from $-B$ to B, the number of discrete samples required to describe $x(t)$ is:

$$N = \frac{2B}{1/T} = 2BT \tag{3.5.12}$$

Whilst by sampling $x(t)$ at Nyquist interval points $1/2B$ apart on the timescale from 0 to T it follows that:

$$N = \frac{T}{1/2B} = 2BT \tag{3.5.13}$$

Sampling rates and aliasing errors

The sampling of analog signals for digital data analysis is usually performed at equally spaced time intervals. It is necessary to determine the appropriate sampling interval Δt. As discussed earlier, the minimum number of discrete samples required to describe the analog signal is given by $N = 2BT$. Therefore the maximum sampling interval is:

$$\Delta t = \frac{1}{2B} \tag{3.5.14}$$

Sampling at more closely spaced than $\frac{1}{2B}$ points will yield correlated and redundant sample values, whilst sampling at points further apart than $\frac{1}{2B}$ will lead to confusion between the low and high frequency components in the original data. This latter problem is called *aliasing* and is a source of error for the

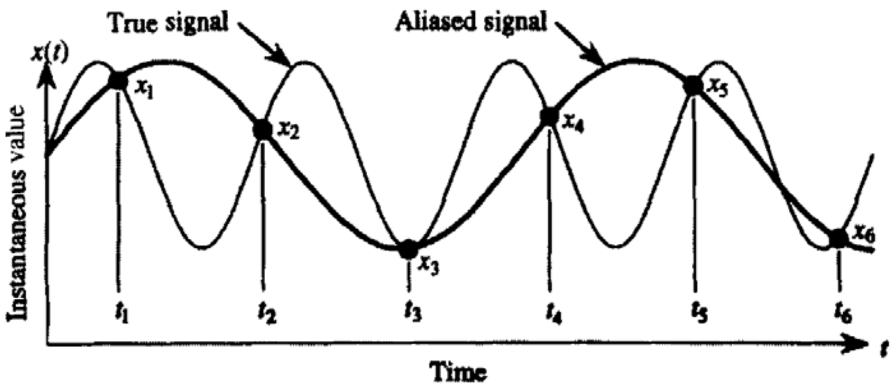


Figure 3.9: Frequency aliasing due to an inadequate digital sampling rate.

processing of the data after the analog-digital conversion. The presence of high frequencies in the original signal could be misinterpreted in the discretisation process, and those frequencies will appear as low frequencies.
If the sampling frequency is f_s, then the signal of frequency ω and signal of frequency $f_s - \omega$ are indistinguishable after discretization, and this causes distortion of the measured spectra using DFT. The highest frequency that can be defined by a sampling rate of $1/\Delta t$ is $1/(2\Delta t)$ Hz. Frequencies in the original data above $1/(2\Delta t)$ Hz will appear below $1/(2\Delta t)$ Hz and be confused with the data in this lower frequency range.

$$f_A = \frac{1}{2\Delta t} \tag{3.5.15}$$

is called the $Nyquist frequency$.
The signal appears in DFT as a distorted form: towards the upper end of the applicable frequency range $(0 - f_A)$ the distortion is due to the fact that the

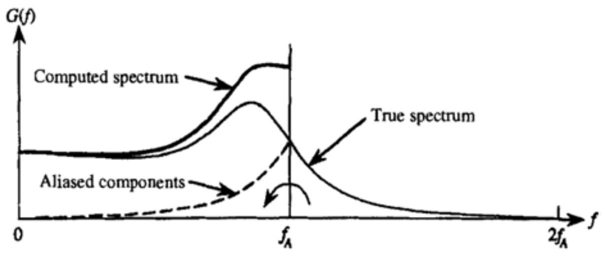


Figure 3.10: Aliasing error in the computation of an autospectral density function.

portion of the signal with frequency components above f_A will be reflected in the range $0 - f_A$. The problem is solved using an *anti-aliasing* filter which subjects the original time signal to a low-pass, sharp cut-off filter. Actually, because the filters used are not perfect and have a finite cut-off rate, the spectral measurements in a frequency range approaching the Nyquist frequency must be rejected. Typically, the range from $0,8 \cdot f_A$ to f_A is rejected.

Leakage

Leakage is a direct consequence of the need to take only a finite length of time history coupled with the assumption of periodicity. We consider a sinusoidal signal. If the signal is perfectly periodic in the time window T, the resulting spectrum will be simply a single line at the frequency of the sine wave. If,

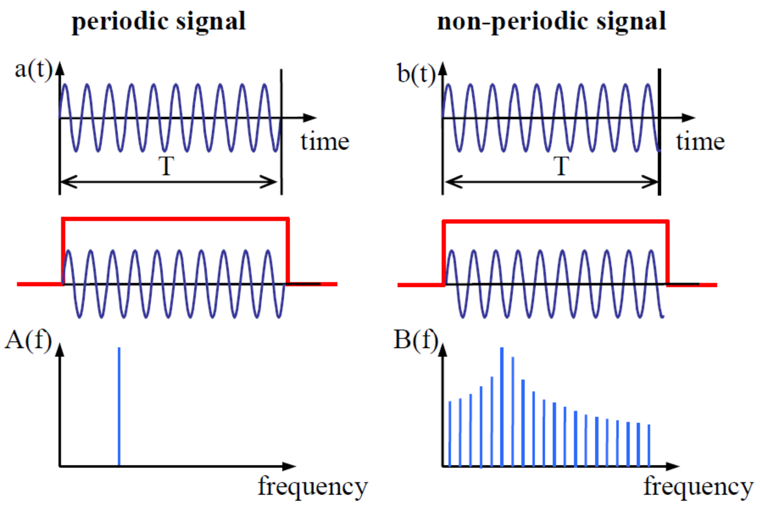


Figure 3.11: Leakage error.

however, the periodicity assumption is not satisfied and there is a discontinuity at the end of the sample, the spectrum will not indicate the single frequency which the original time signal possessed, and this frequency may not even prevail in the spectral lines, because the energy *leaks* into a number of spectral lines close to the true frequency and the spectrum is spread over several lines. Leakage is more relevant for low frequency signals and is a serious problem in may applications of digital signal processing, including FRF measurements. There are several ways of minimizing its effects [7]:

- Changing the duration of the measurement sample length to match any underlying periodicity in the signal. This is possible only if the signal is periodic and its period can be determined.
- Increasing the duration of measurement time T, so the separation between the spectral lines is finer.
- *Windowing*: consists on modifying the signal sample obtained in such a way as to reduce the severity of the leakage effect. This process is

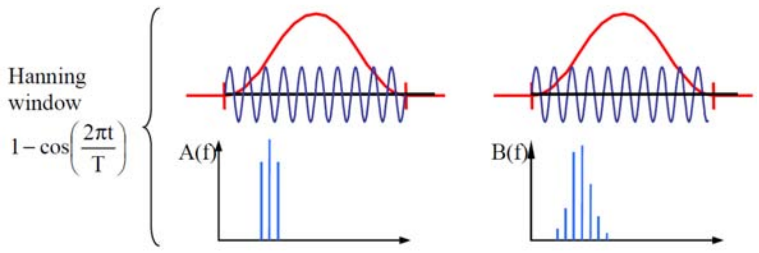


Figure 3.12: Influence of Hanning window to Fourier transform of a signal.

the most widely employed in modal testing. A prescribed profile $w(t)$ is imposed on the time signal prior to performing Fourier transform. The analyzed signal is then product of original signal and window profile as shown in the figure.

3.6 Operational Modal Analysis Techniques

In this thesis system for structural identification and monitoring is discussed, which implements two methods commonly used to perform an Operational

Modal Analysis: *Frequency Domain Decomposition* and *Stochastic Subspace Identification.* Being modal models, their outputs consist in modal parameter of the system, while being Operational models their input consists in the measured response of the system (structure's output). So input and output for these methods are the same. How these two methods work, though, is completely different, and their results are heavily influenced by the mathematical steps involved. Therefore it is necessary to understand the theory behind these two methods, which will be presented in the next section, to be able to handle them correctly.

3.6.1 Frequency Domain Decomposition

Frequency domain OMA methods are based on the previously introduced formula of input and output power spectral density (PSD) relationship for stochastic process [41]:

$$G_{yy}(j\omega) = H(j\omega)^* \cdot G_{xx}(j\omega) \cdot H(j\omega)^T \tag{3.6.1}$$

where G_{xx} and G_{yy} are input and output PSD matrices, $H(j\omega)$ is the FRF matrix which can be expressed as a partial fraction form via poles λ_r and residues R_r:

$$H(j\omega) = \sum_{r=1}^{N} \left(\frac{R_r}{j\omega - \lambda_r} + \frac{R_r^*}{j\omega - \lambda_r^*} \right) \tag{3.6.2}$$

where $R_r = \phi_r \gamma_r^T, \phi_r, \gamma_r$ are mode shape and modal participation vector respectively. When all output measurements are taken as references, then $H(j\omega)$ is a square matrix and $\gamma_r = \phi_r$.

In OMA, the input is given by the natural excitations acting on the system and it can be considered as white noise. Therefore $G_{xx}(j\omega)$ equals to constant, and the modal decomposition of the output PSD matrix $G_{yy}(j\omega)$ can be derived as modal decomposition:

$$G_{yy}(j\omega) = \sum_{r=1}^{N} \left(\frac{A_r}{j\omega - \lambda_r} + \frac{A_r^H}{-j\omega - \lambda_r^*} + \frac{A_r^*}{j\omega - \lambda_r^*} + \frac{A_r^T}{-j\omega - \lambda_r} \right) \tag{3.6.3}$$

where the rth pole $\lambda_r = -\sigma_r + j\omega_{dr}$, corresponding to the rth residue $A_r \approx d_r \phi_r^* \phi_r^T$, $d_r = \gamma_r^H G_{xx} \gamma_r$, is a real scalar for white noise excitation. In the vicinity of a modal frequency, the PSD can be approximated:

$$G_{yy}^{T}(j\omega)|_{\omega\to\omega_r} \approx \phi_r \frac{2d_r}{j\omega - \lambda_r}\phi_r^H = \alpha_r \phi_r \phi_r^H \tag{3.6.4}$$

where α_r is a scalar constant.

Peak Picking

A classical Frequency Domain approach is the Peak Picking technique (PP), in which the natural frequency are directly obtained from the choice of peaks in PSD graph. If the modes are well separated, this technique will lead to acceptable estimates. However Peak Picking technique presents a low accuracy especially for complex structures due to the dependency of the result to the resolution of PSD spectrum, extraction of operational deflection shape instead of the system natural mode shape, low accuracy in calculation of damping ratio, and impossibility of its usage for the systems with close modes.

Frequency Domain Decomposition

At a certain frequency ω, only a limited number of modes will contribute significantly, typically one or two modes. Let this set of modes be denote by $Sub(\omega)$. Thus, in the case of a lightly damped structure, the response spectral density can always be written as:

$$G_{yy}(j\omega) = \sum_{k\in Sub(\omega)} \frac{\alpha_r \phi_r \phi_r^T}{j\omega - \lambda_r} + \frac{\alpha_r^* \phi_r^* \phi_r^H}{j\omega - \lambda_r^*} \tag{3.6.5}$$

PSD $\hat{G}_{yy}(j\omega)$, known at discrete frequencies $\omega = \omega_i$, is then decomposed by taking the *Singular Value Decomposition* (SVD) of the matrix:

$$\hat{G}_{yy}(j\omega_i) = U_i S_i U_i^H \tag{3.6.6}$$

where the matrix $U_i = [u_{i1}, u_{i1}, ..., u_{im}]$ is a unitary matrix holding the singular vectors u_{ij}, and S_i is a diagonal matrix holding the scalar singular values s_{ij}. When only kth mode dominates at the modal frequency ω_r, the PSD matrix approximates to a rank one matrix as:

$$\hat{G}_{yy}(j\omega_i)|_{\omega_i\to\omega_r} = s_i u_{i1} u_{i1}^H \tag{3.6.7}$$

Compared to the previous PSD approximation formula, it is seen that the first singular vector at kth resonance is an estimate of the kth mode shape: $\hat{\phi}_k = u_{k1}$ and the corresponding singular value is the auto power spectral density function of the corresponding single degree of freedom system. In case there are more modes repeated at the same frequency, the rank of the matrix will be equal to the number of multiplicity of the modes.

About Singular Value Decomposition Singular Value Decomposition has found many applications in modal analysis. The Singular Value Decomposition of a complex matrix A of dimensions $n \times m$ is given by:

$$A = U\Sigma V^H \tag{3.6.8}$$

where U and V are unitary matrices, Σ is a diagonal matrix that contains the real singular values:

$$\Sigma = diag(s_1, ..., s_r) \tag{3.6.9}$$

$$r = min(m, n) \tag{3.6.10}$$

The SVD can be computed by using some existing mathematical software. Analytically, the eigenvectors of AA^T constitute U and the eigenvalues of it constitute $\Sigma^T\Sigma$. Likewise, the eigenvectors of A^TA constitute V and the eigenvalues of it (the same as the eigenvalues of AA^T) constitute $\Sigma\Sigma^T$. Therefore, SVD is closely linked to the eigenvalue decomposition.
SVD reveals useful information about A [22]. For instance, the number of non-zero singular values (therefore the rank of Σ) coincides withe the rank of A. Once the rank is known, the first k columns of u form an orthogonal basis for the column space of A. In numerical analysis, zero singular values can become small quantities due to numerical errors, measurement errors, noise and ill-conditioning of the matrix.
The superscript H on the matrix V denotes a Hermitian transformation (transpose and complex conjugate). In the case of real valued matrices, the V matrix is only transposed. The S_i elements in the matrix Σ are called the singular values, and their corresponding singular vectors are contained in the matrices U and V.

The SVD is performed for each data-set at each frequency. The PSD matrix is then approximated to the following expression after SVD:

$$G_{yy}(\omega_i) = \Phi\Sigma\Phi^H \tag{3.6.11}$$

with $\Phi^H\Phi = I$.

Enhanced Frequency Domain Decomposition

The second generation of FDD, which is called as Enhanced FDD technique allows the extraction of the resonance frequency and the damping of a particular mode by computing the auto-correlation function [20]. SDOF auto spectral densities are identified using the modal assurance criterion (MAC) around a peak of resonance, and are taken back to the time domain using the Inverse Discrete Fourier Transform (IDFT). A more accurate estimation of the resonance frequency is obtained by determining the zero crossing times, and the damping ratio is calculated using the logarithmic decrement of the normalized auto-correlation function.

Identification of the SDOF auto spectral densities As mentioned earlier, near a peak corresponding to the kth mode in the spectrum, if only the kth mode is dominating, the first singular vector $\hat{\Phi}_k = u_{k1}$ is an estimate of the mode shape and the corresponding singular value is the auto power spectral density function of the corresponding SDOF system. The singular value data near the peak with corresponding singular vector having enough high MAC value are transferred back to time domain via inverse FFT. The spectral bell is therefore composed using the MAC criterion:

$$MAC = \frac{(u_{1k}u_{1i})^2}{u_{1k}^2 u_{1i}^2} \tag{3.6.12}$$

where u_{1i} are the singular vectors around the peak and u_{1k} is the singular vector at the peak frequency.

The singular values for which $MAC < \ \Omega$ (the threshold MAC value) are not part of the SDOF auto spectral density function so they are set to zero.

Therefore we consider only the singular values around the peak for which the condition $MAC > \ Omega$ is satisfied.
From the fully or partially identified SDOF auto spectral density function, the natural frequency and the damping are obtained by taking the spectral density function back to time domain by inverse FFT.

Calculation of modal parameters From the free decay time domain function, which is also the auto correlation function of the SDOF system, the natural frequency and the damping is found by estimating crossing times and logarithmic decrement. First all extremes r_k, both peaks and valleys, on the correlation function are found. The logarithmic decrement δ is then given by:

$$\delta = \frac{2}{k} ln \left(\frac{r_0}{|r_k|} \right) \tag{3.6.13}$$

where r_0 is the initial value of the correlation function and r_k is the kth extreme. Thus, the logarithmic decrement and the initial value of the correlation function can be found by linear regression on $k\delta$ and $2ln(|r_k|)$. The estimation is performed by applying a linear fit to the part of the curve being close to a straight line, therefore the damping ratio is given by the known formula, used for under-damped systems:

$$\zeta = \frac{\delta}{\sqrt{\delta^2 + 4\pi^2}} \tag{3.6.14}$$

The frequency is found by making a linear regression on the crossing times and the times corresponding to the extremes. The damped natural frequency f_d and the undamped natural frequency f is related by:

$$f = \frac{f_d}{\sqrt{1 - \zeta^2}} \tag{3.6.15}$$

Harmonic Excitations in OMA

The algorithms used in Operational Modal Analysis assume stochastic input forces, and this is usually verified for civil engineering structures, which are mainly loaded by ambient forces like wind, waves, traffic or seismic microtremors. The loading forces of many mechanical structures are, however, often

a combination of harmonic components (deterministic signals) originating from the rotating and reciprocating parts and broadband excitation originating from either self-generated vibrations from, for example bearings and combustions or from ambient excitations like air turbulence and road vibrations [25].
As the input forces to the structure are not measured in OMA, the influence of the harmonic components on the response must be identified and removed. While SSI methods estimate both harmonic components and structural modes, in EFDD the identified SDoF function used for modal parameter estimation may be biased by the harmonic components and harmonic components outside the SDoF function might narrow the SDoF function and might impoverish the identification results.
Therefore the influence of the harmonic components must be eliminated from the SDoF functions before processing them with the modal parameter extraction process. Harmonic components cannot, in general, be removed by simple filtering as this would significantly change the natural frequency and modal damping of the structural modes.

Identification of Harmonic Components using EFDD An efficient way to discriminate harmonics is by the statistical characteristics of the response in a narrow frequency band around a harmonic peak [10]. The statistical properties of a harmonic are in fact different from the properties of a stochastic response. The stochastic distribution of a modal response will be close to Gaussian, due to the central limit theorem and the fact that in practice a structure is loaded by many stochastically independent forces:

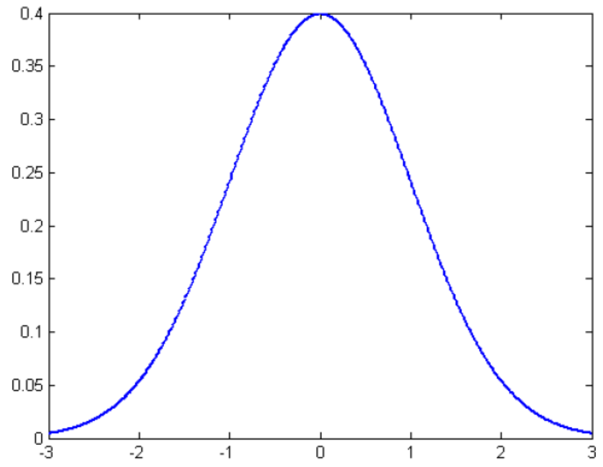


Figure 3.13: Normalized PDF of a pure structural mode.

Further, the distribution of a harmonic is different from Gaussian since it has two distinctive peaks where the distribution gees to infinity [6].

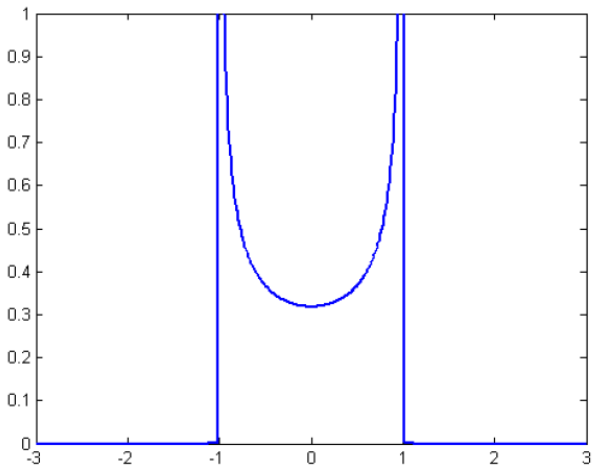


Figure 3.14: Normalized PDF of a pure harmonic component.

In [25] it is shown how to use the *kurtosis* to discriminate between modal peaks and harmonic peaks.

Kurtosis The kurtosis γ of a stochastic variable x provides a way of expressing how peaked or how flat the PDF of x is. The kurtosis is defined as the fourth central moment of the stochastic variable x normalized with respect to the standard deviation σ as follows:

$$\gamma(x|\mu,\sigma) = \frac{E\left\{(x-\mu)^4\right\}}{\sigma^4} \tag{3.6.16}$$

the number 3 is then subtracted from γ to give a kurtosis of zero when x is Gaussian:

$$\gamma^*(x|\mu,\sigma) = \gamma(x|\mu,\sigma) - 3 \tag{3.6.17}$$

Therefore, for a structural mode the kurtosis will be equal to zero, since the associated PDF will be normally distributed, while an harmonic component is associated with a positive kurtosis.

3.6.2 Stochastic Subspace Identification

The data driven Stochastic Subspace Identification (SSI) is considered to be the most powerful class of the known identification techniques for natural input modal analysis in the time domain [9]. SSI methods have been proven efficient for the identification of linear time-invariant systems (LTI), fitting a linear model to output-only measurements taken from a system. During the last two decades, subspace methods found a special interest in mechanical, civil and aeronautical engineering for the identification of modal parameters of vibrating structures, as they are computationally efficient methods and can deal with realistic excitation assumptions.

The discrete time formulation

We consider the stochastic response from a system as function of time:

$$y(t) = \begin{Bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_M(t) \end{Bmatrix} \tag{3.6.18}$$

The system can be considered in classical formulation as a multi-degree-of-freedom structural system:

$$M\ddot{y}(t) + D\dot{y}(t) + Ky(t) = f(t) \tag{3.6.19}$$

Where M, D, K is the mass damping and stiffness matrix and where $f(t)$ is the loading vector. In order to switch to a discrete time formulation, a state space model is introduced, as the one previously shown. A new variable, $x(t)$, is introduced:

$$x(t) = \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix} \tag{3.6.20}$$

With the state-space formulation, the original 2^{nd} order system equation simplifies to a first order equation:

$$\dot{x}(t) = A_C x(t) + Bf(t) \tag{3.6.21}$$

$$y(t) = Cx(t) \tag{3.6.22}$$

Where the system matrix A_C in continuous time and the load matrix B is given by:

$$A_C = \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}D \end{pmatrix} \tag{3.6.23}$$

$$B = \begin{pmatrix} 0 \\ -M^{-1} \end{pmatrix} \tag{3.6.24}$$

This formulation allows to define a direct solution:

$$x(t) = e^{A_c t} x(0) + \int_0^t e^{A_c(t-\tau)} Bf(\tau) d\tau \tag{3.6.25}$$

The first term is the solution to the homogeneous equation and the last term is the particular solution.
To take the solution to discrete time, all variables are sampled like $y_k = y(k\Delta t)$ and thus the solution of the homogeneous equation becomes:

$$x_k = e^{A_C k\Delta t} x_0 = A_d^k x_0 \tag{3.6.26}$$

$$A_d = e^{A_C \Delta t} \tag{3.6.27}$$

$$y_k = CA_d^k x_0 \tag{3.6.28}$$

The Block Hankel Matrix

In discrete time, the system response is normally represented by the data matrix:

$$Y = [y_1, y_2 \cdots y_N] \tag{3.6.29}$$

where N is the number of data points. Let $Y_{1:N-k}$ be the data matrix where we have removed the last k data points, and $Y_{k:N}$ the data matrix where we have removed the first k data points. Then an unbiased estimate of the correlation matrix at time lag k is given by:

$$\hat{R}_k = \frac{1}{N-k} Y_{1:N-k} Y_{k:N}^T \tag{3.6.30}$$

The *Block Hankel* matrix Y_h defined in SSI is simply a gathering of a family of matrices that are created by shifting the data matrix.

$$Y_h = \begin{pmatrix} Y_{1:N-2s} \\ Y_{2:N-2s+1} \\ \vdots \\ Y_{2s:N} \end{pmatrix} = \begin{pmatrix} Y_{hp} \\ Y_{hf} \end{pmatrix} \tag{3.6.31}$$

The upper half of the matrix is called *the past* and is denoted as Y_{hp}, while the lower half of the matrix is called *the future* and is denoted as Y_{hf}. The total shift is $2s$ and is denoted *the number of block rows*. The number of the rows in the Blok Hankel matrix is $2sM$, the number of columns is $N - 2s$.

The Projection

Projection is defined as a conditional mean. In SSI, the projection of the future unto the past defines the matrix

$$O = E(Y_{hf} | Y_{hp}) \tag{3.6.32}$$

The conditional mean can, for Gaussian processes, be totally described by its covariances. Since the shifted data matrices also defines covariances the projection can be calculated directly as:

$$O = Y_{hf} Y_{hp}^T (Y_{hp} Y_{hp}^T)^{-1} Y_{hp} \tag{3.6.33}$$

where the last matrix in the product defines the conditions and the first four matrices introduce the covariances between hannels at different time lags. This form of conditional mean consist of free decays of the system given by different initial conditions specified by Y_{hp}. The matrix is $sM \cdot sM$ and any column in the matrix O is a stacked free decay of the system to a (so far unknown) set of initial conditions. Any column of O can be expressed by:

$$O_{col} = \Gamma_S x_0 \tag{3.6.34}$$

$$\Gamma_S = \begin{pmatrix} C \\ CA_d \\ CA_d^2 \\ \vdots \\ CA_d^{s-1} \end{pmatrix} \tag{3.6.35}$$

where Γ is called the *observability matrix*.

The Kalman States

The *Kalman states* are the initial conditions for all the columns in the matrix O, thus

$$O = \Gamma_s X_0 \tag{3.6.36}$$

where the matrix X_0 contains the Kalman states at time lag zero. If we remove one block row from O from the top, and one block row of Γ_s from the bottom, the Kalman state matrix at time lag *one* X_1 is obtained.

Since the matrix Γ_S is unknown, the states can be estimated using a Singular Value Decomposition (SVD) on the O matrix:

$$O = U \cdot S \cdot V^T \tag{3.6.37}$$

The estimate of the matrix Γ is then given by:

$$\hat{\Gamma} = US^{1/2} \Rightarrow \hat{X}_0 = S^{1/2}V^T \tag{3.6.38}$$

Estimating the system matrices

The system matrix A_d can be found from the estimate of the matrix Γ by removing one block from the top and one block from the bottom:

$$\hat{\Gamma}_{(2:s)}\hat{A}_d = \hat{\Gamma}_{(1:s-1)} \tag{3.6.39}$$

and thus, the system matrix $\hat{A}_d$ can be found by regression. The observation matrix C can be obtained by taking the first block of the observability matrix:

$$\hat{C} = \hat{\Gamma}_{1:1} \tag{3.6.40}$$

Modal Analysis

An eigenvalue decomposition of the system matrix $\hat{A}_d$ is performed:

$$\hat{A}_d = \Psi \left[\mu_i\right] \Psi^{-1} \tag{3.6.41}$$

The continuous time poles λ_i are found from the discrete time poles μ_i by:

$$\mu_i = e^{\lambda_i} \tag{3.6.42}$$

Leading to the formulas for the calculation of the modal parameters:

$$\begin{aligned} \lambda_i &= \tfrac{ln(\mu_i)}{\Delta T} \\ \omega_i &= |\lambda_i| \\ f_i &= \tfrac{\omega_i}{2\pi} \\ \zeta_i &= \tfrac{Re(\lambda_i}{|\lambda_i|} \end{aligned} \tag{3.6.43}$$

The mode shape matrix is found from:

$$\Phi = C\Psi \tag{3.6.44}$$

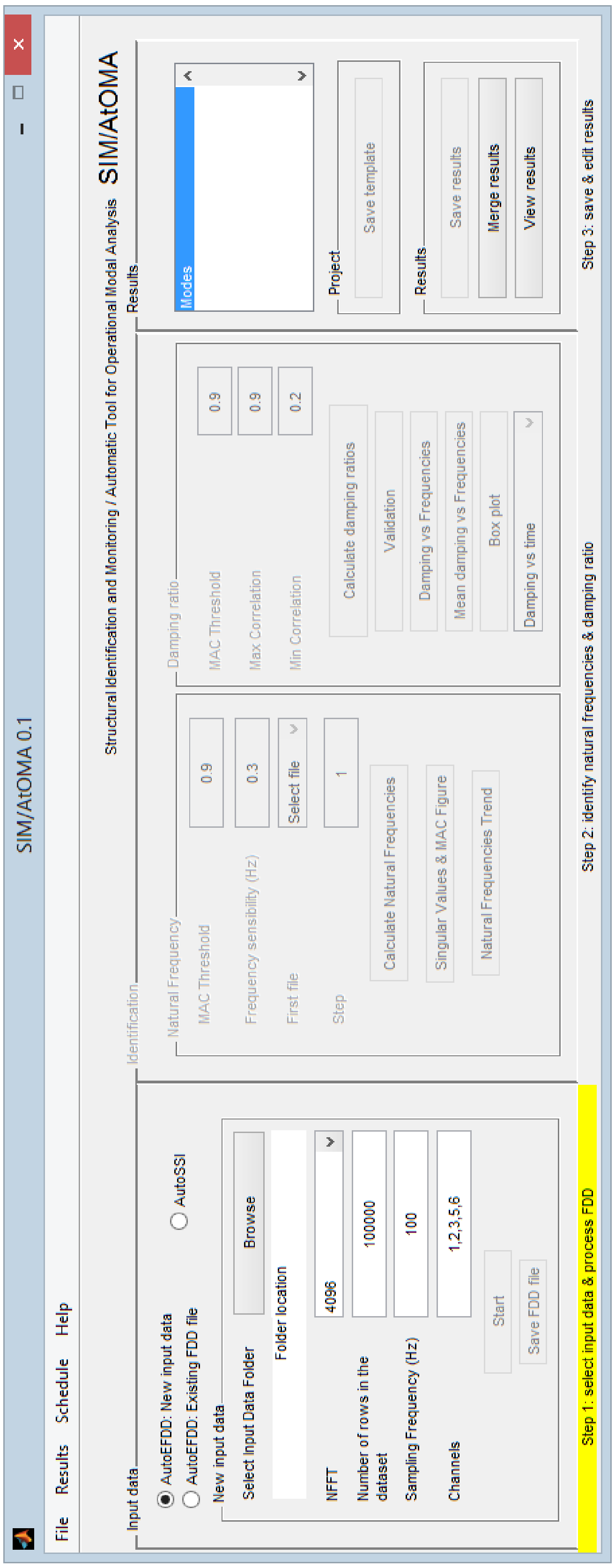
Chapter 4

SIM/AtOMA: User's Manual

4.1 Introduction

4.1.1 Welcome to SIM/AtOMA 0.1

Operational Modal Analysis **SIM/AtOMA** (*System Identification and Monitoring - Automatic tool for Operational Modal Analysis*) is a software tool that allows the user to perform structural dynamic identification and monitoring tasks, implementing two Operational Modal Analysis methods: the *Enhanced Frequency Domain Decomposition* and the *Stochastic Subspace Identification.* The identification procedure requires a consistent set of raw acceleration measurements performed on the structure and provides, for both methods, the structure's natural frequencies and damping ratios. It is also possible to configure **AtOMA** to perform automatically the analysis on new data using template files, in order to allow a continuous monitoring of the dynamic characteristics of the structure. The program disposes of a graphical user interface (GUI) (Figure 4.1), which makes it simple and user friendly.



SIM/AtOMA 0.1

File Results Schedule Help

Structural Identification and Monitoring / Automatic Tool for Operational Modal Analysis SIM/AtOMA

Input data

[✓ AutoEFDD: New input data]
AutoEFDD: Existing FDD file
AutoSSI

New input data

Select Input Data Folder	Browse
Folder location	
NFFT	4096
Number of rows in the dataset	100000
Sampling Frequency (Hz)	100
Channels	1,2,3,5,6

Start
Save FDD file

Identification

Natural Frequency

MAC Threshold	0.9
Frequency sensibility (Hz)	0.3
First file	Select file
Step	1

Calculate Natural Frequencies
Singular Values & MAC Figure
Natural Frequencies Trend

Damping ratio

MAC Threshold	0.9
Max Correlation	0.9
Min Correlation	0.2

Calculate damping ratios
Validation
Damping vs Frequencies
Mean damping vs Frequencies
Box plot
Damping vs time

Results

Modes

Project
Save template

Results
Save results
Merge results
View results

Step 1: select input data & process FDD
Step 2: identify natural frequencies & damping ratio
Step 3: save & edit results

Figure 4.1: AtOMA user interface.

4.2 Getting started

4.2.1 AtOMA: Description of the Graphical User Interface

AtOMA provides a friendly user interface that guides the user through each analysis step. It is composed of three major blocks:

- *Input data* panel: this is where the analysis starts. It is presented the choice of method to be used for the analysis (*AutoEFDD* or *AutoSSI*) and the parameters of the system configuration.
- *Identification*: in this panel the modal parameters for the AutoEFDD method are determined. In the first part the natural frequencies are identified, while the second part concerns the damping ratios.
- *Results*: this panel concerns the results managements. Here it is possible to save new results, to view a results file, to merge compatible results and to save a new template.

The menu bar has a number of global utility functions, some of which are already present in the user interface:

- *File*
  - New
  - New from template
  - Exit
- *Results*
  - View Results
  - Merge Results
- *Schedule*
  - New Schedule

  - Schedule Launcher

- *Help*
  - User Manual
  - About SIM/AtOMA

These functions will be discussed individually below.

4.3 AutoEFDD method

Identification *AutoEFDD* is an algorithm based on the Enhanced Frequency Domain Decomposition. As an Operational Modal Analysis method, it assumes that the input forces are stochastic in nature. The only inputs required by the program are the raw measures sets containing the accelerations of the structure.
First input data (the files containing the accelerations measurements) are selected and the corresponding characteristics are specified by the user. The Cross Power Spectral density function is computed for each file and then the *cpsd* matrices are decomposed using the Singular Value Decomposition. Subsequently the identification of the modal parameters is performed: after the user has set up the parameters, *natural frequencies* and *damping ratios* are calculated automatically. Finally, the user is able to save, merge and view the results of the analysis.
The results of the analysis consist on two tables: *Result_ Frequency* and *Result_ Damping*. The first column of each file contains the date on which the measurement was carried out. Each row contains, in ascending order, the natural frequencies calculated for each mode.
The flow chart in Figure 4.2 summarizes the steps required in order to conduct a new identification.

Monitoring The AutoEFDD method is also suitable for dynamic monitoring. As a frequency domain method, it is proven to be adequately fast and precise.

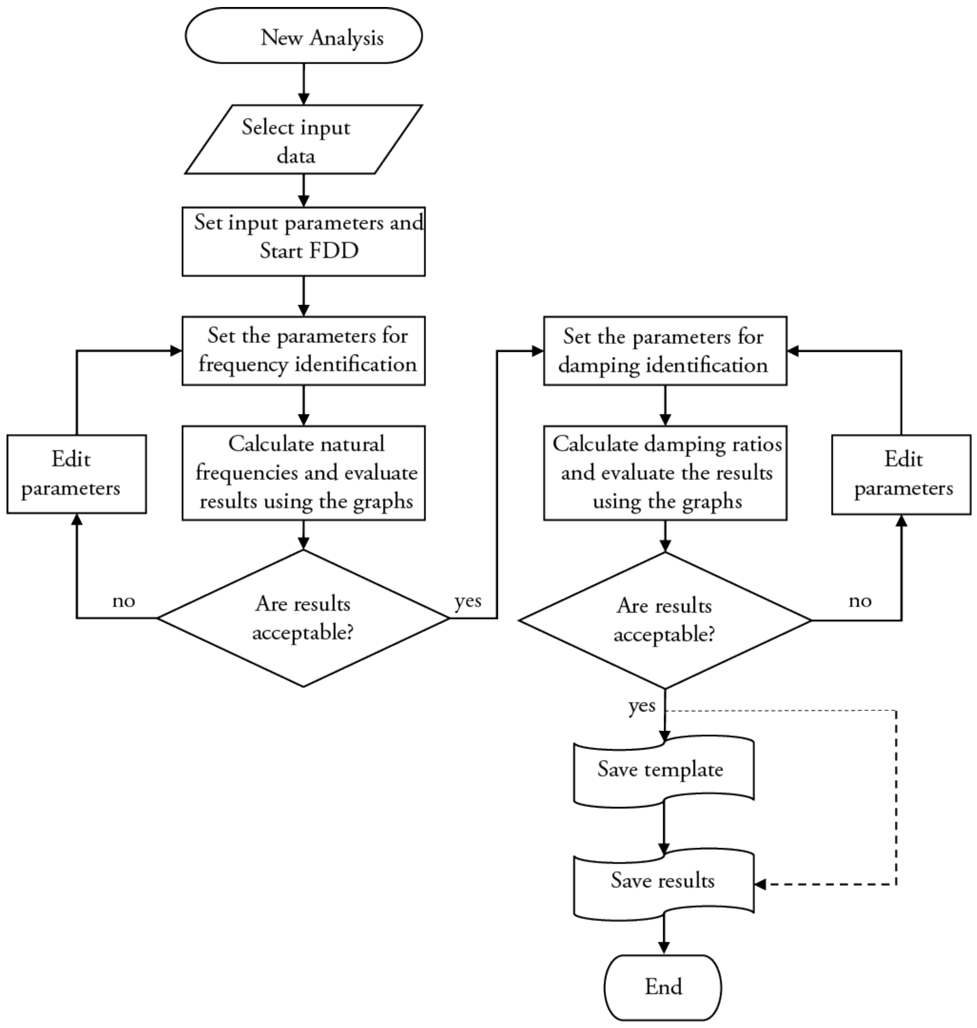

New Analysis
Select input data
Set input parameters and Start FDD
Set the parameters for frequency identification
Calculate natural frequencies and evaluate results using the graphs
Edit parameters
Are results acceptable?
no
yes
Set the parameters for damping identification
Calculate damping ratios and evaluate the results using the graphs
Edit parameters
Are results acceptable?
no
yes
Save template
Save results
End

Figure 4.2: New analysis flow chart.

Once an identification task on new data is performed, it is possible to save a *Template* file. Inside the template files are stored information about the analysis, including the natural frequencies found for the structure. The template file allows the user to perform a new, compatible, identification analysis on the system, loading only the new measurement files. The program identifies, if any, the natural frequencies calculated for the new files starting from those saved in the template, and calculates the corresponding damping ratio.
It is also possible to configure a system for automatic execution of the monitoring tasks on the basis of a template file, by using *schedule* functions.
In Figure 4.3 is shown the flow chart that summarizes the steps required to

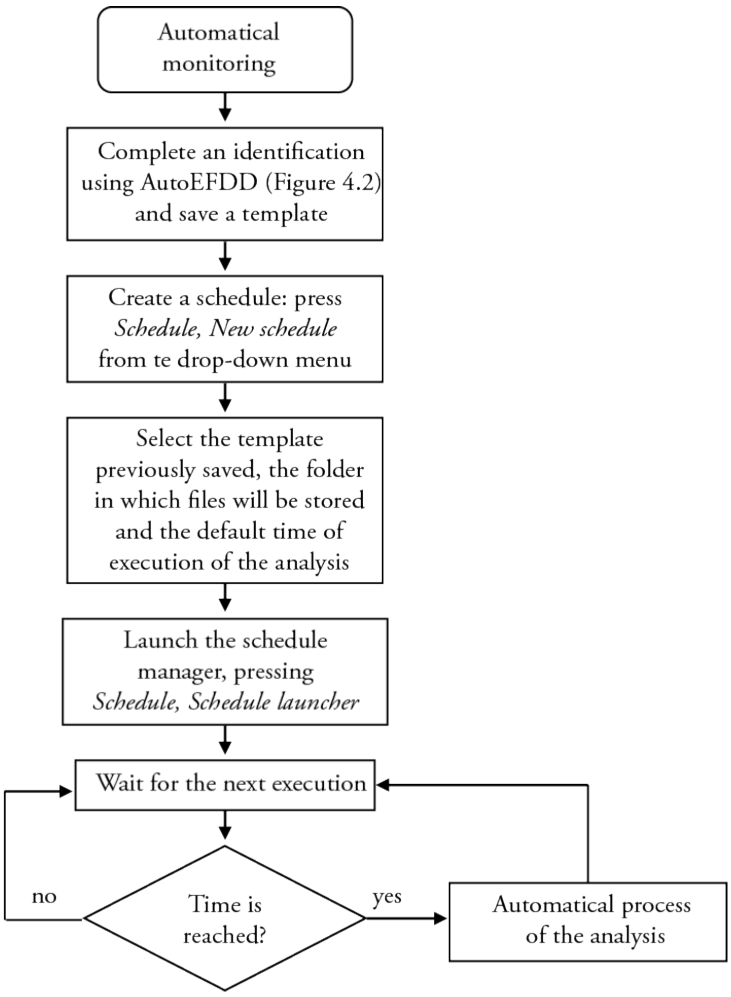

Automatical monitoring
Complete an identification using AutoEFDD (Figure 4.2) and save a template
Create a schedule: press *Schedule, New schedule* from te drop-down menu
Select the template previously saved, the folder in which files will be stored and the default time of execution of the analysis
Launch the schedule manager, pressing *Schedule, Schedule launcher*
Wait for the next execution
no
Time is reached?
yes
Automatical process of the analysis

Figure 4.3: Automatic monitoring flow chart.

perform an automatic monitoring.

4.3.1 Step 1: select input data and perform CPSD & SVD

To perform a new AutoEFDD analysis, select *File, New* from the menu. The user can choose whether to select new data, or using a partial FDD file previously saved by selecting the corresponding radio button.

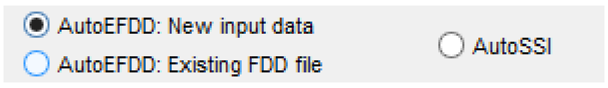


Figure 4.4: Input data type.

The second option may be useful if the user needs to perform an analysis on large amount of data at a later time.

New input data

This function is used to load the raw data files containing the acceleration measures on the structure. The requirements for the input files are:

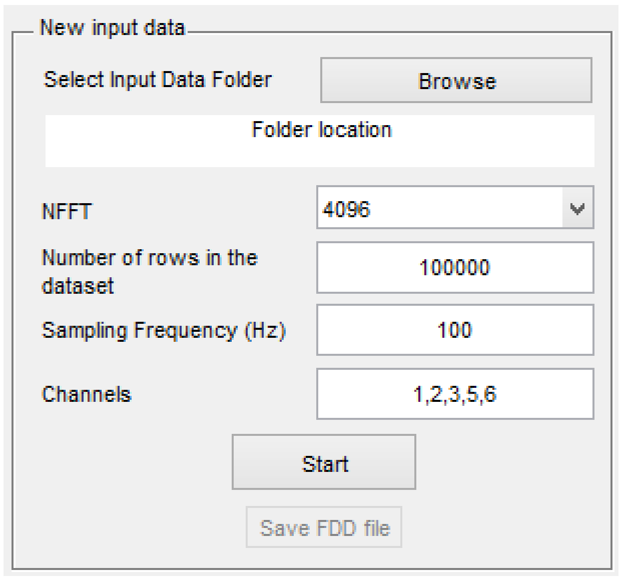
- The file format is **.txt*
- The file names must contain the suffix *’_ Fast’*
- In the file, after the header lines, the first column must contain the time (in seconds) while subsequent columns, each separated by tabs, must contain the accelerations for each channel. At each row corresponds an instant of measurement.

A compatible measurement file is formatted as follows:

```
t0    2013/04/01 13:03:07
dt    0,010000
time   ACC_ 1     ACC_ 2    ACC_ 3     ACC_ 4     ACC_ 5     ACC_ 6


0,000  -0,002162  0,000530  0,000205   -0,000008  0,000596   0,000383
0,010  -0,003193  0,002645  0,001280   -0,000015  0,002258   0,000427
0,020  -0,002941  0,001923  0,000775   -0,000013  -0,000508  0,000598
0,030  0,000869   0,000542  0,000129   -0,000000  -0,002076  0,000295
0,040  0,000703   0,003234  0,000398   0,000001   -0,003874  -0,000573
0,050  0,001065   0,001131  -0,000630  0,000017   -0,002438  -0,000626
0,060  0,005938   0,003836  -0,000669  0,000015   -0,000689  -0,001197
0,070  0,000362   0,002768  -0,000732  -0,000005  0,002879   -0,001018
0,080  -0,001838  0,000678  -0,000241  0,000004   0,003595   -0,000771
...
```

Figure 4.5 shows how the loading interface looks.



New input data

Select Input Data Folder — Browse

Folder location

NFFT: 4096

Number of rows in the dataset: 100000

Sampling Frequency (Hz): 100

Channels: 1,2,3,5,6

Start

Save FDD file

Figure 4.5: New input data.

First, press *Browse* button to select the folder in which the data are stored. Then, set the parameters required for the FDD analysis.

- *NFFT* is the FFT length which determines the number of frequencies in which the frequency domain is subdivided using the discrete Fourier transform.
- *Number of rows in the data-set* is the minimum number of measurements within each file.
- *Sampling frequency* is the number of samples per unit of time.
- *Channels* are the columns of each file to consider in the analysis, separated by commas.

The *Start* button loads the compatible files found inside the selected folder and performs *Cross Power Spectral Density* and *Singular Value Decomposition*. Once you have completed this first phase of analysis, you may save the temporary FDD file using the *Save FDD* button, or simply go on with the analysis and move on to phase two.

Existing FDD file

If you previously saved a temporary FDD file, you can select it using the *Browse* button and then load it using the *Load existing FDD file* button. This allows you to skip the step of calculating the CPSD and SVD matrices.

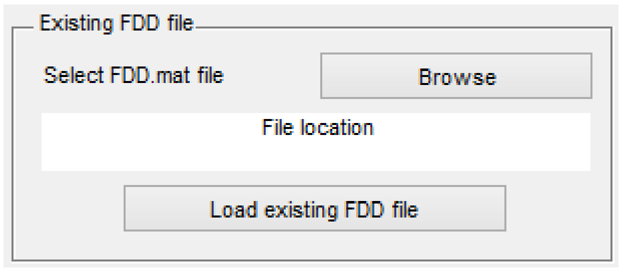
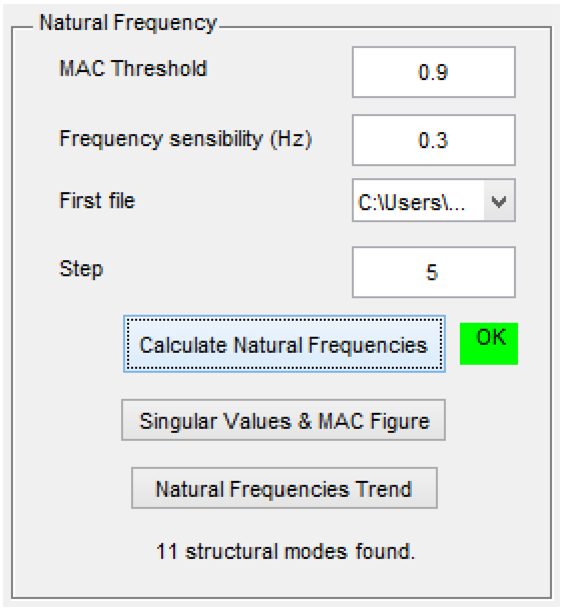


Figure 4.6: Existing FDD file.

4.3.2 Step 2: identification of natural frequencies and damping ratios

Natural frequencies identification

To perform the natural frequencies identification, press the *Calculate Natural Frequency* button.


Natural Frequency
Calculate Natural Frequencies
OK
Singular Values & MAC Figure
Natural Frequencies Trend

MAC Threshold 0.9
Frequency sensibility (Hz) 0.3
First file C:\Users\...
Step 5
11 structural modes found.

Figure 4.7: Natural frequency identification panel.

Required user input The parameters required for the Natural Frequency identification are the following:

1. *MAC Threshold*
2. *Frequency sensibility*
3. *First file*
4. *Step*

Process To understand their purpose you need to consider how the frequencies identification works. For an in-depth explanation of this procedure, look at [36].
In Step 1 power spectral density matrices were calculated and subsequently decomposed using the singular value decomposition.
The singular value matrices are then divided into subgroups, whose dimension is set by the user through the *Step (4)* parameter. All files preceding the *First file (3)* specified by the user are ignored. Each subgroup contains a number of *svd* matrices sets (containing U and S) equal to *step*.
Within each subgroup, for each frequency, a MAC value is calculated by comparing the singular vectors related to each measurement files belonging to the subgroup. A mean MAC value is then calculated between the MAC of each subgroup.
A high mean MAC value in a certain frequency sub-domain indicates a strong correlation between the singular vector and thus it is possible that a mode of vibration may be present in that sub-domain.
In order to adequately separate the potential vibrating modes, a MAC Threshold is set (by the user, through the *MAC Threshold (1)* parameter).
The singular values can be displayed using the *Singular Values & MAC Figure* button. The red line shows the *MAC Threshold* specified by the user and at what level filters the MAC values.

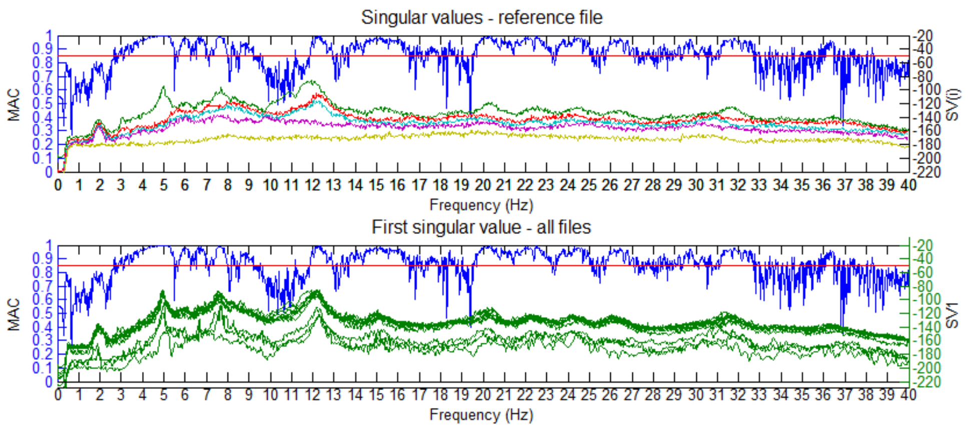


Figure 4.8: Singular Values.

Not all mean MAC peaks that exceed the threshold value are actually structural modes, only the broadest are considered: the parameter *Frequency sensibility (3)* sets the minimum allowable width in frequency for which the values that exceed the threshold are considerable as part of a modal domain.

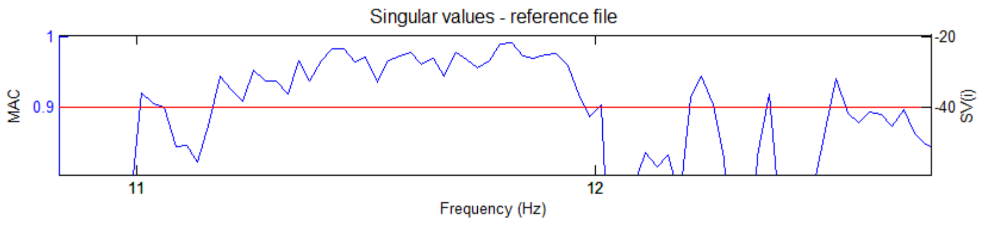


Figure 4.9: MAC Treshold and non-structural peaks filtering.

At higher values of this parameter could correspond a lower sensitivity to the weaker structural modes, while lower values are likely to also identify non-structural modes. A typical case that explains the meaning of the *Frequency sensibility* parameter is shown in figure. It is clear that the frequency subdomain from $11,2$ to 12 Hz contains a structural mode, while the other peaks must be excluded.

Selected potential modal domains are then subject to further analysis to confirm that they actually represent structural modes.

Results Once this step is completed, the natural frequencies found are displayed in the *Results* block.

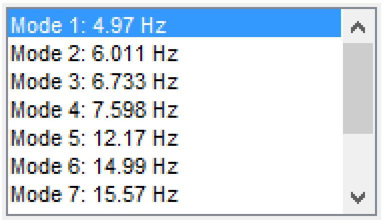

Mode 1: 4.97 Hz
Mode 2: 6.011 Hz
Mode 3: 6.733 Hz
Mode 4: 7.598 Hz
Mode 5: 12.17 Hz
Mode 6: 14.99 Hz
Mode 7: 15.57 Hz

Figure 4.10: Identified Natural Frequencies.

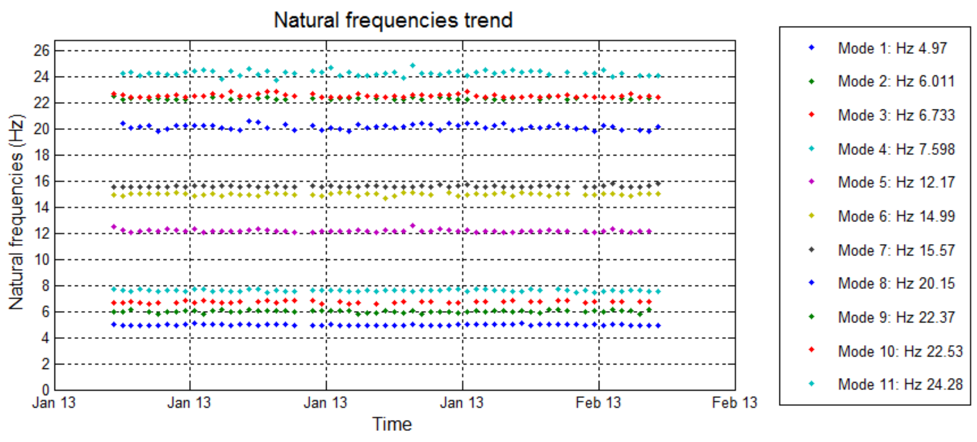
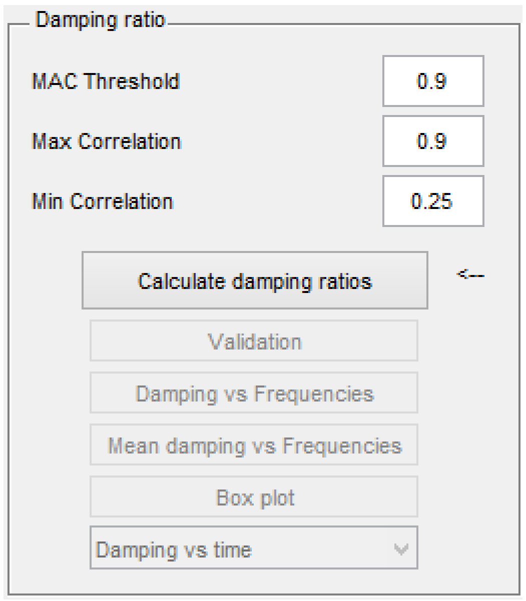


Figure 4.11: Natural frequency trend.

You can observe the trend of the natural frequencies found for each file using the button *Natural Frequencies Trend* (Figure 4.11). You can launch the figure that represents the singular values, pressing the *Singular Values Figure* (Figure 4.8).

Damping Ratio Estimation

To calculate Damping Ratios press the *Calculate damping ratios* button.


Damping ratio

MAC Threshold 0.9

Max Correlation 0.9

Min Correlation 0.25

Figure 4.12: Damping Ratio Estimation Panel.

Required User Input Three parameters are required to perform the process:

1. *MAC Threshold*
2. *Max Correlation*
3. *Min Correlation*

Process The calculation of the damping ratio can be decomposed into three main steps.

1. Identification of the spectral bell: the SDoF Auto-spectral functions are identified around the natural frequencies using MAC criterion;
2. Transformation of the spectral bell to time domain: the normalized SDoF auto-correlation functions are calculated;
3. Damping ratio estimation using logarithmic decrement of the absolute extreme values in the range set by the user.

Identification of the spectral bell The power spectral density function is identified around each peak by comparing the mode shape estimate with the singular vectors for the frequency lines around the peak. As long as a singular vector is found that has MAC value with the mode shape estimate the corresponding singular value belongs to the SDoF density function.

$$MAC = \frac{(u_{rif} u_f)^2}{u_{rif}^2 u_f^2} \tag{4.3.1}$$

The singular values for which $MAC < \Omega$ (where Ω is the user-specified parameter *MAC Threshold (1)*), are set to zero. Therefore only the frequencies around the peak that satisfy the condition $MAC > \Omega$ are considered and constitute the spectral bell.

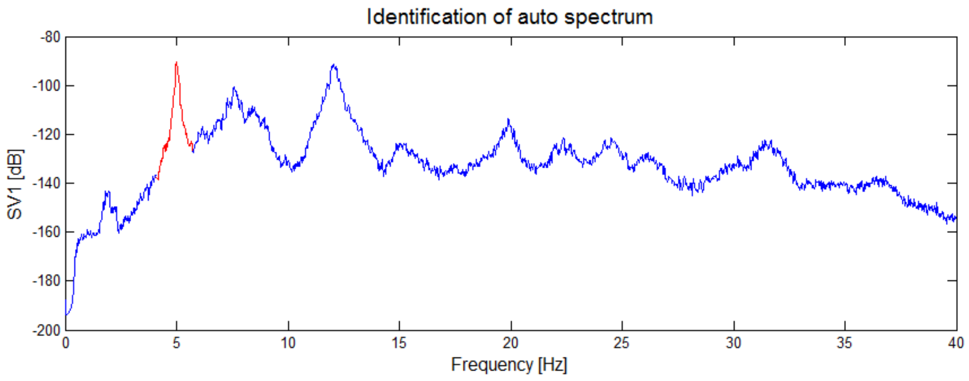


Figure 4.13: Identification of the autospectrum and of the SDoF Auto-Spectral Function.

Setting a higher value for the *MAC Threshold* would shrink the spectral bell, while a lower value would widen it.

Transformation of the spectral bell to time domain From the fully or partially identified SDOF auto spectral density function, the natural frequency and the damping are obtained by taking the spectral density function back to time domain by inverse FFT. A n-point inverse discrete Fourier transformation is used, where n is the previously specified $NFFT$ parameter.

The output given by the Fourier inverse transformation, is then normalized and represents the *Normalized Correlation Function of the Singular Value Spectral*

Bell, and is shown in Figure 4.14.

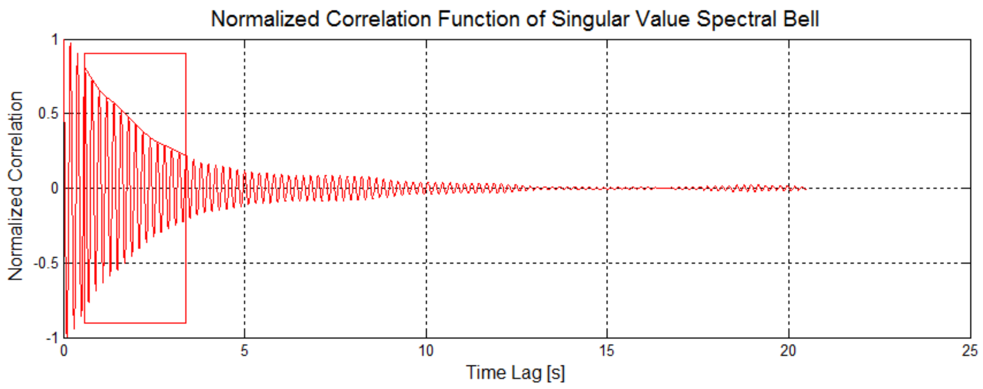


Figure 4.14: Normalized Correlation Function of the Singular Value Spectral Bell.

Damping ratio estimation The rectangle, visible in Figure 4.14, defines the values used for the calculation of the damping ratio. Its limits are specified by the user through the parameters *Max Correlation (2)* and *Min Correlation (3)*. The first part and the tail of the correlation function are excluded from the analysis, as these parts are more sensitive to noise.
For Max Correlation are recommended values around 0.9 and for Min Correlation around 0.2. Since these values influence the final value of damping ratio, it is recommended to calibrate them on the basis of expected damping results, if available.

First all extremes r_k, both peaks and valleys, on the selected part of the correlation function are found. The logarithmic decrement δ is then given by:

$$\delta = \frac{2}{k} ln \left(\frac{r_0}{|r_k|} \right) \tag{4.3.2}$$

where r_0 is the initial value of the correlation function and r_k is the kth extreme. r_0 and r_k correspond to the maximum and minimum values of correlation defined by the user, while k is the number of extreme values (maximum and minimum) present in the interval.

The more the curve exhibits a linear logarithmic decrement, the more the damping value will be significant.

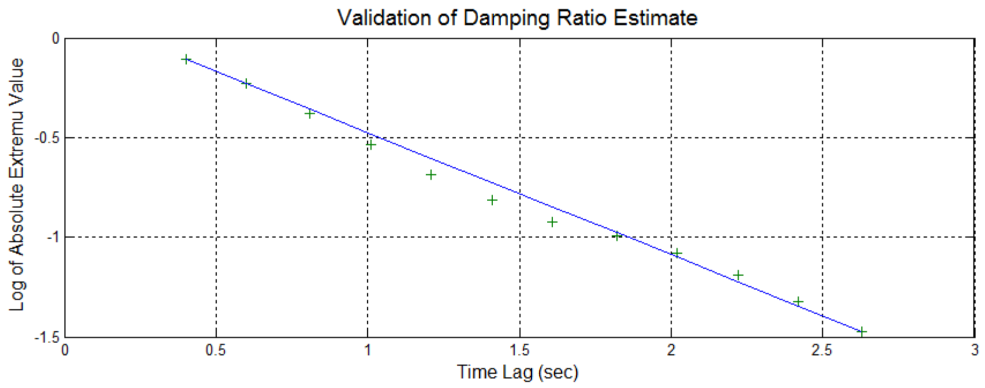


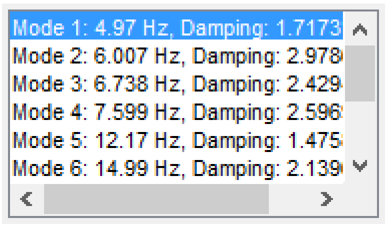
Figure 4.15: Validation of Damping Ratio Estimate: Logarithmic Decrement.

The damping ratio is given by the following formula:

$$\zeta = \frac{\delta}{\sqrt{\delta^2 + 4\pi^2}} \tag{4.3.3}$$

An estimate of the natural frequency is also calculated, by simply counting the number of times in which the correlation function is zero in the range considered.

Results The main output given by this block consists on the estimated damping ratio results for each measurement set and for each structural mode. The mean damping calculated for each structural mode (excluding the outliers) is displayed in the results block.



```
Mode 1: 4.97 Hz, Damping: 1.7173
Mode 2: 6.007 Hz, Damping: 2.978
Mode 3: 6.738 Hz, Damping: 2.429
Mode 4: 7.599 Hz, Damping: 2.596
Mode 5: 12.17 Hz, Damping: 1.475
Mode 6: 14.99 Hz, Damping: 2.139
```

Figure 4.16: Identified Mean Natural Frequencies and Damping Ratios.

The damping results are saved in the results management block.
In the damping identification block it is possible to launch some graphs useful for understanding and analysing the results.

- *Damping vs Frequency*: puts to graph (Figure 4.17) all values of damping ratio calculated with respect to the relative natural frequencies determined in the frequency domain.

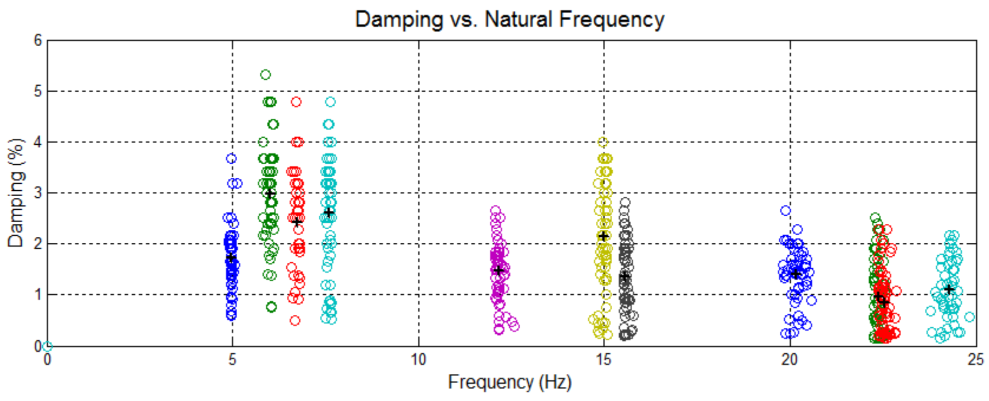


Figure 4.17: Damping versus Natural Frequency Graph.

- *Mean damping vs Frequencies*: shows the mean values of the damping ratio compared with mean values of the relative natural frequencies (Figure 4.18).

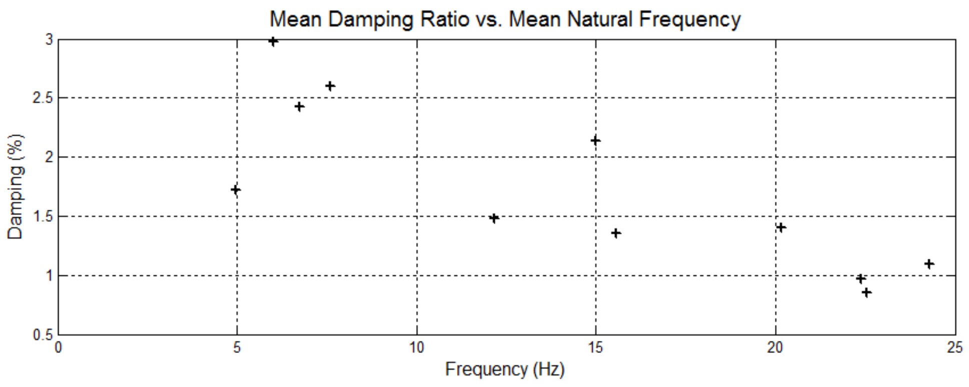


Figure 4.18: Mean Damping Ratio vs. Mean Natural Frequency Graph.

- *Box plot*: it is a statistical description of the data through the quartiles (Figure 4.19).

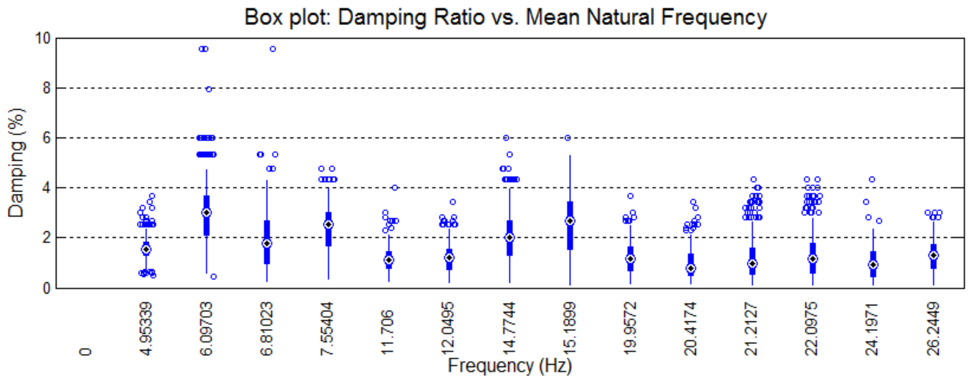


Figure 4.19: Box Plot.

What is a Box-Plot The meaning of a box-plot is explained in Figure 4.20. It is clear that, the narrower the Inter-Quartile Range is, the less data are dispersed and the better the estimate of the damping ratio is.

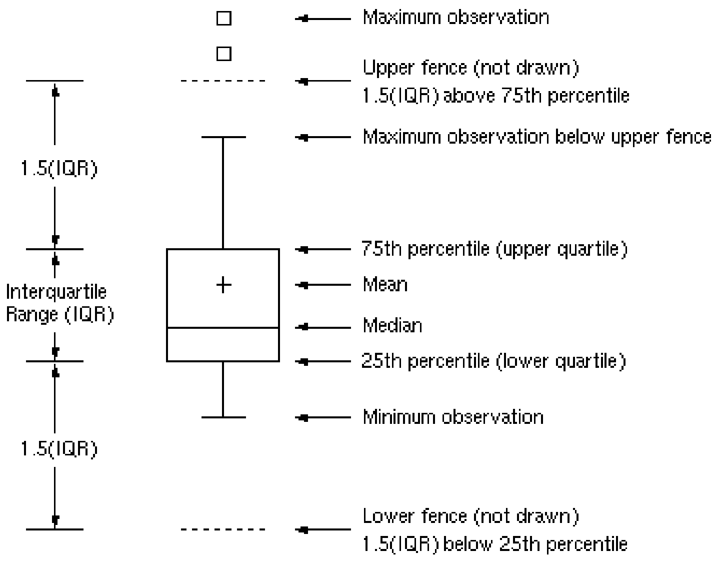

Maximum observation
Upper fence (not drawn)
1.5(IQR) above 75th percentile
Maximum observation below upper fence
1.5(IQR)
75th percentile (upper quartile)
Interquartile Range (IQR)
Mean
Median
25th percentile (lower quartile)
Minimum observation
1.5(IQR)
Lower fence (not drawn)
1.5(IQR) below 25th percentile

Figure 4.20: What is a Box Plot.

- *Damping ratio trend*: it shows, for each structural mode, the trend over time of the calculated damping ratio (Figure 4.21).

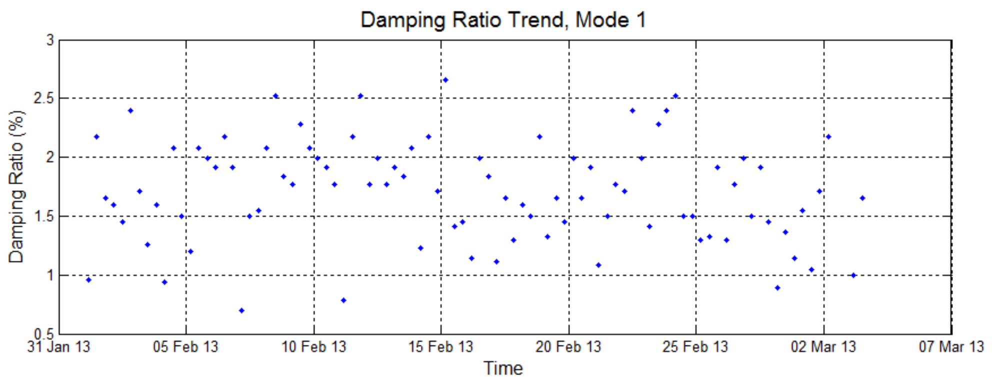


Figure 4.21: Damping Ratio Trend.

It is also possible get more detailed information on the process of the damping ratio calculation by opening the *Validation* window (Figure 4.22).

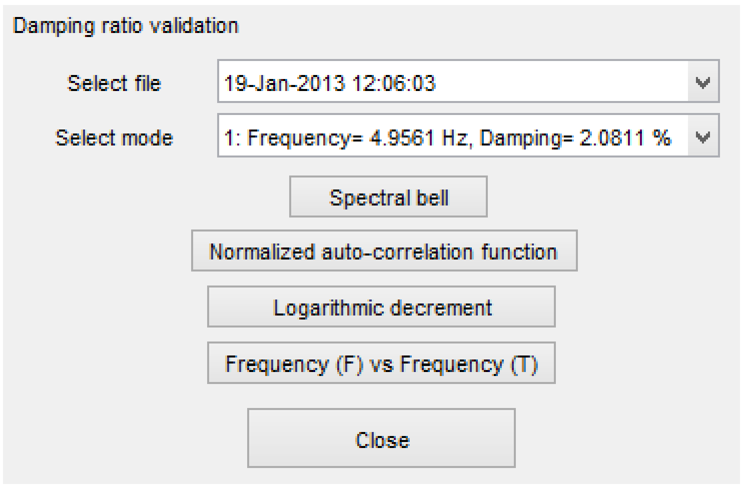

Damping ratio validation
Select file
19-Jan-2013 12:06:03
Select mode
1: Frequency= 4.9561 Hz, Damping= 2.0811 %
Spectral bell
Normalized auto-correlation function
Logarithmic decrement
Frequency (F) vs Frequency (T)
Close

Figure 4.22: Damping ratio validation tool.

This screen allows the user to observe, for each input file, and for each structural mode, some graphs useful in understanding the process and possibly make changes in the configuration parameters. Initially the user must select the file to check and the structural mode of interest from the two drop-down menu. Then it is possible to launch the following graphs:

- *Spectral bell* shows the selected SDOF auto spectral density function, which is subsequently transformed in time domain, as in Figure 4.23. If the selected SDoF auto-spectral density function includes more than one peak the user might want to edit the input parameters for damping ratio calculation.

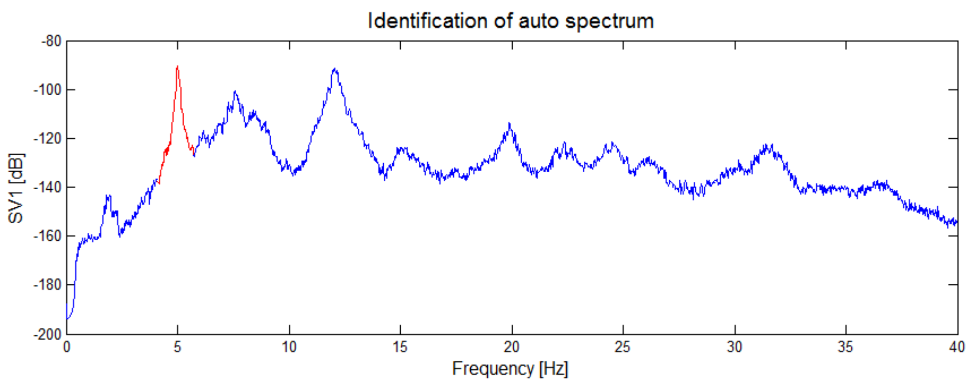


Figure 4.23: Selected SDOF auto spectral density function.

- *Normalized auto-correlation function* shows (Figure 4.24) the result of the normalized inverse Fourier transform of the SDOF auto spectral density function.

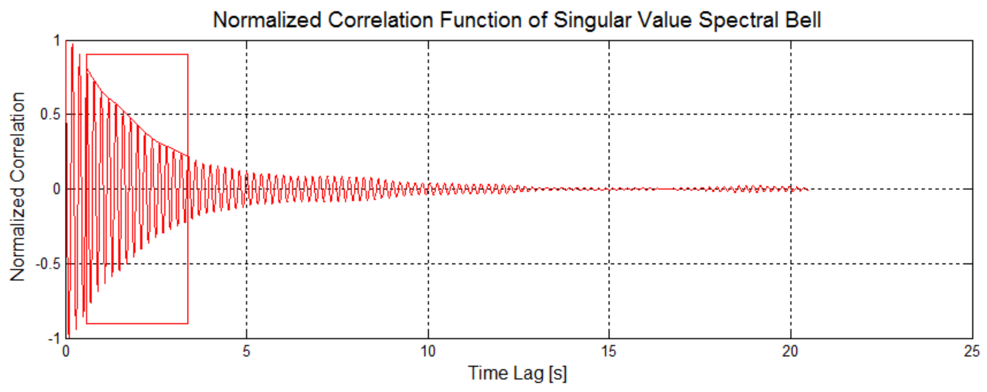


Figure 4.24: Normalized auto-correlation function.

- *Logarithmic decrement* shows (Figure 4.25) the logarithmic decrement of the absolute extreme values compared with the linear trend assumed in the calculation of the damping ratio. If the logarithmic decrement of the absolute extreme values is far from a linear behaviour, the user should check the input parameters.

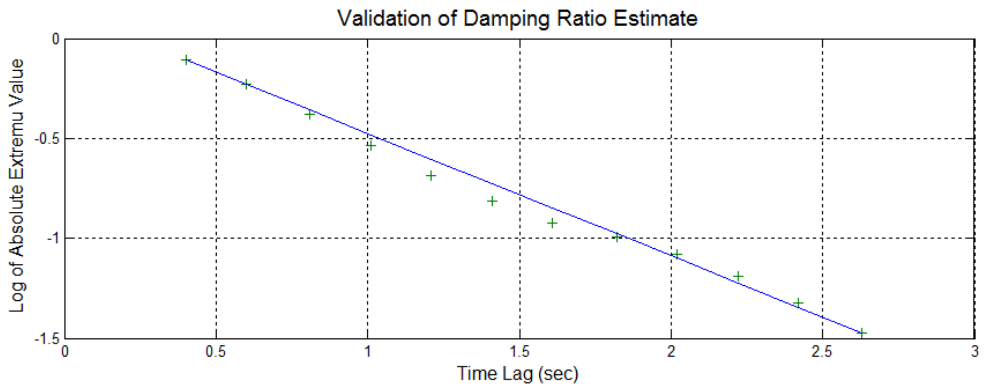


Figure 4.25: Logarithmic decrement of the absolute extreme values.

- *Frequency(F) vs Frequency(T)*: Figure 4.26 shows the relationship between the frequencies previously calculated in the frequency domain and the values estimated in the time domain using Enhanced Frequency Domain Decomposition. Values that deviate from linearity may indicate coupled structural modes or bad selections of the SDoF auto spectrum. If natural frequencies calculated in time domain (using EFDD) are very different than those calculated in frequency domain, then the structural modes are not properly selected in frequency domain.

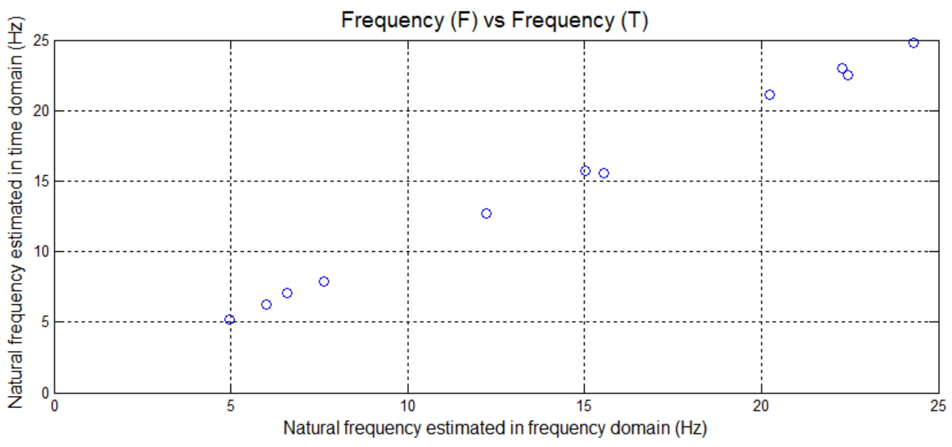


Figure 4.26: Natural frequencies calculated in frequency domain vs natural frequencies calculated in time domain.

4.3.3 Results management

Once the analysis is completed, the results can be saved inside a file. After a results file is saved, it is possible to view this file or to merge it with other compatible results files. The compatibility between results files will be discussed further.

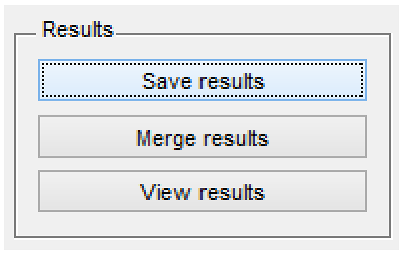


Figure 4.27: Results management panel.

Saving the results

When the calculation of natural frequencies and damping ratios is complete, the *Save Results* button becomes enabled . Once the user presses the button, a dialog window is displayed in order to select the file name and its path inside the computer.
After the saving process is complete a dialog will confirm the save.

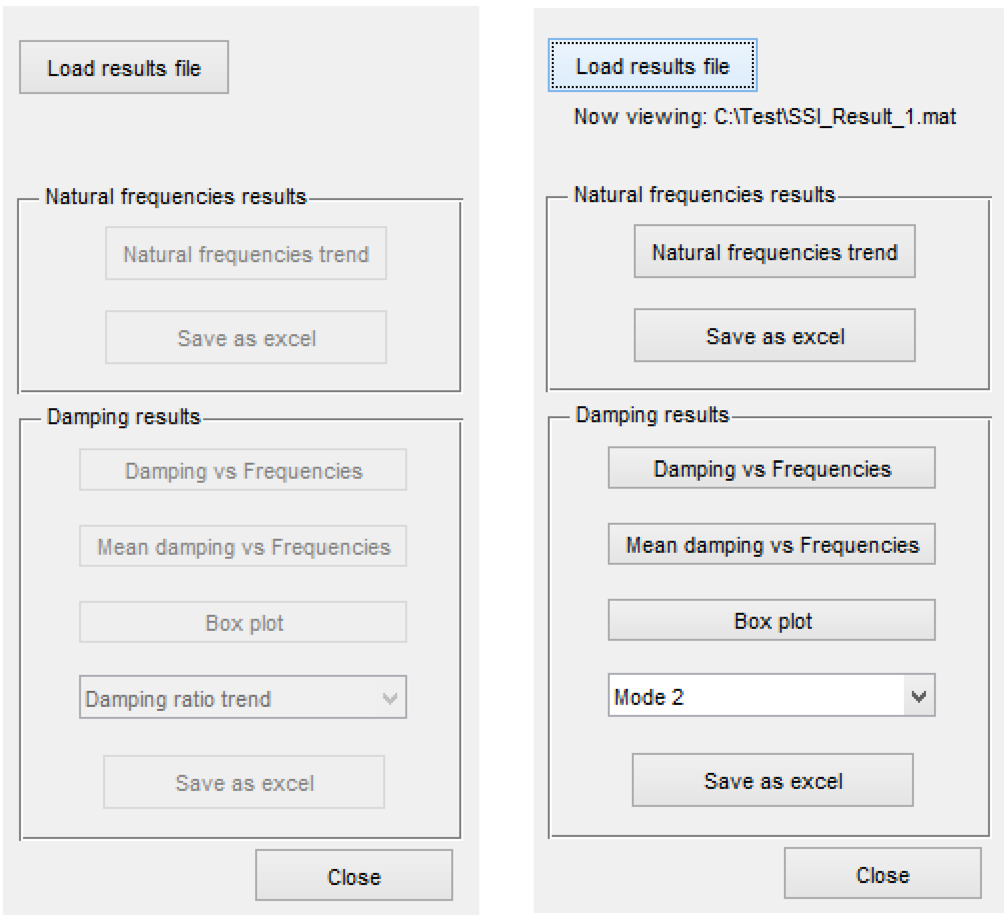
Viewing the results

The *View results* button provides a simple interface that allows the user to display and export the results of a particular file.
First, select a compatible results file pressing the *Load results file* button. Every result saved within the SIM/AtOMA program is compatible, so both *AutoEFDD* and *AutoSSI* results files can be selected.
View results provides some graphs for the reading of the results. All these graphs are also available in the analysis phase and have already been discussed.

- Natural frequencies results
  - *Natural frequencies trend*: the trend of the natural frequencies found for each file.

- Damping results
  - *Damping vs Frequency*: puts to graph all values of damping ratio calculated with respect to the relative natural frequencies determined in the frequency domain.
  - *Mean damping vs Frequencies*: shows the mean values of the damping ratio compared with mean values of the relative natural frequencies.
  - *Box plot*: it is a statistical description of the data through the quartiles.
  - *Damping ratio trend*: it shows, for each structural mode, the trend over time of the calculated damping ratio.

In addition to these charts, you can export the results to excel and save the graphics as an image.



Now viewing: C:\Test\SSI_Result_1.mat

Figure 4.28: 'View results' interface.

Merging the results

It is possible to merge multiple results files into one file using the merge results button. A simple interface is displayed:

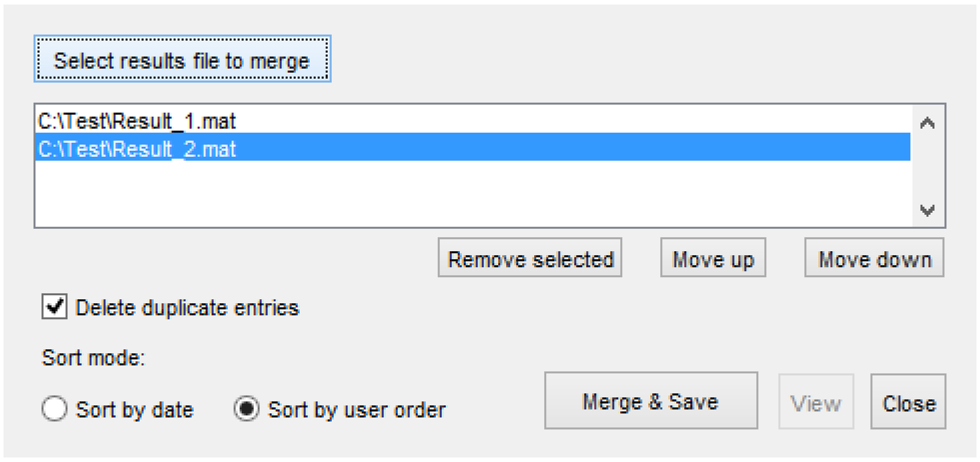


Figure 4.29: 'Merge results' interface.

Press *Select results file to merge* button to load two or more results files. It is possible to remove a file from the list by pressing the *Remove selected* button; a file can be moved up or down on the list by pressing the *Move up* or the *Move down* buttons.
By checking the check-box *Delete duplicate items*, if two items belonging to the files in the list are attributable on the same date of measurement, only the item belonging to the higher file in the list will be preserved in the merged file. If the check-box is not checked, no item will be deleted.
Selecting *Sort by date*, the elements will be sorted in ascending order based on the date of measurement. Otherwise, selecting *Sort by user order*, the order set by the user in the list will be kept in the merged file.
Pressing the *Merge & Save* button the files are merged according to the specified settings and a dialog to save the merged file is displayed.
The *View* button will open the *View Results* window, with the merged file already loaded.

Not all files can be merged In fact, only files with the same number of structural modes are compatible. The results obtained from a template are always mutually compatible.

4.4 Structural monitoring using AutoEFDD method

4.4.1 Monitoring using templates

Structural monitoring requires the ability to update the analysis over time with the new measurements performed on the structure.
AutoEFDD method provides a fast and efficient way to do so: once a first analysis is completed and the major structural modes of the structure are identified, a *Template* file can be saved. The template file contains information about the structure upon which new analysis will be based.
When a new analysis is started on the basis of an existing template the program skips the modal domain identification phase. The auto-spectrum functions of the new files are analyzed in correspondence of the modal domains previously identified and listed in the template, in search of the vibrating modes of the structure. If a vibrating mode exists inside the modal domain, the corresponding natural frequency and damping ratio are calculated and automatically saved in a results file, which is then merged with the previous results files.

Therefore, once you have a template file, simply select the new measurement files available and launch analysis to update the time profile of the results.

Saving a template

To save a template file, complete an identification analysis using AutoEFDD method, then press the *Save template* button.

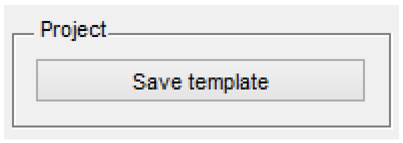


Figure 4.30: Save template button.

The following dialog is then displayed:

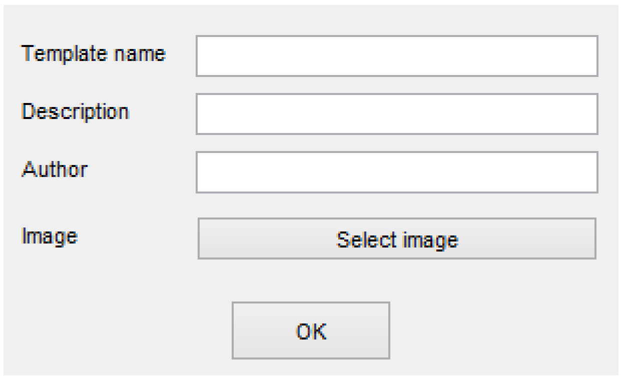


Figure 4.31: Save template dialog.

It is required to set a series of details for the template file:

- *Template name*: an identifying name for the template. Only alphanumeric characters are allowed.
- *Description*: some useful details regarding the design or analysis.
- *Author*: the reference operator who carried out the first analysis.
- *Image*: select an image file representing the structure.

Once you have specified the details, press OK to save the template. The template is automatically saved within the template folder of the program in **.efdd* file format.

New from template

To start a new analysis based on a template select from the drop-down menu *File → New from template.*
A dialog prompts the user to select the template file. Once a template file is loaded, a summary screen of the analysis is shown, which contains the data entered by the user when saving the template.
It is then required to select input files using the *AutoEFDD: New input data* panel. Once the user presses the START button the analysis is carried out automatically using the parameters of the template. The results are saved in the Results folder of the program and automatically merged with the older files.

4.4.2 Automatic monitoring using schedules

AtOMA allows the user to set the automatic execution of dynamic monitoring tasks at specific times through the use of *Schedules.* A schedule is a configuration file that contains information about the required analysis:

- The template to be used for analysis;
- The hours in which to launch the analysis;
- The folder where the new measurement files are placed.

This function is particularly useful in the case in which the computer where AtOMA is installed automatically receives new files measuring one or more times per day, because it allows the operator to have information on the dynamic state of the structure without first having to transfer the new measurement files, an operation that in the case of files of large dimensions can be particularly onerous if made via the internet network.

Creating a new schedule file

To create a new schedule file, select from the drop-down menu *Schedule* $\rightarrow$ *New schedule.*
A window will be displayed. To create a schedule file is required to enter some details:

- The name of the Schedule (only alphanumeric characters allowed);
- The template on which to perform the analysis;
- The folder where new measurement files are put;
- The times in which it is scheduled to run automatic analysis.

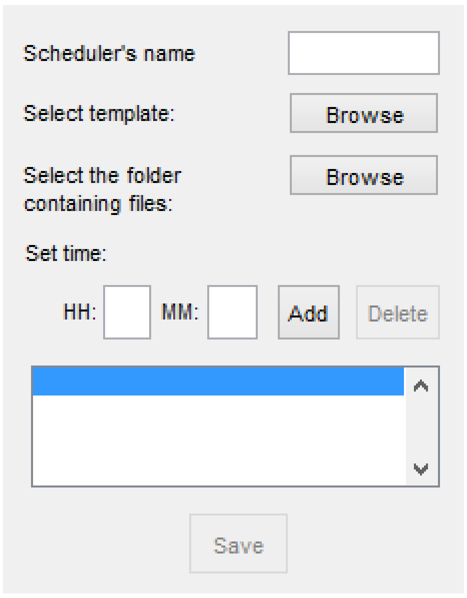


Figure 4.32: Creating a new schedule.

Once the configuration is completed, press *Save* button to save the schedule. The schedule will be saved as a file in the *Schedule* folder of AtOMA program.

Schedule launcher

In order to start a scheduling task, the user has to select from the drop-down menu the voice *Schedule* → *Schedule Launcher*. It is required to select the schedule file.

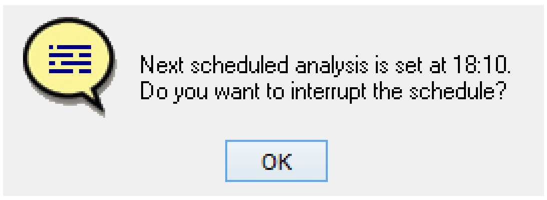

Next scheduled analysis is set at 18:10.
Do you want to interrupt the schedule?

Figure 4.33: Waiting for a scheduled event.

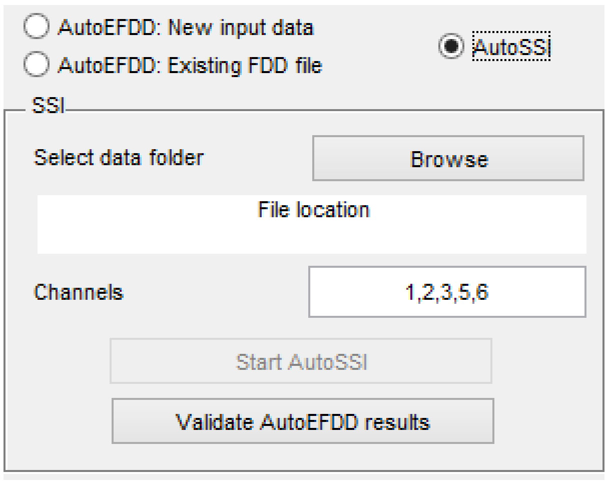
Once the scheduler is active, the time of the subsequent execution of the analysis is displayed. To interrupt the scheduler, press OK in the dialog. At the specified time, AtOMA will launch the analysis without requiring any user input. The results will be saved and merged with the previous ones and the scheduler will wait for the subsequent scheduled execution.

4.5 AutoSSI method

Stochastic Subspace Identification is the second method implemented in AtOMA for structural identification purpose. The main reason for which this method was implemented lies in its strength and reliability. The SSI method is, in fact, considered to be one of the most powerful tools for the structural dynamic identification. SSI is however characterized by a high computational burden that does not makes it very suitable for monitoring operations, or analysis of a large number of measurement files.
The aim is therefore to provide the results of reference on which to base the analysis with the method AutoEFDD, in the absence, or in addition, to the results provided by an FE model of the structure.

4.5.1 Identification using AutoSSI

To select the AutoSSI method for dynamic identification, check the *AutoSSI* check-box in the *input data* block.



AutoSSI [✓]

Channels: 1,2,3,5,6

Figure 4.34: AutoSSI panel.

Then, select the folder containing the measurement files by pressing the *Browse* button.
Channels is the list of valid channels of the measurements files, divided by commas. To start the SSI analysis, press the *Start SSI* button. Please notice that the analysis might take a lot of time and might be very demanding for

the computer's resources.
After The analysis is completed, it is asked to save the results. The results are, like the ones provided with the AutoEFDD method, composed of two tables: *Result_frequency*, containing the natural frequencies, and *Result_damping*, containing the damping ratios.
The results can be displayed using the *View results* button.
Since AutoSSI procedure is much more onerous than AutoEFDD, it has been implemented in order to validate the results provided by AutoEFDD. *Validate AutoEFDD results* is a tool that allows the user to compare the results obtained with AutoEFDD and AutoSSI. It provides information about the relative error between the average results of each structural mode obtained by means of the two methods in the form of two tables (one for the natural frequencies and the other for the damping ratios).

4.6 Tutorial: Ponte sul Mincio, Verona

In this section, an application of the tools in **AToMA** for the identification and monitoring of dynamic structures, will be carried out in relation to a bridge located along the A4 highway in the municipality of Peschiera sul Garda, as a step-by step tutorial for the program.

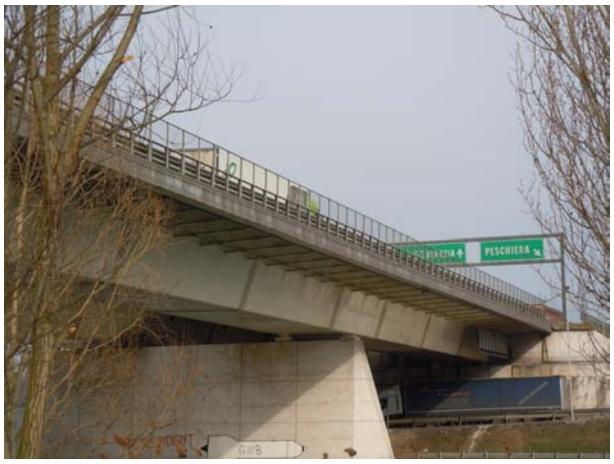


Figure 4.35: Bridge over the River Mincio.

It is assumed to operate in typical conditions in order to perform a dynamic monitoring. Initially an identification of the dynamic structure, with relative calibration of the parameters of the program, is performed. Subsequently a dynamic monitoring system is configured by means of the scheduling tools. Finally, the results provided by the monitoring will be analyzed and compared with the dynamic characteristics of the FEM model of the structure.

4.6.1 Inventory of information

The structure is a three-spans bridge constituted by two side-by-side independent continuous beams, with a box steel section, leaning on concrete piers and abutments [37].
The side spans of the viaduct are 41 m long, while the central span is 70 m long. The total length of the girders measured between the axes of the supports on the abutments is therefore equal to 152 m.
The total width of the deck, for each of the two girders, is equal to 15.55 m, 14.50 m of which are used by the carriageway and 00.50 +0.55 = 1.05 m are occupied by the guard-rails.
The current configuration of the road is due to the works of widening of the road, carried out in the years 1991-1992, which enlarged it from the initial 25 m to the current 33 m. In the occasion of such works the girders of pre-stressed concrete have been replaced with the current steel box girders, while piers and abutments have undergone reinforcement, cant and, only for the abutments, side extension.
The cross section of the bridge has a constant height, approximately equal to 3.30 m, and is constituted by a single-celled body of trapezoidal shape having the bottom flange width of 5 m and the sides wide apart symmetrically upwards forming, with the upper plate of the deck, the upper base of the trapezoid with a width of 7 m.

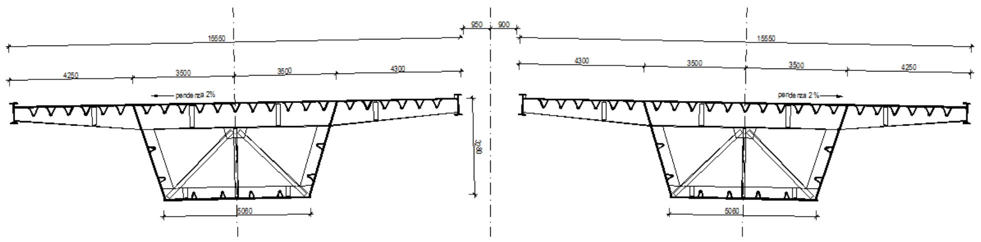


Figure 4.36: Cross section of the decks in the span.

The cross section of the deck is then completed by two lateral cantilevers, supported by continuous crosspieces on three supports, which are part of the diaphragms transverse stiffening of the body. The road surface was obtained directly on the top flange of the girder, through processes of blasting and waterproof protective flooring with application of a special polymer conglomerate. The plates constituting the thin walls of the cross section of the girders of the bridge are made longitudinally rigid through the use of "V" stiffeners, with are welded with continuity to the plates and form, with these, closed sub-cells. The thickness of the plates and the distribution of the stiffeners, are as follows:

- Top flange:
  - Metal sheet with constant thickness for all types of cross-section equal to 13 mm;
  - "V" stiffeners with a thickness of 7 mm, with a gap equal to 30 cm, distributed at a distance of about 60 cm.
- Bottom flange:
  - Metal sheet of variable thickness, ranging from a minimum of 14 to a maximum of 28 mm;
  - "V" stiffeners with a thickness of 7 mm, in number of 4, distributed at a distance of about 100 cm,
  - "T" stiffeners with a height of 200 mm, consisting of profiles 10 mm thick, replacing the "V" stiffeners, continuing along the axis line for a distance of approximately 3 m before and after the intermediate supports (on the piers).

- Lateral sides:
    - Variable metal sheet thicknesses ranging from 13 to 16 mm,
    - "V" stiffeners, 7 mm thick, in number of 2, respectively at 70 cm and 180 cm from the bottom edge of the sides.

For all plates S355 steel was used, while the corner pieces that form the intermediate diaphragms utilize S235 steel.
Each of the girders is constrained in four cross sections characterized by the presence of particularly stiff continuous diaphragms, namely in correspondence of the abutments and of the two intermediate piers. There are two support apparatus for each of these sections: they are arranged symmetrically with respect to the vertical plane of symmetry of the section itself, at a distance from the section of approximately 1.80 m. Their arrangement does not contrast the deformations of the structure due to both operational forces and thermal variations.

4.6.2 Overview of damage

Following a series of inspections on the structure in 2011 performed by *SM Ingegneria S.r.l.*, an overview of the state of deterioration of the bridge was outlined.
Inspections carried out inside the box girder of the two decks have revealed damage to the structure related to the phenomena of fatigue. Some of the major issues on the structure are constituted by [37]:

- Cracks on the outer ribs;
- Cracks on the inner ribs;
- The infiltration of material from the roadway within some stiffeners resulting in a high state of corrosion;
- The top plate was damaged by the milling of the layer of asphalt of the road.

4.6.3 Structural analysis

Model preparation

A finite element model of the structure was developed using the Strand7 software (HSH Computing, Padua).
In this study case only one type of solver is used: *Natural Frequency Analysis*. The model is three-dimensional type (Figure 4.37).

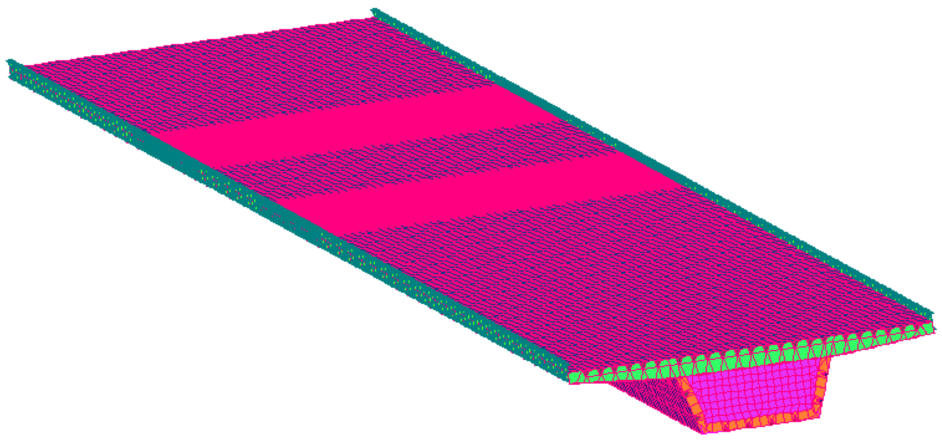


Figure 4.37: FE model of the bridge.

The side plates of the deck are modeled using plate elements characterized by a variable thickness over the length of the deck, ranging from 13 mm to 16 mm (Figure 4.38).

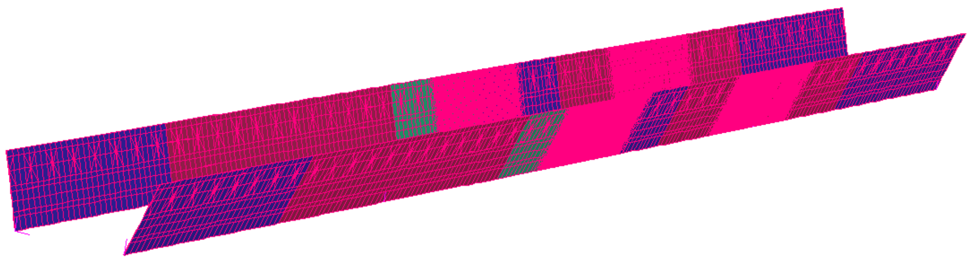


Figure 4.38: Modeling of the side plates of the deck.

The top and the bottom flanges are modeled using plate elements. The thickness of the bottom flange is variable over the length of the bridge. The meshing is more dense in correspondence of the piers (Figure 4.39).

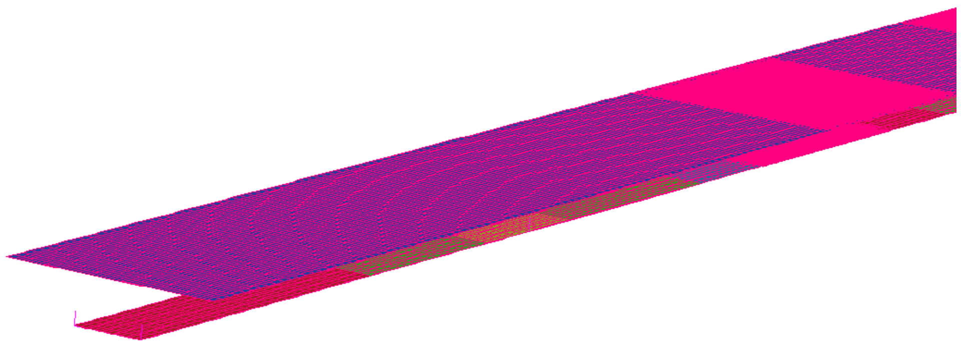


Figure 4.39: Modeling of the top and bottom flanges of the deck.

The ribs are modeled as plate elements. The property is constant along the length of the deck (Figure 4.40).

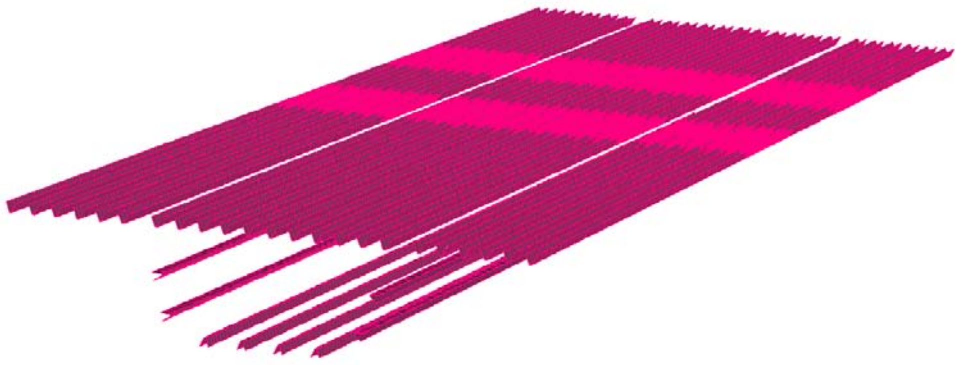


Figure 4.40: Modeling of the ribs of the bridge.

The edge beams and the transverse joists are modeled using beam elements (Figure 4.41).

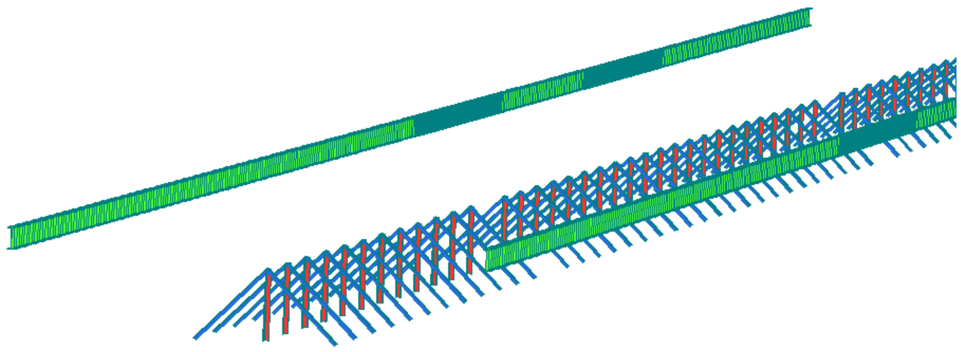


Figure 4.41: Modeling of the edge beams and of the transverse joists of the bridge.

The mechanical properties of the components of the model were initially assumed as the design characteristics of the materials.

Dynamic analysis

The purpose of the *Natural frequency analysis* is to provide an interpretation of the structural behaviour of the structure in order to allow an adequate choice for the placement of the measurement instruments. The structural vibration modes and the corresponding natural frequencies represent the structural response of the structure. The superposition of all the structural modes provides, in fact, the global dynamic behaviour of the structure.
Following the dynamic analysis of the structure, the results for the first 8 modes of vibration observed by means of the FE model are shown.

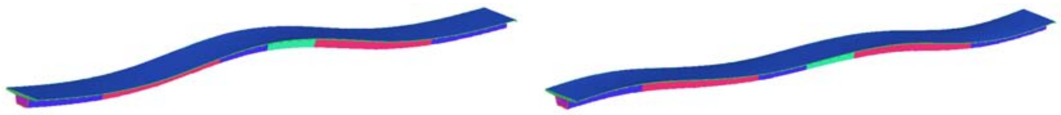


Figure 4.42: Mode 1: 2.73 Hz and Mode 2: 5.40 Hz

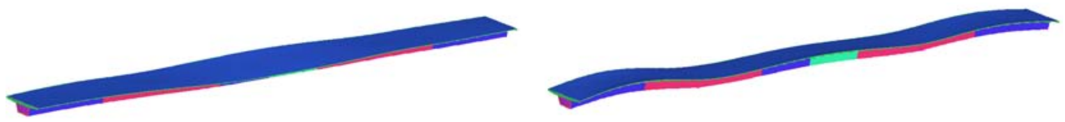


Figure 4.43: Mode 3: 6.01 Hz and Mode 4: 6.17 Hz

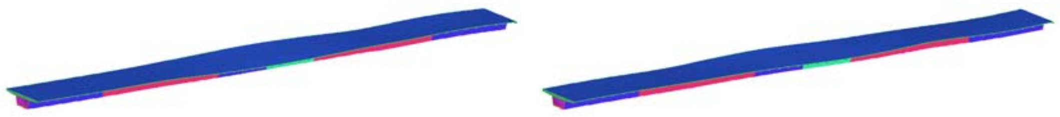


Figure 4.44: Mode 5: 7.22 Hz and Mode 6: 7.84 Hz

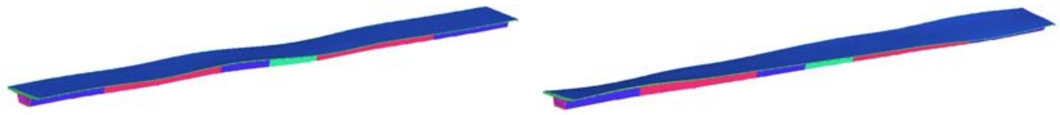


Figure 4.45: Mode 7: 8.31 Hz and Mode 8: 8.72 Hz

4.6.4 Dynamic monitoring

The results of the dynamic FE model made it possible to arrange the installation of the monitoring system in order to perform the modal testing.
Following test installations made during 2012, the final installation of the monitoring system has been completed July 20, 2012 [38].

Insitutions involved	UNIPD
Period of implementation	July 2012 - December 2012
Date of activation	20/7/2012
Dynamic System	56 Strain gauges
State of the system	Not active

Table 4.1: Details for the measurements on Ponte sul Mincio

The monitoring system is composed of 56 channels and uses a series of strain gauges as measuring instruments (Figure 4.46). The installation was performed by *EXPIN s.r.l. - ADVANCED STRUCTURAL CONTROL*.
Since the strain gauges measure the deformation of a particular object, a conversion has been necessary in order to use the measurements for a dynamic analysis using **AtOMA**.

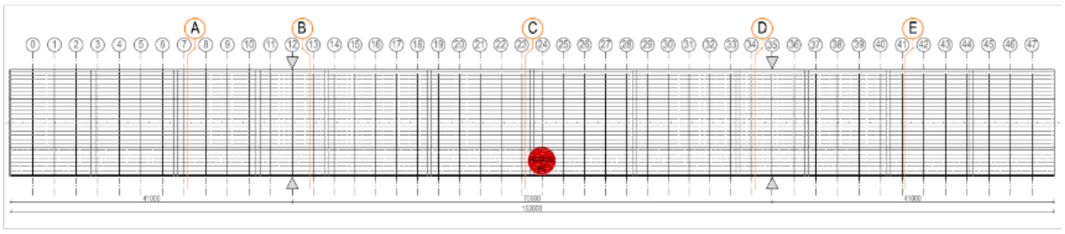
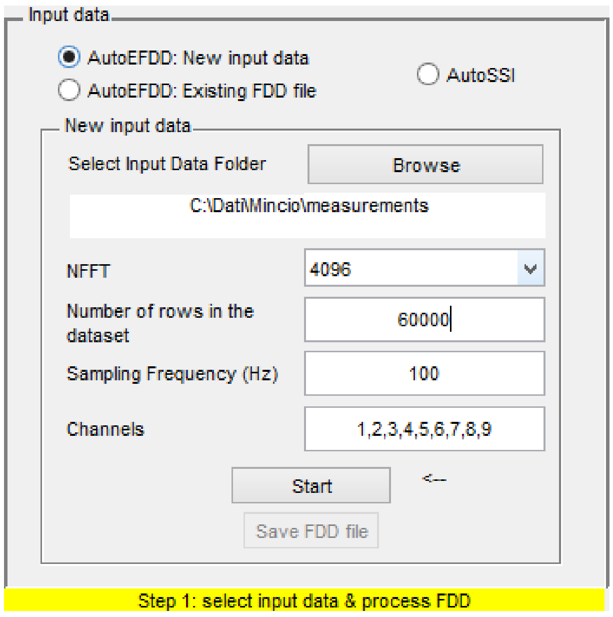


Figure 4.46: Schematic summary of the positioning of the instrumentation

Only 15 channels were available to perform the dynamic analysis: channel 20, 21, 24, 28, 29, 32, 36, 37, 40, 44, 45, 48, 52, 53, 56.

4.6.5 Tutorial: identification of the structure using AtOMA - AutoEFDD

Selection of input data Once *AtOMA* is launched, the check-box *AutoEFDD: New input data* was selected. Then, in order to choose the folder where the measurement files were located the *Browse* button was pressed.



Input data

(●) AutoEFDD: New input data
( ) AutoEFDD: Existing FDD file
( ) AutoSSI

New input data

Select Input Data Folder	Browse
	C:\Dati\Mincio\measurements
NFFT	4096
Number of rows in the dataset	60000
Sampling Frequency (Hz)	100
Channels	1,2,3,4,5,6,7,8,9

Start <--

Save FDD file

Step 1: select input data & process FDD

Figure 4.47: AutoEFDD: select input data

The parameters are chosen as displayed in Figure 4.47. Note that not all the channels available in the files were selected, since channels from 10 to 15 were excluded from the analysis.

The button *Start* is pressed in order to perform the first step of the FDD analysis (calculation of the *cpsd* matrices and *singular value decomposition*). A progress bar will appear that allows you to watch the progress of the calculations (Figure 4.48).

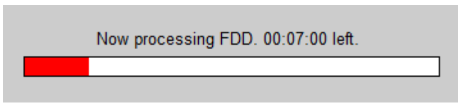
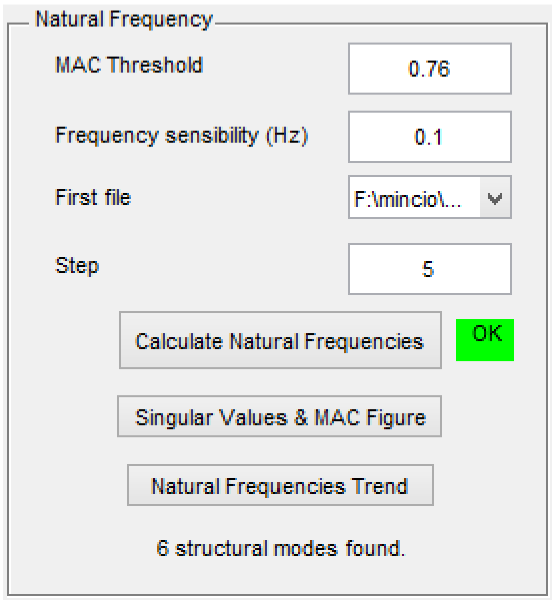

Now processing FDD. 00:07:00 left.

Figure 4.48: Progress bar.

Identification of the structural modes Once the calculation is completed, it is required to perform the natural frequencies identification. The parameters are set as displayed in Figure 4.49


Natural Frequency

MAC Threshold	0.76
Frequency sensibility (Hz)	0.1
First file	F:\mincio\...
Step	5

Calculate Natural Frequencies
OK
Singular Values & MAC Figure
Natural Frequencies Trend

6 structural modes found.

Figure 4.49: Natural frequencies identification.

A progress bar will inform you on the development of identification (Figure 4.50).

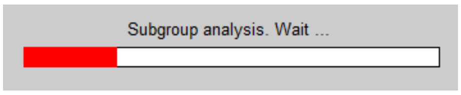
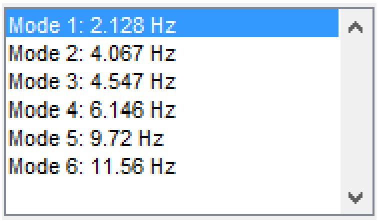

Subgroup analysis. Wait ...

Figure 4.50: Progress bar.

The mean values of the results of the frequency identification are displayed in the text field in the Results Panel (Figure 4.51).



Mode 1: 2.128 Hz
Mode 2: 4.067 Hz
Mode 3: 4.547 Hz
Mode 4: 6.146 Hz
Mode 5: 9.72 Hz
Mode 6: 11.56 Hz

Figure 4.51: Mean values of the natural frequencies.

The development over time of the natural frequencies is displayed by pressing the *Natural Frequencies Trend* button (Figure 4.52).

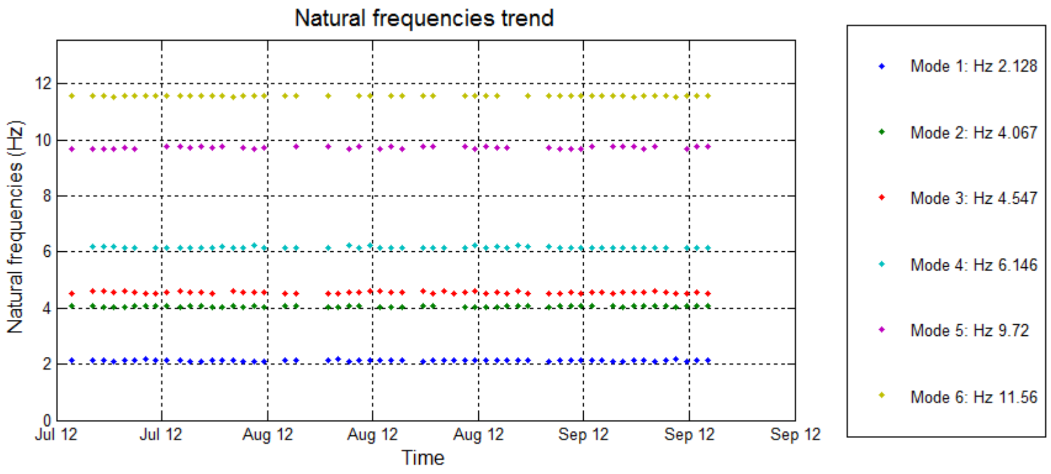


Figure 4.52: Development of natural frequencies.

The auto-spectrum function is displayed by pressing the *Singular values & MAC Figure* button. In Figure 4.53 the peaks corresponding to identified natural frequencies have been highlighted.

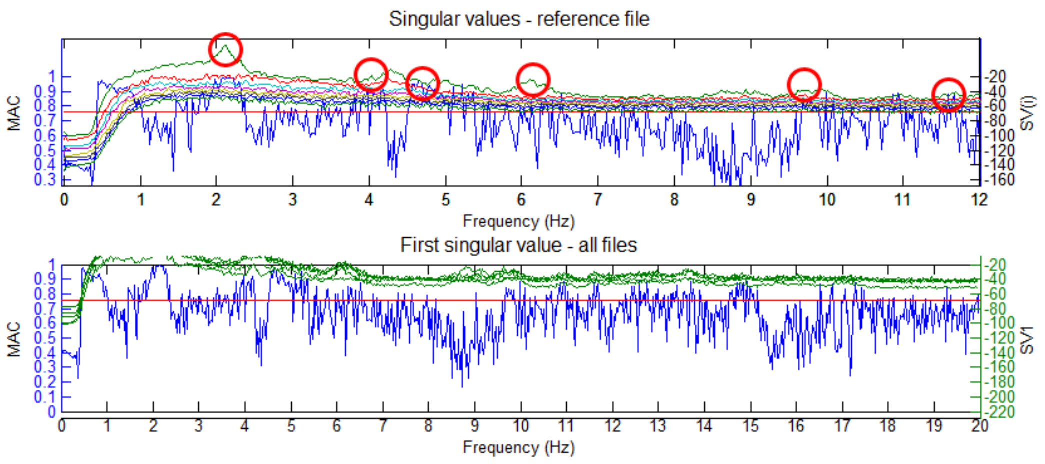


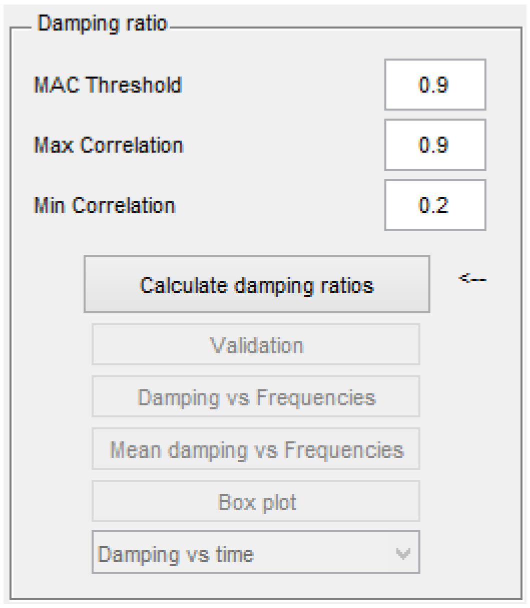
Figure 4.53: Singular values & MAC Figure.

The results are summarized in Table 4.2. It can be seen that the identification has been successfully held for almost every measurement file. The low standard deviation indicates that the frequencies are poorly dispersed.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
f_{max}	2,173	4,077	4,590	6,177	9,741	11,572
f_{min}	2,100	4,028	4,517	6,128	9,692	11,523
f_{mean}	2,130	4,064	4,548	6,139	9,719	11,561
std.dev	0,021	0,018	0,029	0,015	0,022	0,016
success%	96,4%	92,7%	98,2%	74,5%	76,4%	87,3%

Table 4.2: Frequency results of the monitoring using AutoEFDD.

Extraction of damping ratio Then, once the identification of the structural modes is complete, damping ratio parameters must be calculated. First the parameters (*MAC Threshold, Max Correlation and Min Correlation*) are configured as shown in Figure 4.54.


Damping ratio

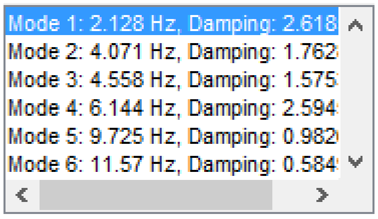
MAC Threshold 0.9

Max Correlation 0.9

Min Correlation 0.2

Figure 4.54: Damping ratio calculation.

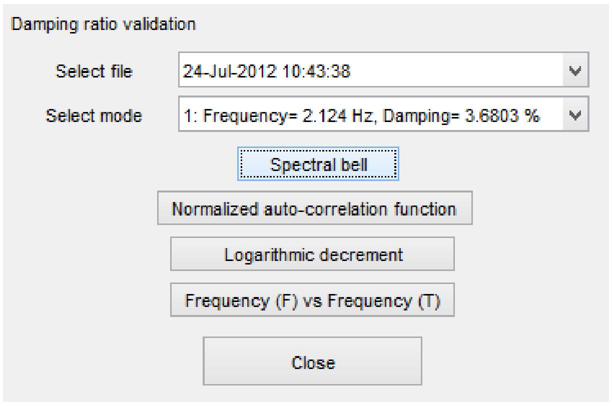
By pressing the *Calculate Damping Ratio* button the damping ratio parameters are extracted and their mean values are displayed in the text field in the Results panel (Figure 4.56).



Mode 1: 2.128 Hz, Damping: 2.618
Mode 2: 4.071 Hz, Damping: 1.762
Mode 3: 4.558 Hz, Damping: 1.575
Mode 4: 6.144 Hz, Damping: 2.594
Mode 5: 9.725 Hz, Damping: 0.982
Mode 6: 11.57 Hz, Damping: 0.584

Figure 4.55: Mean values of the damping ratios.

Validation of the results It is possible to launch the validation panel by pressing the *Validate* button. From this panel the user is allowed to observe the results obtained for each file and each structural mode. The tasks available through this panel are listed in Section 4.3.2.



Damping ratio validation

Select file: 24-Jul-2012 10:43:38

Select mode: 1: Frequency= 2.124 Hz, Damping= 3.6803 %

Spectral bell
Normalized auto-correlation function
Logarithmic decrement
Frequency (F) vs Frequency (T)
Close

Figure 4.56: Validation of the damping ratios.

The graphs for the first mode (2,124 Hz) of the file of the 24th July are displayed in the following figures.
In Figure 4.57 is displayed the SDoF Auto-Spectral function for the first structural mode. The selected frequency function shows good characteristics, as it is well defined and there is no noticeable noise.

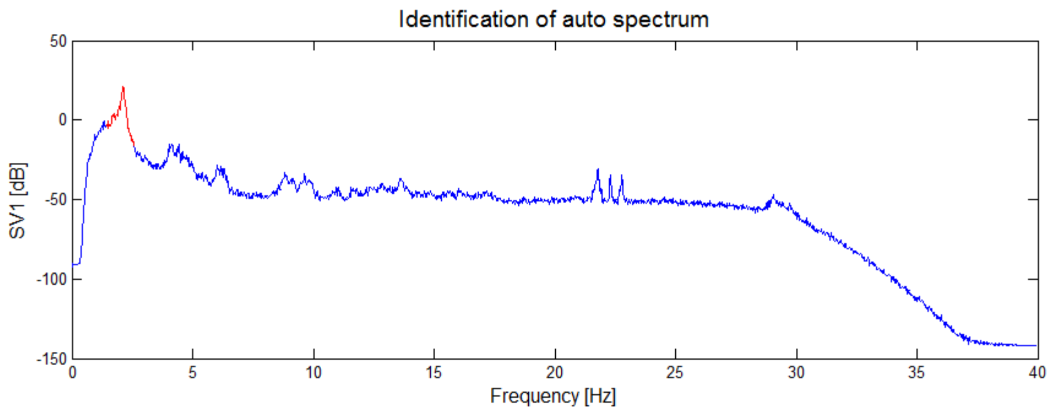


Figure 4.57: SDoF Auto-Spectral function.

In Figure 4.57 the normalized auto-correlation function for the first structural mode is displayed. The behavior exhibited by the function is that of a typical structural decay function and does not appear distinctive of a harmonic excitation.

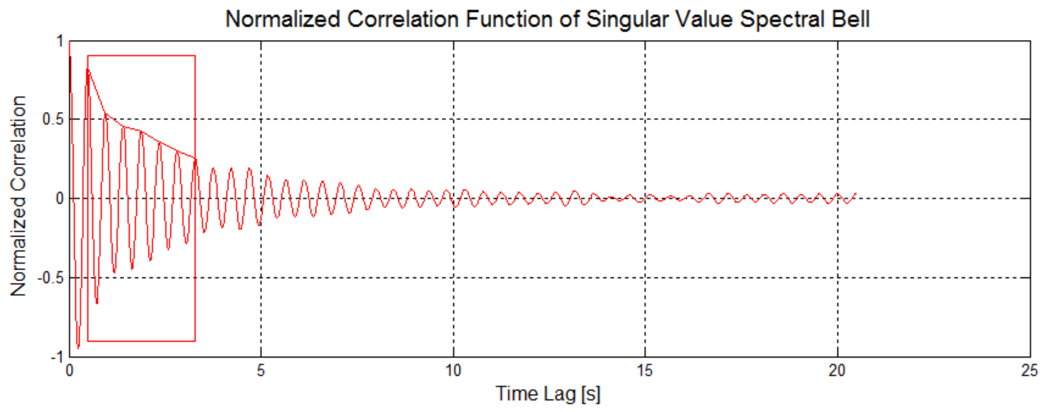


Figure 4.58: Normalized auto-correlation function.

The logarithmic decrement shown in Figure 4.59 detaches itself, especially in the initial portion, from a linear trend.

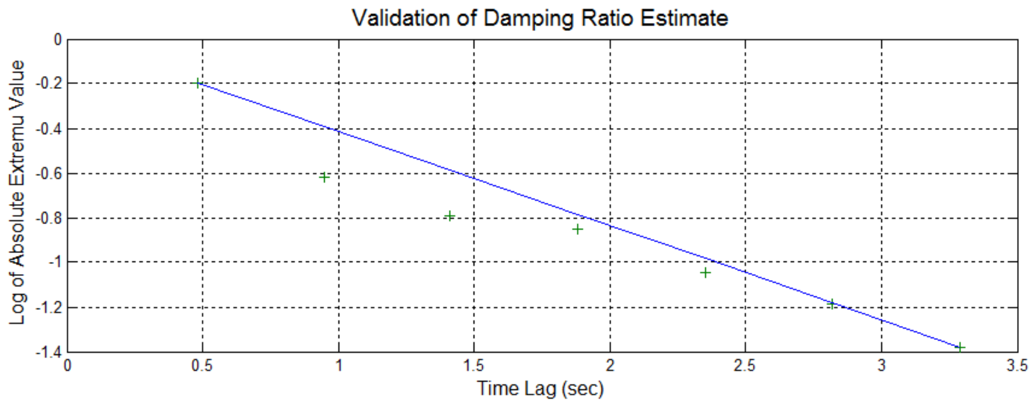


Figure 4.59: Logarithmic decrement.

Therefore a lower value for the parameters *Max Correlation and Min Correlation* is chosen: *Max Correlation=0,6* and *Min Correlation=0.1*. Calculation of the damping ratios is performed again by clicking the *Calculate damping ratios* button. The normalized auto-correlation function and the corresponding logarithmic decrement are shown in Figure 4.60.

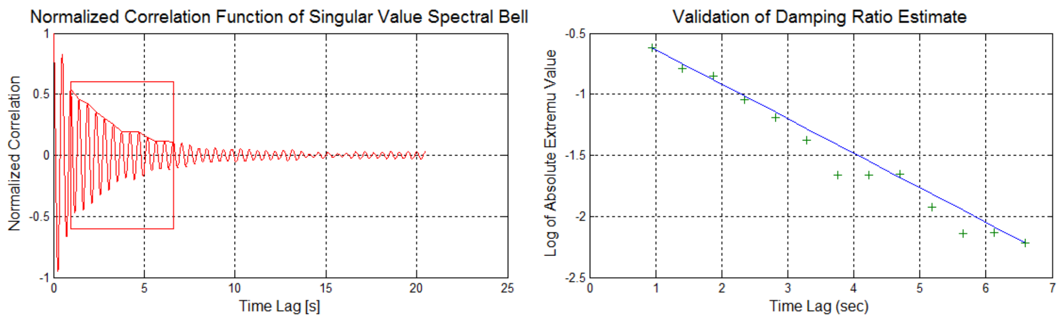


Figure 4.60: Normalized auto-correlation function and logarithmic decrement with new parameters.

Figure 4.61 displays the correlation between the natural frequencies calculated using the EFDD analysis in frequency domain and those calculated, again using EFDD, in time domain. The values thus calculated are practically equal to each other.

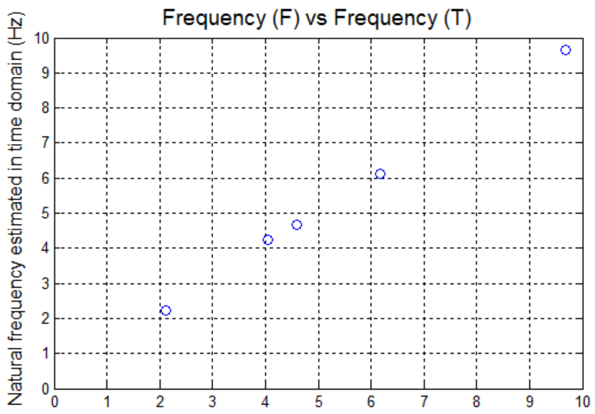


Figure 4.61: Frequency (F) vs Frequency (T).

Damping ratio results Once the validation is complete, results of the calculation can be displayed by pressing the buttons *Damping vs Frequency (Figure 4.62), Mean damping vs Frequencies, Box plot (Figure 4.63)* and *Damping vs time (Figures 4.64, 4.65 and 4.66).*

In Figure 4.62 the extracted modal parameters are displayed for all the files. Mode 1 and 4 are the structural modes that have allowed a better identification of the damping ratio, although presenting some dispersion in the results.

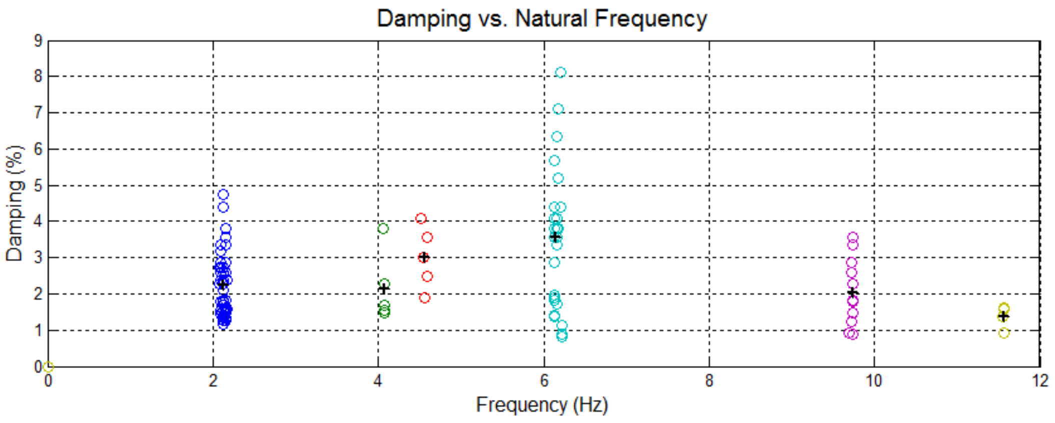


Figure 4.62: Damping vs frequency figure.

The box-plot description shown in Figure 4.63 is actually useful only for mode 1, 4 and 5 as they are the only modes to possess a sufficient success rate.

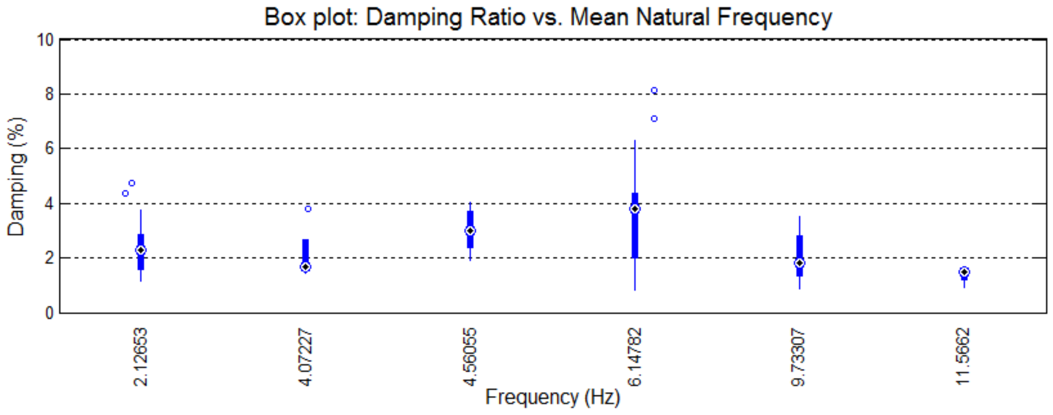


Figure 4.63: Box plot description of damping ratio results.

In Figures 4.64, 4.65, 4.66 are shown the extracted damping ratio values for each structural mode over time. It is clear that modes 2, 3 and 6 do not have a sufficient number of results and the correspondent damping ratio can not be used, at this stage, for model updating operations.

Since the time span for the monitoring was too narrow, it was not possible to observe any type of seasonal fluctuation of the results.

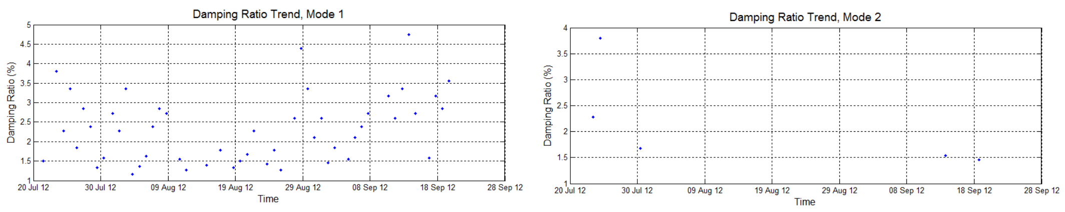


Figure 4.64: Trend over time of damping ratios for mode 1, 2.

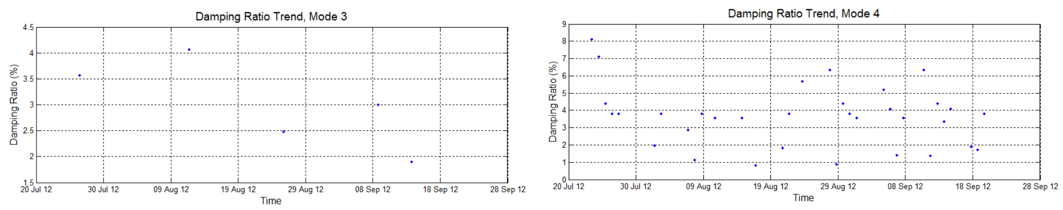


Figure 4.65: Trend over time of damping ratios for mode 3, 4.

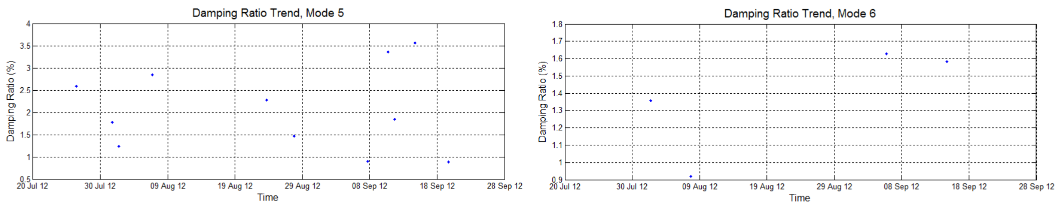


Figure 4.66: Trend over time of damping ratios for mode 5, 6.

It can be seen from Table 4.3 that damping ratio results for this structure are affected by a high level of uncertainty, which is confirmed by the low success rate for modes 2, 3, 5, 6 and the high dispersion of the results especially for mode 4. Therefore it is required to perform a validation of the results using AutoSSI method.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
ζ_{max}	4,747%	3,799%	4,070%	7,111%	3,562%	1,629%
ζ_{min}	1,164%	1,462%	1,901%	0,827%	0,891%	0,920%
ζ_{mean}	2,314%	2,152%	3,003%	3,500%	2,069%	1,373%
std.dev	0,864%	0,975%	0,858%	1,608%	0,935%	0,324%
success%	89,1%	9,1%	9,1%	58,2%	20,0%	7,3%

Table 4.3: Damping results of the monitoring using AutoEFDD.

Saving the template If the parameters used to configure the analysis (*NFFT, Channels, MAC threshold for frequency, Frequency sensibility, MAC Threshold for damping, Max and Min Correlation*) are considered to be the ones that provide the best results for the structure, then it is possible to save a *Template* file. The template file contains information about the position of the modal domains of the structure, therefore, subsequent analysis are carried out to confirm the presence of a mode of vibration around the modal domain previously defined.

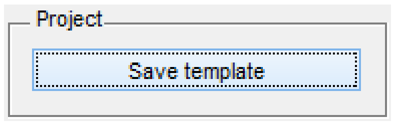


Figure 4.67: Save template button.

To save a template file, press the *Save template file* button. A dialog requires the user to specify some details of the analysis. By pressing OK the template is automatically saved inside the *Template* folder of the program. The name of the template file is automatically defined by the name of the template set by the user: if the template is called *Mincio OK*, the file name will be $Template\_Mincio\_OK.efdd$.

Save results To save the results of the analysis, press the *Save results* button.

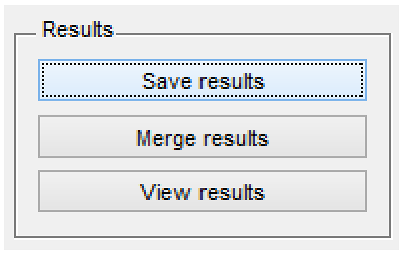
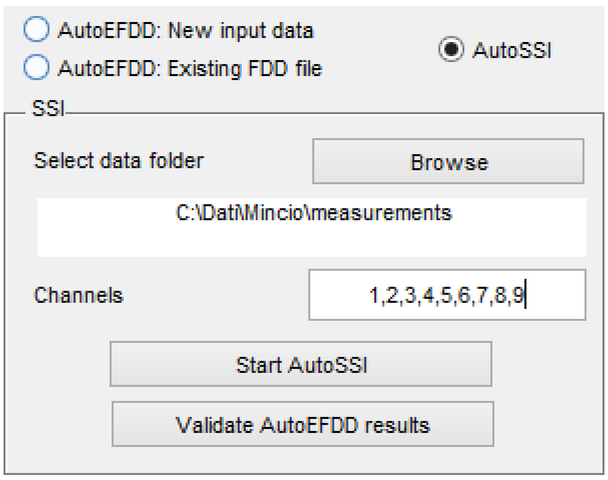


Figure 4.68: Save results button.

If a template file was previously saved, the program automatically saves the results file inside the Results folder corresponding to the template. Otherwise, a dialog asks the user to specify the folder where the file will be saved.

Analysis using AutoSSI To perform a SSI analysis, press *Browse* to select the folder containing the files and set the channels to use in the analysis. For the current analysis, the chosen channels are *1,2,3,4,5,6,7,8,9* as shown in Figure 4.69.



AutoSSI [✓]

Select data folder: C:\Dati\Mincio\measurements

Channels: 1,2,3,4,5,6,7,8,9

Figure 4.69: Analysis using SSI.

Then, press *Start SSI* in order to launch the analysis. Note that AutoSSI procedure requires requires much more time and processing power higher than AutoEFDD.
Once the analysis is finished, the user is asked to save the SSI results file.

The results given by AutoSSI procedure are displayed in Tables 4.4 and 4.5. AutoSSI analysis could not identify the 6th structural mode at 11,56 Hz, probably due to the calibration of the internal parameters of the model.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
f_{max}	2,165	4,279	4,710	6,111	9,751
f_{min}	2,098	4,199	4,310	5,988	9,735
f_{mean}	2,126	4,235	4,553	6,057	9,743
$std.dev$	0,021	0,027	0,132	0,047	0,011
$success\%$	100,0%	72,7%	90,9%	54,5%	18,2%

Table 4.4: Frequency results of the monitoring using AutoSSI.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
ζ_{max}	3,551%	3,613%	7,329%	4,252%	3,734%
ζ_{min}	1,451%	1,692%	2,464%	2,496%	2,606%
ζ_{mean}	2,755%	2,700%	4,590%	3,253%	3,170%
$std.dev$	0,723%	0,596%	1,474%	0,700%	0,798%
$success\%$	100,0%	72,7%	90,9%	54,5%	18,2%

Table 4.5: Damping results of the monitoring using AutoSSI.

The reason why these results are of interest resides in the possibility to compare them with those extracted by the method AutoEFDD. By pressing the *Validate AutoEFDD results* button in the SSI panel, it is possible to compare the results obtained with the two different procedures. The program asks the user to select the AutoEFDD and AutoSSI results files to compare. Then Tables 4.6 and 4.7 are displayed, which contain the results for both AutoEFDD and AutoSSI, including the relative error between the results.
In Table 4.6 are shown the natural frequencies extracted using the two procedures. The results are very close, except for the mode 6 which is not identified by the method AutoSSI.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
$f(EFDD)$	2.128	4.067	4.547	6.146	9.720	11.564
$Success\%(EFDD)$	100%	92.7%	98.2%	90.9%	76.4%	87.3%
$f(SSI)$	2.126	4.230	4.554	6.057	9.743	-
$Success\%(SSI)$	100%	81.8%	90.9%	54.5%	18.2%	-
$Error\ \epsilon$	0.1%	3.9%	0.1%	1.5%	0.2%	-

Table 4.6: Comparision of the natural frequencies extracted using AutoEFDD and AutoSSI.

In Table 4.7 are displayed the damping ratios obtained with the two procedures. AutoSSI presents a higher success rate than AutoEFDD for modes 2, 3, 5. The average results show an error ranging from 9% to 44%. These errors are considerable acceptable, given the structural differences of the models. The order of magnitude of the parameters is in fact confirmed, and these values could be used for the calibration of a FE model.

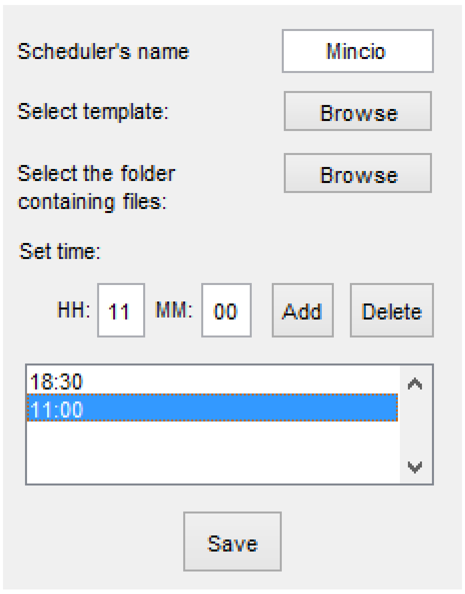
	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
$\zeta(EFDD)$	2.236%	2.152%	3.003%	3.545%	2.034%	1.373%
$Success\%(EFDD)$	89.1%	9.1%	9.1%	60.0%	20.0%	7.3%
$\zeta(SSI)$	2.755%	2.700%	4.590%	3.253%	3.170%	-
$Success\%(SSI)$	100%	72.7%	90.9%	54.5%	18.2%	-
$Error\ \epsilon$	20.8%	22.6%	41.8%	8.6%	43.7%	-

Table 4.7: Comparision of the damping ratios extracted using AutoEFDD and AutoSSI.

Overall, it can be said that the identification made by using the AutoEFDD is validated.

New AutoEFDD analysis using template To start an analysis based on a previous one, select from the drop-down menu *File → New from template.* Then, select the template file previously saved (for example $Template\_Mincio\_OK.efdd$).
In the *Input data* panel, select the check-box *AutoEFDD: New input data* and press *Browse* to select the folder containing the measurement files, then press *Start.* The program will automatically process all the steps of the analysis, since no user input is required. When the calculation of the damping ratios is completed, the program saves the results inside the template's *Results* folder (in this case, *Results/Mincio*$_O$*k/* and *merges* the new results file with the other files in the folder (if any). The merged results file is called *current_sequence.mat* and it can be opened using the *View results* button.

Schedule functions To set up the automatic execution of the template-driven analysis, select *Schedule → New Schedule* from the drop-down menu. The panel shown in figure 4.70 displays a typical configuration of the schedule.

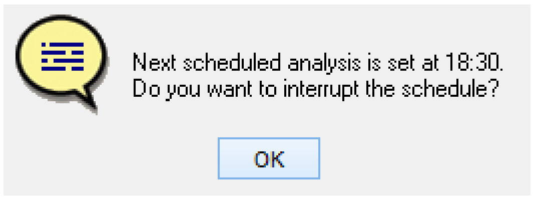


Scheduler's name	Mincio
Select template:	Browse
Select the folder containing files:	Browse
Set time:	
HH: 11 MM: 00	Add Delete

18:30
11:00

Figure 4.70: New Schedule.

In this case, the chosen template is $Template\_Mincio\_OK.efdd$, and the folder is the one in which the program that handles the measurements of the structure puts new files. Since new files are usually stored at 10:45 and 17:45, the analysis is programmed to automatically start at 11:00 and 18:30.
Pressing *Save*, the scheduled task is saved as a configuration file inside the *schedule folder*, whose name is $sch\_Mincio.schedule$.
In order to enable the scheduler, select from the drop-down menu *Schedule* → *Schedule launcher*. The user is asked to choose the configuration file, in this case $sch\_Mincio.schedule$. A dialog displays the next scheduled analysis.



Next scheduled analysis is set at 18:30.
Do you want to interrupt the schedule?

Figure 4.71: Schedule launcher.

Pressing *OK*, the scheduler is disabled.
When the specified time is reached, the analysis is performed automatically without user intervention, and once completed, the system gets back waiting for the next run.

Conclusions With the perspective of the program user, an analysis of the dynamic identification of the bridge over the Mincio was developed, demonstrating the reliability of the results obtained by the procedure AutoEFDD by the procedure AutoSSI also available as a ATOMA.
At the current state, it is necessary to rely on other software in to correlate the modal shapes of the FE model with those identified with modal testing. For other expected future developments for the program, look at 5.6.

Chapter 5

Program code

5.1 Introduction

In this chapter the code that constitutes the program is described. **AtOMA** is a program developed using *Matlab*® environment. A Graphical User Interface (GUI) is composed by various graphical objects, usable by the user. Each component reacts to the user's interaction, by triggering the activation of a corresponding function, which in a graphical interface context, is called *Callback function*. Each callback function contains a set of commands aimed to perform specific tasks.
The first part of the implementation of the AutoEFDD method, consisting in the identification of the natural frequencies, was implemented by Piovesan D. in [36]. The SSI method was implemented by Islami K. in [24]. The AutoEFDD method was improved by enabling the calculation of the damping ratio.
In this chapter, initially a detailed description of the code that leads to the calculation of the damping ratio for the AutoEFDD procedure will be carried out. Subsequently, the global behaviour of the program will be addressed and the purpose of the main callback functions will be described.
We will proceed in the conceptual order associated with the use of the program:

1. The functions called by the input selection panel's objects;

2. The functions called by the identification panel's objects;

3. The functions called by the result management panel's objects;
4. The functions called by the menu elements.

Many application variables used across the entire program are defined using the *setappdata* function. The most important ones are defined below, in order to help the comprehension of the code.

- *usingtemplate*: if it is set to 1, then a template has been loaded and thus, the process is template-driven. In this case a user intervention is partially required.
- *scheduleron*: if it is set to 1, then a scheduled task is active and the user intervention is not required.
- *FDD_location* contains the path for the *FDD.mat* file used in the analysis.
- *mergetemplate*: if it is set to 1, then the process is template-driven and the results files contained in the default results folder are automatically merged once the analysis is finished.
- *Result_frequency* contains the variable of the calculated natural frequencies.
- *Result_damping* contains the variable of the calculated damping ratios.

5.1.1 Graphical User Interfaces using Matlab®

What is a GUI? A graphical user interface is a graphical display in one or more windows containing controls, called *components* that enable a user to perform interactive tasks.
GUI components can include menus, tool-bars, push buttons, radio buttons, list boxes, and sliders. Usually a GUI waits for the user to manipulate a control, and subsequently responds to each action. Each control can, usually by means of a particular user interaction, launch a user-written sub-routine, which is called *Callback function*.

This kind of programming is often referred to as *event-driven* programming, since it requires specific situations to happen in order to trigger a specific action.

A GUI is actually a collections of graphical components and functions. Graphical components are designed using a specific tool available for Matlab® called *GUIDE. Callback functions* are actually like normal Matlab functions. What actually distinguishes GUI programming to command programming is that the callback execution is asynchronous, that is, it is triggered by events external to the software. This means that there is no default path for the execution of the program, and the data must be exchanged between the functions and interface components when they are needed, asynchronously. Data management is therefore a key point when programming a GUI.

Data management with a GUI The user is often required to change the status of some objects, i.e. to insert some text or a numerical value inside a text field, or to select a check-box and so on. This status is usually required to perform the execution of a Callback. Inside the callback, this particular value needs to be read. The callback function must invoke a special variable called *handles.* The handles variable is an input variable required by default for every callback function (it might be excluded in case it's not necessary). Its name may actually be misleading. A handle has in fact the purpose of providing access to properties of an object, while *handles* variable is precisely a structure of multiple handles. By invoking the *handles* variable, the user is able to read and write the properties of every single component of the interface, by using *get* and *set* function. Each component is in fact defined by a set of properties. Let's assume that a text field is called *text1* (it means that its property *tag* is equal to *text1*). When a user writes something inside *text1*, he is, in fact, changing the status of its *String* parameter. For callback function, the content of the text field it is not readily available as a global variable. It can be be read using *handles*: *MyString=get(handles.text1,'String');* saves the value of the *String* property of the component called *text1* inside the variable *MyString*, that can now be used by the callback function.
It is also possible for a callback to write a new string inside *text1* by using the

set command: *set(handles.text1,'String','This is my new string');*. What has been said about the *String* property of the component is applicable for every other property.
handles variable can also be used for passing a variable between callbacks. The command *handles.NewVar=content;* saves a new variable inside the *handles* structure, which can be invoked from another function with the command *content=handles.NewVar;* .
It is important, however, to make an observation about the nature of handles. As it was introduced, variable *handles* is passed as an input argument for the callback. This means that what is being used inside the function is actually only an *instance*, a *photograph* of the real structure of handles taken at the moment in which the callback is invoked. This is important if *handles* is used to pass a variable between callbacks, in fact, if a new variable is saved in *handles*, it is actually stored only inside a *copy* of *handles* and will be lost after the function is finished. Therefore *handles* must be updated after the change, using the command *guidata*.

Another way of storing variables so that they are globally available among the different callbacks of the program is by using *setappdata* and *getappdata* functions.

Opening functions The opening function is the first callback in every GUI M-file. It performs tasks that need to be done before the user has access to the GUI. For example, the opening function of *AutoEFDDgui* (the main interface) is used to initialize the application variables *usingtemplate* and *scheduleron* to zero, which respectively determine if a template is used and if the scheduler is on, and to disable the identification panel which is not yet available for user interaction.

```
function AutoEFDDgui_OpeningFcn(hObject, eventdata, handles,
    varargin)
guidata(hObject, handles);
setappdata(0,'usingtemplate',0);
setappdata(0,'scheduleron',0);
set(findall(handles.uipanel7,'-property','enable'),'enable','off')
```

```
    ;
```

Listing 5.1: Opening function

5.2 Input selection panel

The check-boxes at the top of the panel are used to select the required input panel. Each radio object is characterized by a *tag*, which identifies the Callback's name.

AutoEFDD: New input data this checkbox enables the panel for the selection of new input data, setting *newdatapanel* as visible and the other input panels (*ssipanel* and *existingdatapanel*) as not-visible.

AutoEFDD: Existing FDD file, AutoSSI these check-boxes perform the same operations of the first check-box, relatively to each own panel.

AutoEFDD: new input data panel

Once the *AutoEFDD: new input data* check-box is pressed, the corresponding panel becomes visible.
The *Browse* button allows the selection of the folder containing the measurements files, and ultimately displays it in the text field situated below it.

```
function browsenewfdd_Callback(hObject, eventdata, handles)
folder_name = uigetdir('','Please select the folder with the data
    file');
handles.fileList = getAllFiles(folder_name, 500);
guidata(hObject,handles);
set(handles.folder,'String',folder_name);
...
```

Listing 5.2: AutoEFDD: Browse button

The *Start* button begins the FDD procedure.
First the parameters specified by the user (*L, fs, NFFT, channels*), and the

path of the directory containing the files, are read using the *get* function. Subsequently *cpsd* and *svd* matrices are calculated (for more details about this portion of code, look at [36]).

```
function [handles]=startfdd_Callback(hObject, eventdata, handles)
%Reading of the parameters.
canali=get(handles.channels,'String');
canali_ind=strfind(canali,',');
v(1)=str2num(canali(1:canali_ind(1)-1));
for i=2:length(canali_ind)
    v(i)=str2num(canali(canali_ind(i-1)+1:canali_ind(i)-1));
end
v(i+1)=str2num(canali(canali_ind(i)+1:length(canali)));
...
```

Listing 5.3: AutoEFDD: Start FDD button

After the *Singular Value Decomposition* is performed for all of the measurements files contained in the directory, the objects inside the identification panel are enabled for user interaction.
Processed data is then saved inside a *FDD.mat* file.
If the process is driven by a scheduled event (*if getappdata(0,'scheduleron')==1*), the last part of the next block is used to move the files to a subfolder called *archive*. If the process is driven by a template, subsequent callbacks (*frequencies_Callback, damping_Callback, saveres_Callback and merge*) are launched.

```
...
close(h);
hhh=handles;
clear handles hObject;
save FDD;
load (getappdata(0,'FDD_location'),'FastNameList');
set(findobj('Tag','ifast0'),'String',FastNameList,'Value',1);

if getappdata(0,'usingtemplate')==1
    if getappdata(0,'scheduleron')==1 %spostamento files
        archive=[getappdata(0,'fastdir') '\archive\'];
        if ~exist(archive,'file') mkdir(archive); end
        for i=1:length(FastNameList)    movefile(FastNameList{i},
   archive,'f'); end
```

```
    end
    handles=frequencies_Callback([],[], hhh);
    handles=damping_Callback([],[], handles);
    saveres_Callback([],[], handles);
    setappdata(0,'mergetemplate',1);
    merge;
end
```

Listing 5.4: AutoEFDD: Start FDD button - 2

The *Save FDD file* button allow the user to copy the *FDD.mat* file to any directory.

```
function savefddfile_Callback(hObject, eventdata, handles)
[FileName,PathName] = uiputfile('FDD_*.mat');
copyfile(getappdata(0,'FDD_location'),[PathName FileName],'f');
```

Listing 5.5: AutoEFDD: Save FDD button

AutoEFDD: Existing FDD file

The *Browse* button allows the user to select a valid *FDD.mat* file, created in a previous analysis. The path of the file is displayed in the text field below the button.

```
function browsefdd_Callback(hObject, eventdata, handles)
[FileName,PathName,FilterIndex] = uigetfile('*.mat','Please select
    the FDD.mat file.');
fdd_name=[PathName FileName];
set(handles.fddfile,'String',fdd_name);
...
```

Listing 5.6: AutoEFDD: Browse existing FDD button

Once the FDD file is selected, by pressing the *Load FDD file* button, the application variable data *FDD_location* is set by using the *setappdata* function. If the analysis is performed on the basis of a template, this button will automatically launch the subsequent callbacks (*frequencies_Callback, damping_Callback, saveres_Callback and merge*).

```
function loadfdd_Callback(hObject, eventdata, handles)
setappdata(0,'FDD_location',get(handles.fddfile,'String'));
FDD=getappdata(0,'FDD_location');
...
load (getappdata(0,'FDD_location'),'FastList');
FL=struct2cell(FastList);
set(findobj('Tag','ifast0'),'String',FL,'Value',1);
if getappdata(0,'usingtemplate')==1
    handles=frequencies_Callback([],[], handles);
    handles=damping_Callback([],hObject, handles);
    saveres_Callback([],[], handles);
    setappdata(0,'mergetemplate',1);
    merge;
end
```

Listing 5.7: AutoEFDD: Load existing FDD button

AutoSSI

The *Browse* button in the AutoSSI panel leads to the selection of the folder containing the measurement files. The path of the directory is then displayed inside the text field below the button.

```
function ssibrowse_Callback(hObject, eventdata, handles)
folder_name = uigetdir('','Select data folder');
if folder_name(1)~=0
    set(handles.ssidir,'String',folder_name);
    set(handles.ssistart,'Enable','on');
end
```

Listing 5.8: AutoSSI: Browse button

The *Start SSI* button reads the path of the directory written inside the text field and executes the SSI analysis. For more informations about the AutoSSI procedure, look at [24].
The results are saved in the same format as the ones obtained from the AutoEFDD procedure.

```
function ssistart_Callback(hObject, eventdata, handles)
```

```
folder_name=get(handles.ssidir,'String');
...
    canali=get(handles.channelsSSI,'String');
    canali_ind=strfind(canali,',');
    v(1)=str2num(canali(1:canali_ind(1)-1));
    for i=2:length(canali_ind)
        v(i)=str2num(canali(canali_ind(i-1)+1:canali_ind(i)-1));
    end
    v(i+1)=str2num(canali(canali_ind(i)+1:length(canali)));
    Signals = load([folder_name '\' 'Temp.txt']);
    TN_Channel = size(Signals,2);
    Signals(:,1) = [];
    Signals = Signals(:,v);
...
%SSI
...
    Result_frequency(ifast,:) =[kn appr(1:12,1)'];
    Result_damping(ifast,:) =[kn  (appr(1:12,2)')./100];
...
if ~exist('..\Results','file')
    mkdir('..\Results');
end
[file,path] = uiputfile('..\Results\SSI_Result_','Save SSI Results
     File.');
filessi=[path file];
save(filessi,'Result_frequency', 'Result_damping');
setappdata(0,'CurrentResults',filessi);
viewresults;
```

Listing 5.9: AutoSSI: Start SSI button

5.3 Identification panel

The identification panel serves, exclusively for the AutoEFDD procedure, to identify natural frequencies and damping ratios.

5.3.1 Natural frequencies identification panel

The *Calculate Natural Frequencies* button performs the natural frequencies identification according to the parameters specified by the user.

For detailed informations about the identification of the natural frequencies, look at [36].

The beginning of the callback consists in loading the *FDD* file and reading the parameters. If the process is driven by a template, the first part is bypassed because the modal domains are already defined.

```
function [handles] = frequencies_Callback(hObject, eventdata,
    handles)
set(handles.statusF,'String','Wait..','BackgroundColor',[1 0 0]);
FDD=getappdata(0,'FDD_location');
load(FDD);
gg=2;
templ=getappdata(0,'usingtemplate');
if  templ == 0
  pp  = 1;
else
  pp  = -1;
end
if pp==1
   ...
```

Listing 5.10: Calculate natural frequencies button

The parameter N is given by the *Frequency sensibility* transformed into a number of frequency points:

$$N = \frac{NFFT}{f_s \cdot frequency_{sensibility}} \tag{5.3.1}$$

```
      if ss==3
         mac      = str2double(get(handles.mac,'String'));
         N        = round(NFFT/fs*str2double(get(handles.n,'String'))
    );
```

Listing 5.11: Calculate natural frequencies button - mac and N parameters

After the modal domains are found [36], the natural frequencies are identified for each file. If the process is template-driven, then the first part was skipped and the variables that define the modal domains are loaded from the template.

```
...

clearvars -except handles

if getappdata(0,'usingtemplate')==1
    load (getappdata(0,'savefile'));
    load (getappdata(0,'indici'));
else
    load savefile
    load indici
end
    load(getappdata(0,'FDD_location'));
...
```

Listing 5.12: Calculate natural frequencies button - template-driven procedure

The natural frequencies are then calculated and saved in the variable *newf2*. The file *PERDAMPING.mat* contains the data needed for damping ratio calculation.
The frequency results are averaged excluding the outliers and displayed inside the Results panel in order to provide a quick view of the results.

```
...
    save PERDAMPING SS0 Umodi NFFT fs newf2 f1
    FDD=getappdata(0,'FDD_location');
    load(FDD, 'Result');
    for i=1:ifast2
        for j=1:size(INDICI,2)
            Result(i,j+1)=newf2(i,j);
        end
    end
    Result(~Result)=nan;
    modelist=findobj('Tag','modelist');
    for i=2:size(Result,2)
      listSTR{i-1}=['Mode ' num2str(i-1) ': ' num2str(trimmean(
    Result(:,i),20),4) ' Hz'];
    end
    set(modelist,'String',listSTR);
...
```

Listing 5.13: Calculate natural frequencies button - data saving

The *Singular values & MAC figure* provides a plot of the Singular Values of the Cross Power Spectral Density functions of the selected files.
At first, the data are prepared for plotting.

```
function svbutton_Callback(hObject, eventdata, handles)
    FDD=getappdata(0,'FDD_location');
    load(FDD);

    ifast0  = get(handles.ifast0,'Value');
    ifast   = str2double(get(handles.ifast,'String'));
    ifast   = ifast0+ifast-1;
    fmTOT   = zeros(floor(size(FastList)/(ifast-ifast0+1)),50);
    appAX   = zeros(50,300,floor(size(FastList)/(ifast-ifast0+1)));
    ZZ      = appAX;
    YY      = appAX;
    indiciTOT=zeros(floor(size(FastList)/(ifast-ifast0+1)),50);

    UUrif(:,:)=Umodi(:,:,ifast0);
    for j=ifast0+1:ifast
        for i=1:NFFT/2+1
            UUrif2=UUrif(i,:)*UUrif(i,:)';
            P=(UUrif(i,:)*Umodi(i,:,j)').^2;
            ui=Umodi(i,:,j)*Umodi(i,:,j)';
            MACuu(j-ifast0,i)=P/(UUrif2*ui);
        end
    end
    STD=std(MACuu);
    miniSTD=STD(1:0.4*NFFT);
    mSTD=sum(miniSTD)/(0.4*NFFT);
    for n=1:NFFT/2+1
        MACuuMEDIO(n)=sum(MACuu(:,n)')/(ifast-ifast0);
    end
```

Listing 5.14: Singular Values & MAC Figure Button

The plotting of the figures is subsequently developed using the *subplot* function, which allows to place multiple graphs within the same window.

```
    x=0:NFFT;
    y=x-x+str2double(get(handles.mac,'String'));

    figure1=figure('Color',[1 1 1]);
    set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
    movegui(figure1, 'center');

            subplot(2,1,1);
            [AX,H1,H2]=plotyy(f1(1:0.4*NFFT+1),MACuuMEDIO(1:0.4*NFFT
   +1), f1(1:0.4*NFFT+1),SS1(1:0.4*NFFT+1,:,ifast),'plot'); hold
   on
...
            plot(x,y,'-r','linewidth',1);
            set(get(gca,'Xlabel'),'String','Frequency (Hz)','Color','
   k');
            title('\fontsize{12}Singular values - reference file');
            subplot(2,1,2);
      for i=ifast0:ifast
            [AX,H1,H2]=plotyy(f1(1:0.4*NFFT+1),MACuuMEDIO(1:0.4*NFFT
   +1),f1(1:0.4*NFFT+1),SS1(1:0.4*NFFT+1,1,i),'plot'); hold on
...
      end
...
            plot(x,y,'-r','linewidth',1);
            set(get(gca,'Xlabel'),'String','Frequency (Hz)','Color','
   k');
            title('\fontsize{12}First singular value - all files');
```

Listing 5.15: Singular Values & MAC Figure Button - 2

A typical output graph given by *Singular Values & MAC Figure* is shown in Figure 5.1

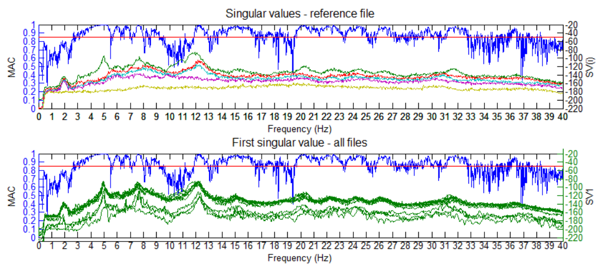


Figure 5.1: Singular Values.

Natural frequencies trend button shows the trend over time of the natural frequencies calculated for the selected measurements files. The callback *frequenzetempo_Callback* calls an external function (*plotfreq*), by specifying the frequency results (obtained using *getappdata* function and 0 (it could be, otherwise, the handles of a graph inside the interface) as parameters.
The function *plotfreq* is defined as follows:

```
function plotfreq(Result,fraxes)
    Result0 = Result;
    Result0(find(isnan(Result0))) = 0;
    for j=1:size(Result,2)
        media(j) = sum(Result0(:,j))/sum(sign(Result0(:,j)));
    end
    if fraxes~=0        axes(fraxes);
    else
      figure1=figure('Color',[1 1 1]);
    end
    maxf=max(max(Result(:,2:end)));
    for i = 2:size(Result,2)
        fr=num2str(trimmean(Result(:,i),20),4);
        set_legend(i-1) = cellstr(['Mode ' num2str(i-1) ': Hz ' fr
    ]);
    end
    plot(Result(:,1),Result(:,2:end),'linestyle','none','LineWidth
    ',1.2,'Marker','.','Markersize',5);
...
```

Listing 5.16: Natural frequencies development function

A typical output graph given by *Natural frequencies trend* is shown in Figure 5.2.

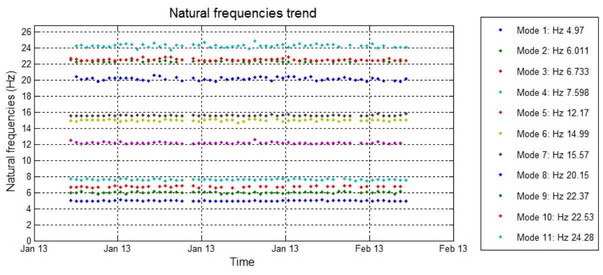


Figure 5.2: Natural frequency trend.

Once the natural frequency identification is done, damping ratios must be computed in order to complete the analysis.

5.3.2 Damping Ratio Calculation

Damping ratio calculation was one of the major purposes of this thesis and was implemented based on theoretical background introduced in Chapter . As was previously seen from a theoretical point of view, the calculation of the Damping Ratio parameter for the Enhanced Frequency Domain Decomposition method consists in several steps:

1. The SDoF Auto-Spectrum functions are identified.
2. Transform of the "spectral bells" to time domain using Inverse Fourier Transform.
3. The normalized auto-correlation functions are calculated.
4. For each natural frequency, damping ratio is calculated.
5. The global results table is built.

The *damping_ Callback* function processes all of these steps.
Other objects provide the necessary tools to view the results and to save them to file.
The *damping_ Callback* function is activated when the user presses the button labeled "Calculate damping ratio" in the main interface (Figure 4.1).

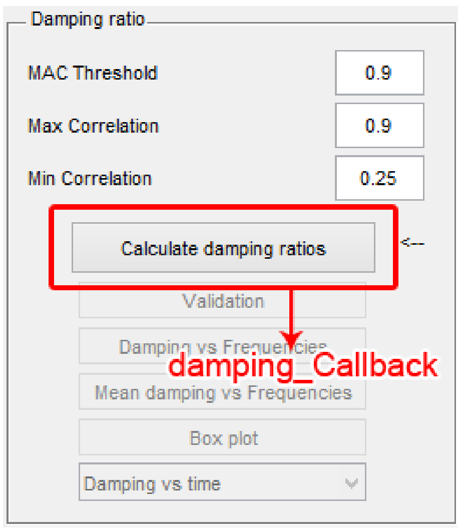


Figure 5.3: damping_ Callback activation.

Damping Callback Function

User configuration In the following block the parameters specified by the user via the GUI (*MAC Threshold, Max Correlation, Min Correlation*) are read using the *get* command. The *PERDAMPING.mat* file, previously stored in the *natural frequencies identification* step, contains the following variables:

- *NFFT*: the FFT length which determines the frequencies at which the PSD is estimated. It is required to transform the spectral bells to time domain.
- *SS0*: the matrix containing the singular values for each measure set.
- *Umodi*: the matrix containing the singular vectors for each measure set.
- *f1*: the vector of frequencies at which the CPSD is estimated.
- *fs*: the sampling frequency for the data sets.
- *newf2*: the natural frequencies previously calculated for each measure set.

Search of the position of natural frequencies In the following section the position of the PSD peaks in the frequency array is found and stored in the index array *indiceF*. This is done for each measure set: *size(SS0,3)* defines the number of the considered measure sets.

```
...
for ifast3=1:size(SS0,3);
    Segnale=SS0(1:round(intervallo*NFFT+1),1,ifast3);
    indiceF=ones(lastfm,1);
    damping=zeros(lastfm,1);
    j=1;
    i=1;
    while j<=lastfm && i < intervallo*NFFT+1
        i=i+1;
        if newf2(ifast3,j)==0
            j=j+1;
        else
            if (f1(i)>=newf2(ifast3,j) && f1(i-1)<=newf2(ifast3,j))
                indiceF (j) = i;
                j=j+1;
            end
        end
    end
```

Listing 5.17: Search of the position of natural frequencies

Identification of the SDoF Auto Spectral Function - 1 The SDoF auto spectral function is identified using the MAC criterion. For each structural mode, the variables *rangeL* and *rangeR* are defined as the number of singular values to the left and right of the peak to include within the SDoF auto spectral density function.
For each structural mode, a reference singular vector is selected in correspondence to the peak (*Urif=Umodi(indiceF(Fj),:,ifast3)*).

$$MAC = \frac{(u_{1k}u_{1i})^2}{u_{1k}^2 u_{1i}^2} \tag{5.3.2}$$

For each frequency line around the peak the MAC value is calculated, initially decreasing the frequency of a value at a time to obtain *rangeL*, subsequently increasing it to a value at a time from the frequency corresponding to the peak to determine *rangeR*. This value is compared with the threshold specified by the user (*MAC Threshold*). As long as the MAC is sufficiently high, the considered frequency is included in the SDoF auto spectral function and the *rangeL* o *rangeR* variables are updated.

```
rangemax=100;
rangeL=zeros(length(indiceF),1);
rangeR=zeros(length(indiceF),1);
k=1;

for Fj=1:length(indiceF) %Fj indice del modo considerato
  if indiceF(Fj)==1||indiceF(Fj)==0
      rangeR(Fj)=0;
      rangeL(Fj)=0;
  else

      Urif=Umodi(indiceF(Fj),:,ifast3);
      i=1;
        MAC=1;
      while i<rangemax && MAC>=sogliaMAC && i < indiceF(Fj)
          Uint=Umodi(indiceF(Fj)-i,:,ifast3);
         MAC=(Urif*Uint').^2/((Uint*Uint')*(Urif*Urif'));
      if( MAC < sogliaMAC)
             rangeL(Fj)=i-1;
           end
             i=i+1;
        end
        i=1;
        MAC=1;
        while i<rangemax && MAC>=sogliaMAC && i < indiceF(Fj)
          Uint=Umodi(indiceF(Fj)+i,:,ifast3);
         MAC=(Urif*Uint').^2/((Uint*Uint')*(Urif*Urif'));
          if( MAC < sogliaMAC)
             rangeR(Fj)=i-1;
          end
          i=i+1;
        end
```

Listing 5.18: Identification of the SDoF Auto Spectral Function - 1

This check serves to maintain the bell centered on the natural frequency. If the left range is too different from the right range, the higher range is reduced.

At most a 20% difference between the *rangeR* and *rangeL* is allowed.

Identification of the SDoF Auto Spectral Function - 2 Once the size in frequency of each SDoF Spectral Bell is found and stored in *rangeL(Fj)* and *rangeR(Fj)*, the SDoF auto spectral functions are calculated in variable *Bell*. The first dimension of *Bell* is the length of the First Singular Value vector (variable *Segnale*), whilst the second dimension is the number of vibrating modes.
Each bell is equal to the variable *Signal* only within the frequency range specified by *rangeL* and *rangeR* from the natural frequency corresponding to index *Fj*, while elsewhere is zero.

```
FF=newf2(ifast3 ,:);
Bell=zeros(length(Segnale),length(indiceF));
for Fj=1:length(indiceF)
  for i=1:length(Segnale)
    if i >= indiceF(Fj)-rangeL(Fj) && i <= indiceF(Fj)+rangeR(Fj)
      Bell(i,Fj)=Segnale(i);
    end
  end
end
```

Listing 5.19: Identification of the SDoF Auto Spectral Function - 2

From Frequency to Time Domain Each SDOF Auto spectral density function is then transformed to time domain using an Inverse Discrete Fourier transform, through the *ifft* Matlab function.
ifft is considered in the form *ifft(X,n)*, where X is the previously defined SDoF Auto Spectral Function, and n is the number of *FFT* points used in the *cpsd* operation: the variable *NFFT*.
What is obtained is the autocorrelation function of the SDoF system. The autocorrelation function is then normalized, dividing all of its values by its biggest value (*TBell(1)*). Only the first half of the normalized auto-correlation function is considered because the second half is given by the first part mirrored with respect to the central point.

The time domain is defined by variable *iT* using a time step equal to $1/f_s$, where f_s is the sampling frequency *fs*.

```
iT=(0:1/fs:200);
for Fj=1:lastfm
   TBell = real(ifft(Bell(:,Fj),NFFT));
   TBell0(:,Fj) = TBell(1:length(TBell)/2)./TBell(1);
   MyTBell=TBell0(:,Fj);
```

Listing 5.20: Transform of the SDoF auto-spectral function to time domain

Absolute maximum values envelope For each SDoF auto-correlation function, the maximum and minimum envelopes are found. The maximum auto-correlation function values are stored in *MaxTBell* and the corresponding time lags are stored in *MaxT*, while the minimum auto-correlation function values are stored in *MinTBell* and the corresponding time lags are stored in *MinT*.

```
      j=1;
      MaxTBell(j,Fj)=MyTBell(1);
      MaxT(j,Fj)=iT(1);
      j=j+1;
      for i=2:length(MyTBell)-1
            if MyTBell(i)>MyTBell(i-1)&&MyTBell(i)>MyTBell(i+1)
                  MaxTBell(j,Fj)=MyTBell(i);
                  MaxT(j,Fj)=iT(i);
                  j=j+1;
            end
      end
      j=1;
      minTBell(j,Fj)=MyTBell(1);
      minT(j,Fj)=iT(1);
      j=j+1;
      for i=2:length(MyTBell)-1
            if MyTBell(i)<MyTBell(i-1)&&MyTBell(i)<MyTBell(i+1)
                  minTBell(j,Fj)=MyTBell(i);
                  minT(j,Fj)=iT(i);
                  j=j+1;
            end
```

```
    end
end
```

Listing 5.21: Absolute maximum values envelope

In the next blocks damping ratios are actually calculated. First the indices that determine the position in the vectors of the maximum and minimum envelopes at the minimum and maximum correlation set by the user are identified and saved in *minTlag, maxTlag, minTlagmin, maxTlagmin.*

```
for Fj=1:lastfm
  if indiceF(Fj)==1 || (rangeL(Fj)==0 && rangeR(Fj)==0)
      FT(Fj)=NaN;
      damping(Fj)=NaN;
      damp_std(Fj)=NaN;
  else
      minTlag(Fj)=find (MaxTBell(:,Fj)< max_corr,1,'first');
      maxTlag(Fj)=find (MaxTBell(:,Fj)<min_corr,1,'first')-1;
    minTlagmin(Fj)=find (abs(minTBell(:,Fj))< max_corr,1,'first');
      maxTlagmin(Fj)=find (abs(minTBell(:,Fj))<min_corr,1,'first')
    -1;
```

Listing 5.22: Calculation of indices required to evaluate damping ratio

Then, the frequency estimation in time domain is calculated: $Frequency\_D = length(find(diff(MyTBell > 0)\tilde{} = 0)) + 1$; is the number of zero-crossing times in the chosen interval, and FT is calculated by dividing the number of zero-crossing times for the considered time interval.

```
      minTindex=find(abs(TBell0(:,Fj))<max_corr,1,'first');
      maxTindex=find(abs(TBell0(:,Fj))>min_corr,1,'last');
      MyTBell=TBell0(minTindex:maxTindex,Fj);
      Frequency_D = length( find(diff(MyTBell>0)~=0))+1;
        FT(Fj)=Frequency_D/(2*(iT(maxTindex)-iT(minTindex)));
```

Listing 5.23: Estimation of the natural frequencies in time domain from the SDoF free-decay function

Subsequently the damping ratio is calculated. First k and the logarithmic decrement are evaluated: the number of both maximum and minimum auto-correlation function values is required to calculate the parameter k, which is necessary for the logarithmic decrement δ calculation.

$$\delta = \frac{2}{k} ln\left(\frac{r_0}{|r_k|}\right) \tag{5.3.3}$$

Then the damping ratio is calculated as:

$$\zeta = \frac{\delta}{\sqrt{\delta^2 + 4\pi^2}} \tag{5.3.4}$$

```
          k=maxTlag(Fj)-minTlag(Fj)+maxTlagmin(Fj)-minTlag(Fj);
          delta(Fj)=2/k*log(max_corr/abs(min_corr));
          damping(Fj)=1/sqrt(1+(2*pi/delta(Fj))^2);
```

Listing 5.24: Calculation of damping ratio

In the next block, the graph of validation of the logarithmic decrement of the envelope of the absolute values is developed and saved in the variable *RettaRif*. This variable will be invoked within the validation graph.

```
          Rnorm=zeros(maxTlag(Fj)-minTlag(Fj)+1,1);
          RettaRif(1,Fj)=log(MaxTBell(minTlag(Fj),Fj));
          RettaRif(maxTlag(Fj)-minTlag(Fj)+1,Fj)=log(MaxTBell(
    maxTlag(Fj),Fj));
          Rnorm(1)=0; Rnorm(maxTlag(Fj)-minTlag(Fj)+1)=0;
          for rr=2:(maxTlag(Fj)-minTlag(Fj))
             RettaRif(rr,Fj)=RettaRif(1,Fj)-(RettaRif(1,Fj)-RettaRif(
    length(Rnorm),Fj))/(MaxT(maxTlag(Fj),Fj)-MaxT(minTlag(Fj),Fj))
    *(MaxT(minTlag(Fj)+rr-1,Fj)-MaxT(minTlag(Fj),Fj));
             Rnorm(rr)=log(MaxTBell(minTlag(Fj)+rr-1,Fj))-RettaRif(rr
    ,Fj);
          end
  end
end
```

Listing 5.25: Calculation of logarithmic decrement graph

The validation variables are saved inside temporary external files called *VALIDAZIONEx* where x is corresponding to the index of the file being analysed. *damping* vector is stored as a row of the global damping matrix.

```
    save (filename,'Segnale','FF','FT','damping','indiceF','f1','
      Bell','damp_std','minTlag','maxTlag','MaxT','max_corr','
      min_corr','MaxTBell','iT','TBell0','newf2','RettaRif');
    clear RettaRif;
    d_results(ifast3,:)=damping;
end
```

Listing 5.26: Saving of validation variables

Once all the files have been processed, global analysis is carried on damping values. In order to perform a mean of the values excluding outliers, temporary matrices are built, excluding *NaN* values which are not accepted by the *trimmean* function. $m = trimmean(X, percent)$ calculates the trimmed mean of the values in X. For a vector input, m is the mean of X, excluding the highest and lowest k data values, where $k = n \cdot \frac{percent}{2 \cdot 100}$ and where n is the number of values in X. In this case, k was set to 20%.
A mean value of damping and frequencies is calculated for each structural mode and stored in the variables d_mean and f_mean.

```
d_results0=zeros(size(d_results));
newf20=zeros(size(newf2));
for ii=1:lastfm
    iii=1;
    for ifast3=1:size(SS0,3);
        if ~isnan(d_results(ifast3,ii))
            d_results0(iii,ii)=d_results(ifast3,ii);
            newf20(iii,ii)=newf2(ifast3,ii);
            iii=iii+1;
        end
    end
    indz=find(d_results0(:,ii)>0,1,'last');
    d_mean(ii)=trimmean(d_results0(1:indz,ii),20);
    f_mean(ii)=trimmean(newf20(1:indz,ii),20);
end
```

Listing 5.27: Averaging operations

The global results are then finalized by attaching the first column, which comprises the dates of the measurements, and saved as an application data using the *setappdata* function.

```
load(Fdd_loc, 'Result');
d_results2=[Result d_results];
setappdata(0,'Result_damping',d_results2);
setappdata(0,'date_files',Result);
```

Listing 5.28: Results saving

The last block of the callback is composed of interface control commands.
The remaining buttons of the damping panel are enabled.
The *Save results* button is enabled, so that the user is able to save the global results.
The *Save template* button is enabled, so that the user is able to save a template on the basis of the current analysis.
The *damping vs time* drop-down menu is populated and the damping results are displayed, together with natural frequencies, inside the *Results panel*.

```
set(handles.validation,'Enable','on');
set(handles.dampingvsfrequencies,'Enable','on');
set(handles.meanvsfrequency,'Enable','on');
set(handles.boxplot,'Enable','on');
set(handles.saveres,'Enable','on');

strm{1}='Damping vs time';
for i=2:size(d_results2,2)
    strm{i}=['Mode ' num2str(i-1)];
    strl{i-1}=['Mode ' num2str(i-1,4) ': ' num2str(f_mean(i-1),4)
   ' Hz, Damping: ' num2str(d_mean(i-1)*100) '%'];
end
set(handles.modelist,'String',strl);
set(handles.dmode,'String',strm);
set(handles.dmode,'Enable','on');
set(handles.savetemplate,'Enable','on');
set(handles.statusD,'String','OK','BackgroundColor',[0 1 0]);
set(handles.phase2,'BackgroundColor',[0 1 0]);
set(handles.phase3,'BackgroundColor',[1 1 0]);
guidata(gcbo,handles);
```

Listing 5.29: Managements of the graphics components of the interface

Damping vs frequencies button plots the calculated damping ratios related to the corresponding natural frequencies. The callback *dampingvsfrequencies_Callback* calls an external function, *f_vs_d*, by specifying the variables *newf*20, *d_results*0, *f_mean* and *d_mean* as input parameters. The function *f_vs_d* is defined as follows:

```
function f_vs_d (newf2, d_results, f_mean, d_mean)

    figure1 = figure ('Color',[1 1 1]);
    % Create axes
  ...
    for i=1:size(d_results,2)
        scatter (newf2(:,i),d_results(:,i)*100); hold on;
    end
    scatter(f_mean,d_mean.*100,'MarkerFaceColor',[0 0 0],'
   MarkerEdgeColor',[0 0 0],...
        'Marker','+',...
        'LineWidth',2);
    hold off;
    ylabel({'\fontsize{12}Damping (%)'});
    xlabel({'\fontsize{12}Frequency (Hz)'});
```

Listing 5.30: Damping vs frequencies function

A typical output figure for this function is shown in Figure 5.4.

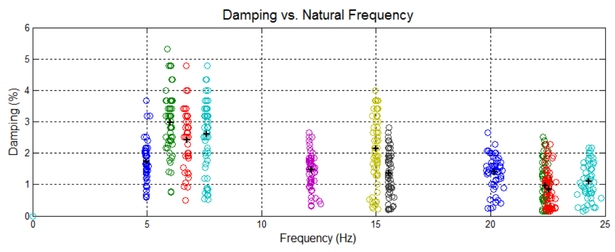


Figure 5.4: Damping versus Natural Frequency Graph.

Mean damping vs frequencies button plots the averaged damping ratios related to the corresponding averaged natural frequencies. The callback *meanvsfrequencies_Callback* calls an external function, *f_mean_vs_d_mean*, by specifying the variables *f_mean* and *d_mean* as input parameters. The function *f_mean_vs_d_mean* is defined as follows:

```
function f_mean_vs_d_mean(f_mean,d_mean)
    figure1 = figure ('Color',[1 1 1]);
...
    scatter(f_mean,d_mean.*100,'MarkerFaceColor',[0 0 0],'
   MarkerEdgeColor',[0 0 0],...
        'Marker','+',...
        'LineWidth',2);
    ylabel({'\fontsize{12}Damping (%)'});
    xlabel({'\fontsize{12}Frequency (Hz)'});
```

Listing 5.31: Mean damping vs frequencies function

A typical output figure for this function is shown in Figure 5.5.

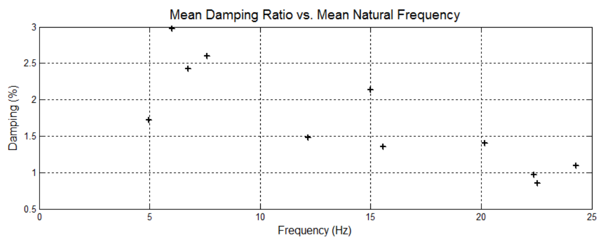


Figure 5.5: Mean Damping Ratio vs. Mean Natural Frequency Graph.

Box-plot button plots the box-plot representation of the damping ratios of each mode related to the corresponding averaged natural frequency.The callback *box_plot_Callback* calls an external function, *box_plot*, by specifying the variables *d_results*, *f_mean* and *d_mean* as input parameters. The function *box_plot* is defined as follows:

```
function box_plot (d_results, f_mean, d_mean)
    figure1 = figure ('Color',[1 1 1]);
...
    boxplot (d_results*100,f_mean,'plotstyle','compact','whisker'
   ,1); hold on;
    ylabel({'\fontsize{12}Damping (%)'});
    xlabel({'\fontsize{12}Frequency (Hz)'});
```

Listing 5.32: Box-plot function

A typical output figure for this function is shown in Figure 5.6.

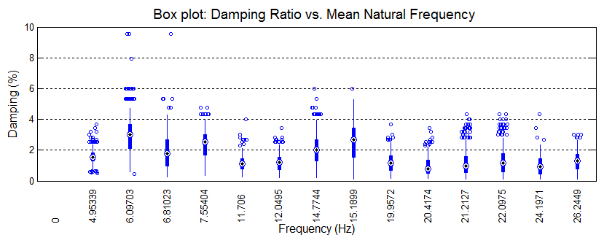


Figure 5.6: Box Plot.

Selecting an entry on the *Damping vs time* drop-down menu displays the corresponding trend over time of the damping ratio for each structural mode.

```
function dmode_Callback(hObject, eventdata, handles)
    load DAMPING_RESULTS;
    ind=get(hObject,'Value');
    if ind>size(d_results2,2) || ind <= 0
        errdlg('Indice non valido.');
    else
        if ind>1
            figure1 = figure ('Color',[1 1 1]);
...
            plot(d_results2(:,1),d_results2(:,ind).*100,'linestyle
    ','none','LineWidth',1.2,'Marker','.','Markersize',5); hold on;
    grid on;
            datetick('x','dd mmm yy','keeplimits','keepticks');
            xlabel('Time','FontSize',12); ylabel('[%]','FontSize'
    ,12);
            ylabel('Damping Ratio (%)','FontSize',12);
            title(['\fontsize{14}Damping Ratio Trend, Mode '
    num2str(ind-1)]);
        end
    end
```

Listing 5.33: Damping ratio developing over time for each mode

A typical output figure for this function is shown in Figure 5.7.

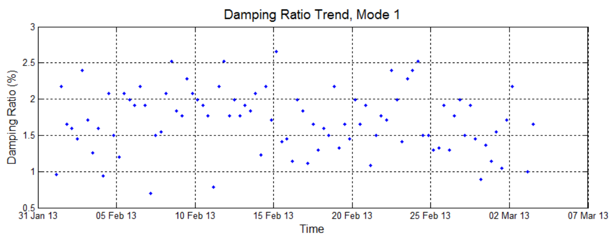


Figure 5.7: Damping Ratio Trend.

The button *Validate* simply launches a new interface, which contains tools to display the intermediate results obtained in the calculation of damping.

Damping Validation interface

The opening function of the interface populates the drop-down menu containing the dates of the measurements files and executes just before *validate_dampingGUI* is made visible. Selecting a date from the drop-down menu leads to the filling of the menu containing the modal parameters found for each file.

```
function date_Callback(hObject, eventdata, handles)
contents = cellstr(get(hObject,'String'));
ind=get(hObject,'Value');
handles.fileval=['damping_temp/validazione' num2str(ind) '.mat'];
guidata(hObject, handles);
load (handles.fileval, 'damping', 'newf2');
for i=1:length(damping)
    STR{i}= [ num2str(i) ': Frequency= ' num2str(newf2(ind,i)) '
    Hz, Damping= ' num2str(damping(i).*100) ' %'];
end
set(handles.mode, 'String', STR);
...
```

Listing 5.34: Drop-down menu of the measurement dates

The *Spectral bell button* displays the SDoF auto-spectral function corresponding to the selected mode for the selected file.

```
function spectral_Callback(hObject, eventdata, handles)
load (handles.fileval);
contents = cellstr(get(handles.mode,'String'));
selected=contents{get(handles.mode,'Value')};
```

```
index=str2num(selected(1:(strfind(selected,':')-1)));
Fj=index;
figure1=figure('Color',[1 1 1]);
...
plot1=plot(f1(1:length(Segnale)),20*log10(Segnale),f1(1:length(
    Segnale)),20*log10(Bell(:,Fj)));
...
title('\fontsize{14}Identification of auto spectrum');
```

Listing 5.35: Spectral bell button

A typical output figure for this function is shown in Figure **??**.

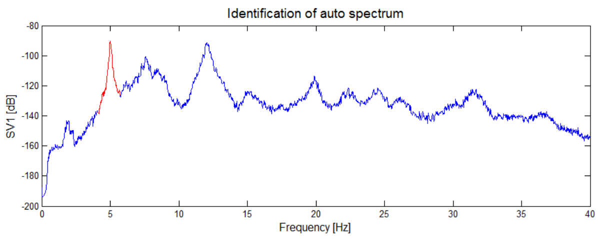


Figure 5.8: Identification of the autospectrum and of the SDoF Auto-Spectral Function.

The *Logarithmic decrement* displays the logarithmic decrement of the absolute envelope function of the SDoF normalized correlation function corresponding to the selected mode for the selected file.

```
function logaritmic_Callback(hObject, eventdata, handles)
load (handles.fileval);
contents = cellstr(get(handles.mode,'String'));
selected=contents{get(handles.mode,'Value')};
index=str2num(selected(1:(strfind(selected,':')-1)));
Fj=index;
figure1 = figure ('Color',[1 1 1]);
...
scatter(MaxT(minTlag(Fj):maxTlag(Fj),Fj),log(MaxTBell(minTlag(Fj):
    maxTlag(Fj),Fj)),'MarkerEdgeColor',[0 0.498039215803146 0],'
    Marker','+'); hold on
plot(MaxT(minTlag(Fj):maxTlag(Fj),Fj),RettaRif(1:(maxTlag(Fj)-
    minTlag(Fj)+1),Fj));
%                           plot(MaxT(minTlag(Fj):maxTlag(Fj),Fj),
    RettaRif(1:(maxTlag(Fj)-minTlag(Fj)),Fj));
```

```
ylabel('\fontsize{12}Log of Absolute Extreme Values');
xlabel('\fontsize{12}Time Lag (sec)');
```

Listing 5.36: Logarithmic decrement button

A typical output figure for this function is shown in Figure 5.9.

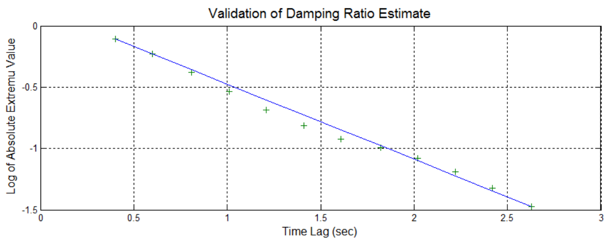


Figure 5.9: Validation of Damping Ratio Estimate: Logarithmic Decrement.

The *Frequency (F) vs Frequency (T)* displays the comparison between the natural frequencies calculated in frequency domain and those estimated in time domain (using EFDD) a for the selected file.

```
% --- Executes on button press in ff.
function ff_Callback(hObject, eventdata, handles)
load (handles.fileval);
contents = cellstr(get(handles.mode,'String'));
selected=contents{get(handles.mode,'Value')};
index=str2num(selected(1:(strfind(selected,':')-1)));
Fj=index;
figure1 = figure ('Color',[1 1 1]);
...
scatter(FF,FT);
ylabel({'\fontsize{12}Natural frequency estimated in time domain (
    Hz)'});
xlabel({'\fontsize{12}Natural frequency estimated in frequency
    domain (Hz)'});
```

Listing 5.37: Frequency (F) vs Frequency (T) button

A typical output figure for this function is shown in Figure 5.10.

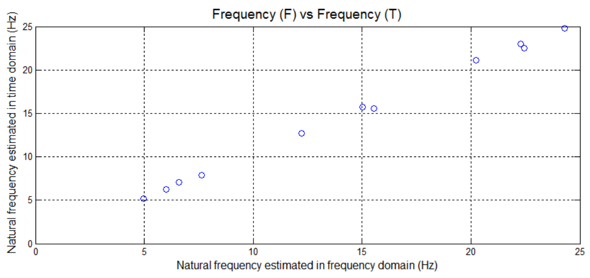


Figure 5.10: Natural frequencies calculated in frequency domain vs natural frequencies calculated in time domain.

The *Normalized auto-correlation function* displays the SDoF normalized correlation function corresponding to the selected mode for the selected file.

```
function normalized_Callback(hObject, eventdata, handles)
load (handles.fileval);
contents = cellstr(get(handles.mode,'String'));
selected=contents{get(handles.mode,'Value')};
index=str2num(selected(1:(strfind(selected,':')-1)));
Fj=index;
Rectx=[MaxT(minTlag(Fj),Fj) MaxT(maxTlag(Fj),Fj) MaxT(maxTlag(Fj),
    Fj) MaxT(minTlag(Fj),Fj) MaxT(minTlag(Fj),Fj)];
Recty=[max_corr max_corr -max_corr -max_corr max_corr];
figure1 = figure ('Color',[1 1 1]);
...       plot(Rectx,Recty,MaxT(minTlag(Fj):maxTlag(Fj),Fj),MaxTBell(
    minTlag(Fj):maxTlag(Fj),Fj),iT(1:length(TBell0)),TBell0(1:
    length(TBell0),Fj),'LineWidth',1,'Color',[1 0 0],...
          'DisplayName','funzione di autocorrelazione normalizzata')
    ;
     xlabel({'\fontsize{12}Time Lag [s]'});
     ylabel({'\fontsize{12}Normalized Correlation'});
```

Listing 5.38: Normalized auto-correlation function button

A typical output figure for this function is shown in Figure 5.11.

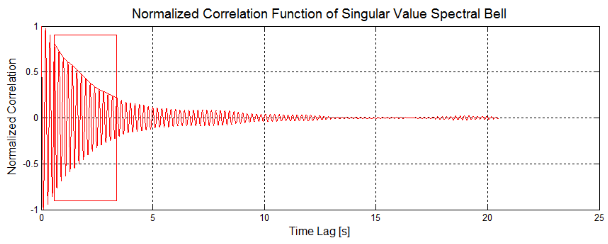


Figure 5.11: Normalized Correlation Function of the Singular Value Spectral Bell.

5.4 Management of results and templates.

5.4.1 Result management panel

After an identification task using AutoEFDD is completed, it is possible to save a *template file* using the button *Save template*.
The parameters (*NFFT*, *L*, *fs*, *channels*, *mac*, *n*, *mac_damping*, *max_corr*, *min_corr*) are saved inside the template file, together with the template's name and its description, author, date and image (set through a simple interface defined by *savetemplate*);

```
function savetemplate_Callback(hObject, eventdata, handles)
...
st=savetemplate;
waitfor(st);
selectfilename=getappdata(0,'tempname');
description=getappdata(0,'tempdesc');
author=getappdata(0,'author');
image=getappdata(0,'tempimage');
tempdate=datestr(now, 'dd/mm/yy HH:MM');
templatename=strrep(selectfilename,' ','_');
if ~exist('..\template','file')
    mkdir('..\template');
end
% templatename=selectfilename{1};
selectfolder=['..\template\' templatename];
mkdir(selectfolder);
savefile=[selectfolder '\Template_' templatename '.mat'];
save(savefile,'templatetype','tempdate','NFFT','L','fs','channels'
    ,'mac','n','mac_damping','max_corr','min_corr','templatename','
    description','author');
```

Listing 5.39: Save template button

The template is initially saved in *.mat* format. It is then inserted inside a *.zip* archive together with the files *indici.mat*, *savefile.mat* and the selected image.
The *.zip* file is then moved inside the default folder for the templates and its extension is renamed to *.efdd*.

```
copyfile('indici.mat',[selectfolder '\indici.mat'],'f');
copyfile('savefile.mat',[selectfolder '\savefile.mat'],'f');
copyfile(image,[selectfolder '\tempimage.jpg'],'f');
zip(['..\template\Template_' templatename '.zip'],'*',['..\
    template\' templatename]);
movefile(['..\template\Template_' templatename '.zip'],['..\
    template\Template_' templatename '.efdd'],'f');
...
setappdata(0,'templatename',templatename);
setappdata(0,'savingtemplate',1);

helpdlg(['The template file: Template_' templatename '.efdd has
    been successfully saved in the template folder.'])
```

Listing 5.40: Save template button - 2

Results panel

Save results The *Save results* panel saves the variables *Result_frequency* and *Result_damping* to file.
If the process is template-driven, the program does not display any dialog and automatically saves the results file inside the default directory.

```
function saveres_Callback(hObject, eventdata, handles)
Result_frequency=getappdata(0,'Result_frequency');
Result_damping=getappdata(0,'Result_damping');
if getappdata(0,'usingtemplate')==1 | getappdata(0,'savingtemplate
    ')==1
    foldername=['../Results/' getappdata(0,'templatename')];
```

```
     if ~exist(foldername,'file')
         mkdir(foldername);
     end
...
     templatename=getappdata(0,'templatename');
     setappdata(0,'resultsfolder',foldername);
     save([foldername '/Result_' num2str(indice) '.mat'],'
    Result_frequency','Result_damping','templatename');
else
     uisave({'Result_frequency', 'Result_damping'},'Result_');
end
set(handles.phase3,'BackgroundColor',[0 1 0]);
hh=helpdlg('Results successfully saved.');
...
```

Listing 5.41: Save results button

Merge results *Merge results* button loads an interface used for joining multiple results files. The callback *merge_Callback* calls the function *merge*.
If the process is driven by a template, the files contained in the default results folder of the template are automatically loaded in the text list.
Otherwise, the user can select one file at a time pressing the *Select results file to merge* button. The path of the file will then appear inside the text list below the button.
The user is able to change the position of a selected file in the list by pressing *Move up* and *Move down* buttons, while *Remove* button eliminates the file from the list (it does not delete the actual file).
The core of the merging procedure in *merge* resides in the *merge_Callback* function.
First, a check is performed in order to verify that the files are compatible.

```
function [handles]=merge_Callback(hObject, eventdata, handles)
files=get (handles.files,'String');
len=0;
dupl=0;
index=1;
deleteduplicate=get(handles.duplicate,'Value');
for i=1:length(files)
```

```
    vars=who('-file', files{i});
    isok=cell2mat(strfind(vars,'Result_frequency'))+cell2mat(
   strfind(vars,'Result_damping'));
    if isempty(isok) || isok~=2
        errordlg(['The file ' files{i} ' is not a result file.']);
        return
    end
    load (files{i});
    if i>1
        if size(Result_frequency,2)~=modi
            errordlg('The selected result files are not compatible
   . Different number of modes.');
            return
        end
    end
    modi=size(Result_frequency,2);
```

Listing 5.42: Merge and save button

Then, the rows of all the results variables are attached in a single variable. If the first element of a row, which represents the date of measurement, is already present into the variable, and if the user has checked the box *Delete duplicate entries*, then the entire row is removed.

```
%removal of duplicate entries
    if i>1 && deleteduplicate==1
        sizeRes=size(Result_frequency,1);
        for k=1:sizeRes
            if find(Sequence_f(:,1)==Result_frequency(sizeRes-k
   +1,1),1,'First')>0
                Result_frequency(sizeRes-k+1,:)=[];
                Result_damping(sizeRes-k+1,:)=[];
                dupl=dupl+1;
            end
        end
    end
%merge
    len=size(Result_frequency,1)+len;
    Sequence_f(index:len,:)=Result_frequency(1:size(
   Result_frequency,1),:);
```

```
    Sequence_d(index:len,:)=Result_damping(1:size(Result_frequency
   ,1),:);
    index=index+size(Result_frequency,1);
    clear Result_frequency Result_damping;
end
```

Listing 5.43: Merge and save button - 2

If the user has checked the box *Sort by date*, the variables are sorted in ascending order of dates in the first column.

```
%sort by date
if get(handles.sortdate,'Value')==1
    Sequence_f=sortrows(Sequence_f,1);
    Sequence_d=sortrows(Sequence_d,1);
end
Result_frequency=Sequence_f;
Result_damping=Sequence_d;
dialogstr=[num2str(length(files)) ' results files merged. '
   num2str(size(Result_frequency,1)) ' total rows. '];
if deleteduplicate==1
    dialogstr=[dialogstr num2str(dupl) ' duplicate entries deleted
   .'];
end
hd=helpdlg(dialogstr,'Merge details');
```

Listing 5.44: Merge and save button - 3

Once the sorting is complete, the user is asked to save the merged results file, and is allowed to view the results. If the process is template-driven, the results file is automatically saved.

```
if getappdata(0,'mergetemplate')==1
    save([getappdata(0,'resultsfolder') '/current_sequence.mat'],'
   Result_frequency','Result_damping');
    setappdata(0,'CurrentRes',[getappdata(0,'resultsfolder') '/
   current_sequence.mat']);
else
    [FileName,PathName] =uiputfile([getappdata(0,'path') '
   Sequence_'],'Save merged file');
```

```
    save([PathName FileName],'Result_frequency','Result_damping');
    setappdata(0,'CurrentRes',[PathName FileName]);
end
set(handles.view,'Enable','on');
setappdata(0,'mergetemplate',0);
pause(3);
if ishandle(hd) close(hd); end
```

Listing 5.45: Merge and save button - 4

View results The button labeled *View results* leads to the opening of a graphical interface used to display the results files. The callback *viewres_Callback* calls the function *viewresults* with no input parameters.
Inside the *View results* interface, *Load results file* button allows the user to choose the results file of interest.
The tools available in the *View results* interface are the same already introduced earlier.

5.5 Menu elements

The menu items can always be called by the user, given that they manage control operations or operations of global configuration.

File category

New The menu item *File* → *New* provides a clean interface discarding any changes made earlier.

```
function new_Callback(hObject, eventdata, handles)
button = questdlg('Are you sure you want to start a new project?
    Any unsaved progress will be lost.','title','Yes','No',2);
if strcmp(button,'Yes')==1
    curr=findobj('Tag','autoefdd');
    close(curr);
    AutoEFDDgui;
    setappdata(0,'usingtemplate',0);
    setappdata(0,'savingtemplate',0);
    setappdata(0,'scheduleron',0);
end
```

Listing 5.46: 'New' menu button

New from template The menu item *File* → *Newfromtemplate* requires the user to choose a template file. Once the file is loaded the informations of the template are displayed into a dialog (the interface *loadtemplate*), and all the parameters of the analysis are set in all the fields. The text fields are then disabled for user editing and the only enabled button is *Browse*, which is used for selecting the folder containing the measurements files.
If the process is schedule-driven, the selection of the input data folder is performed automatically without requiring any user intervention.

```
 function [handles]=load_Callback(hObject, eventdata, handles)
...
if getappdata(0,'scheduleron')==1
    FileName=getappdata(0,'schfilename');
    PathName=getappdata(0,'schpathname');
else
    [FileName,PathName] = uigetfile('../template/*.efdd','Select
   the template file to use.');
end
setappdata(0,'usingtemplate',1);
newfn='current.zip';    tmp='../tmp/';
copyfile([PathName FileName],[tmp newfn],'f');
unzip([tmp newfn],tmp);
delete([tmp newfn]);
templatename=regexprep(FileName,'.efdd','.mat');
load([tmp templatename]);
if ~strcmp(templatetype,'AUTOEFDD')
    errordlg('The selected file is not an AutoEFDD template.');
    return;
end
...
loadtemplate;
```

Listing 5.47: 'New from template' menu button

Results category

The items contained in the Results menu are:

- View results;
- Merge results.

These two functions are also available in the results panel and have already been discussed earlier.

Schedule category

The commands available from the *schedule* category relate to the planning and execution of monitoring activities over time, using the AutoEFDD procedure.

New schedule *New schedule* launches a window used to configure a new scheduled activity.
The *Browse* button at the right of *Select template* leads to the choice of a template file;
The *Browse* button at the right of *Select the folder containing files* leads to the choice of the default input data directory;
Setting a value in the *HH* and *MM* fields and pressing the *Add* button appends a time of execution of the analysis on the list. The *Save* button writes to file the *scheduled analysis*.

```
function salva_Callback(hObject, eventdata, handles)
schfilename=getappdata(0,'schfilename');
schpathname=getappdata(0,'schpathname');
fastdir=getappdata(0,'fastdir');
orari=get(handles.alarms,'String');
nome=get(handles.name,'String');
...
if ~exist('..\schedule','file')
    mkdir('..\schedule');
end
filename=['..\schedule\sch_' nome '.mat'];
```

```
save(filename,'schfilename','schpathname','orari','nome','fastdir'
    );
movefile(filename,['..\schedule\sch_' nome '.schedule'],'f');
ww=helpdlg('Schedule successfully create. Launch it with Schedule
    Launcher.');
pause(5);
if ishandle(ww) close(ww); end
```

Listing 5.48: 'New schedule' menu button

Schedule launcher Schedule launcher enables the execution of scheduled tasks. By setting *setappdata(0,'scheduleron',1);* the process becomes schedule-driven. This means that no user interaction is required in order to perform the analysis, since the input folder is already defined in the schedule file. First the details of the scheduled analysis are read from the schedule file.

```
function schedulelauncher_Callback(hObject, eventdata, handles)
[FileName,PathName] = uigetfile('..\schedule\*.schedule','Select
    the scheduler you want to launch.');
schname=[PathName FileName];
if ~exist('..\tmp\','file') mkdir ('..\tmp\'); end
newfile=['..\tmp\' strrep(FileName,'.schedule','.mat')];
copyfile(schname,newfile,'f');
continua=1;
while continua==1
     clearvars -except handles continua newfile;
     load(newfile);
     setappdata(0,'scheduleron',1);
     setappdata(0,'schfilename',schfilename);
     setappdata(0,'schpathname',schpathname);
     set(handles.folder,'String',fastdir);
     orari=datenum(orari,'HH:MM');
     orari=datenum(0,0,0,hour(orari),minute(orari),0);
     adesso=clock;
     curr_time=datenum(0,0,0,adesso(4),adesso(5),0);
     for i=1:length(orari)
        if orari(i)<=curr_time
           oraritmp(i)=etime(datevec(orari(i)+datenum(0,0,1,0,0,0)),
    datevec(curr_time));
```

```
        else
          oraritmp(i)=etime( datevec(orari(i)),datevec(curr_time));
        end
    end
```

Listing 5.49: 'Schedule launcher' menu button

Then, the times of execution are sorted in ascending order with respect to the present time. The first time the list is the next time at which the analysis will be launched. The *timer* function is used in order to set the execution of the subsequent task in a given time interval.

```
    oraritmp=sort(oraritmp);
    prossimo=datestr((datenum(0,0,0,0,0,oraritmp(1))+curr_time),'
  HH:MM');
    t = timer('TimerFcn', 'setappdata(0,''execute'',1);',...
          'StartDelay',oraritmp(1),'TasksToExecute',1);
    start(t)
    hhh = helpdlg(['Next scheduled analysis is set at ' prossimo '
  .' char(10) 'Do you want to interrupt the schedule?'],'Schedule
   Launcher');
    while strcmp(get(t,'Running'),'on')
        pause(10)
        if ~ishandle(hhh)
            stop(t);
            delete(t);
            setappdata(0,'execute',0);
            continua=0;
        end
    end
```

Listing 5.50: 'Schedule launcher' menu button - 2

Once the timer has reached the target time, the analysis is launched.

```
    if ishandle(hhh) close(hhh); end
    if getappdata(0,'execute')==1
        [handles]=load_Callback([],[],handles);
        [handles]=startfdd_Callback([],[],handles);
    end
    delete(t);
    rmappdata(0,'execute');
end
setappdata(0,'scheduleron',0);
```

Listing 5.51: 'Schedule launcher' menu button - 3

5.6 Conclusions and future developments

An overview of the Matlab code at the base of the application interface has been carried out within this chapter.
First the algorithm used for the damping ratio calculation was accurately analysed. Then all the major callbacks of the interfaces that constitute the program were considered, especially focusing on the way that data are passed from a function to another and from graphical objects to functions.

The program has, however, not yet reached a final state and it is expected a number of improvements and additions for future versions, including:

- Mode shape correlation with those obtained by FE models;
- Log files to trace the operations;
- Optimization of data reading and Fourier transform operations in order to improve the speed of the analysis;
- Automatic setting of some parameters such as *MAC Threshold* and *Step* in the identification of natural frequencies;
- Automatic Output in the form of printable report, covering graphs and analysis results;
- Online alarm in case of any significant changes of the parameters of the structure as a result of exceptional events such as earthquakes;
- Elimination of the environmental effects (temperature, humidity) affecting the results using ARX models;
- Check of the input files in order to determine the validity of the measures.

Chapter 6

Application: Ponte Nuovo Del Popolo, Verona

Shown below is an application of the methods and tools presented in the thesis, by means of a bridge located in Verona, Ponte Nuovo del Popolo.
First the geometric and mechanical characteristics of the structure will be described. Then we will introduce the FEM model of the structure, from which a first evaluation of the dynamic characteristics of the bridge will be derived. Afterwards it will be explained the process of planning the work of dynamic identification and the results obtained with the AutoEFDD and AutoSSI methods will be presented and compared with those provided by the FEM model and other commercial software, such as Artemis. Finally we will cover an overview on model updating and the setting of seismic analysis for the bridge.

6.1 Inventory of information on the bridge

Ponte Nuovo del Popolo bridge, often referred to in abbreviated form as *Ponte Nuovo*, is located in the city center of Verona, on the Adige river. It is a reinforced concrete bridge, three-span arched, with an outer coating made of stone. The bridge was built just after World War II to replace the previous bridge, which was inaugurated just six years before, and was destroyed in the bombing.

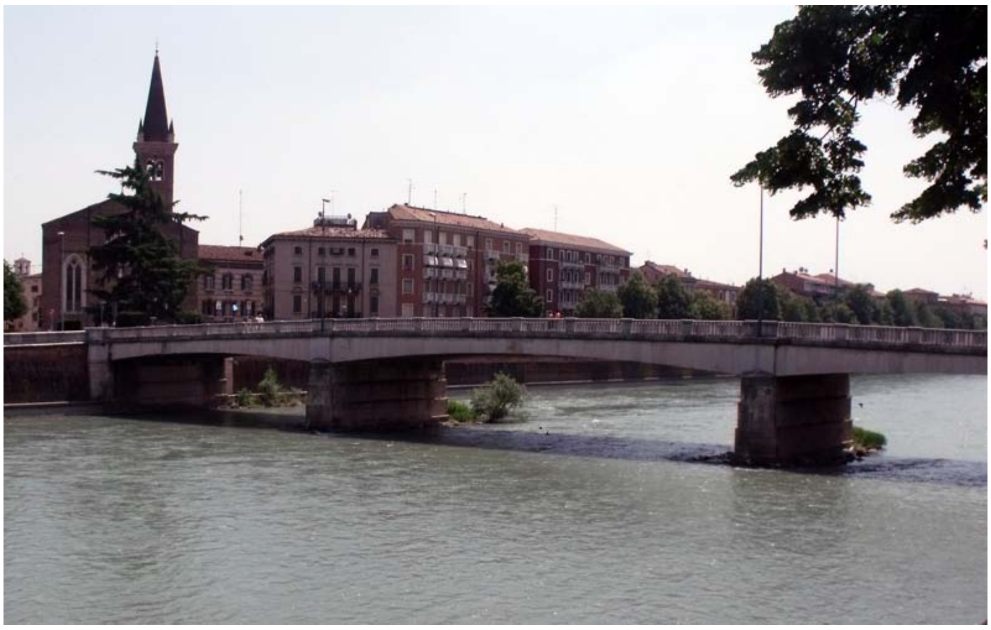


Figure 6.1: Ponte Nuovo del Popolo, Verona.

6.1.1 Geometric and material survey

The geometry of the structure is known based on some original drawings and measurements performed at the intrados of the bridge in correspondence of all three spans [8]. The bridge has a total length of 90.5 meters. Regarding the three spans, the central one has a length greater than the two lateral, in particular, the central arch has a span of 33.42 meters, while the two lateral arches have a span of 25.23 meters.
The overall width of the deck is 14.32 meters, the frame supports the roadway with two lanes, one in each direction, plus two sidewalks on both ends of the cross section. The sidewalks have a width of 2.55 meters each.
The top of the deck slab in reinforced concrete has a project thickness equal to 18 cm.
The static scheme consists of seven main beams, placed longitudinally along the length of the bridge, and eleven of the stiffening crosspieces, which connect in the transverse direction all the main beams.
There are also transversal joists, smaller in size compared to the crosspieces, which connect at regular intervals the five central longitudinal beams, that is, from number 2 to number 6, they form a lattice of crosspieces, which complements a series of strengthening bars cross .

The intrados of the bridge presents itself closed by a slab of reinforced concrete

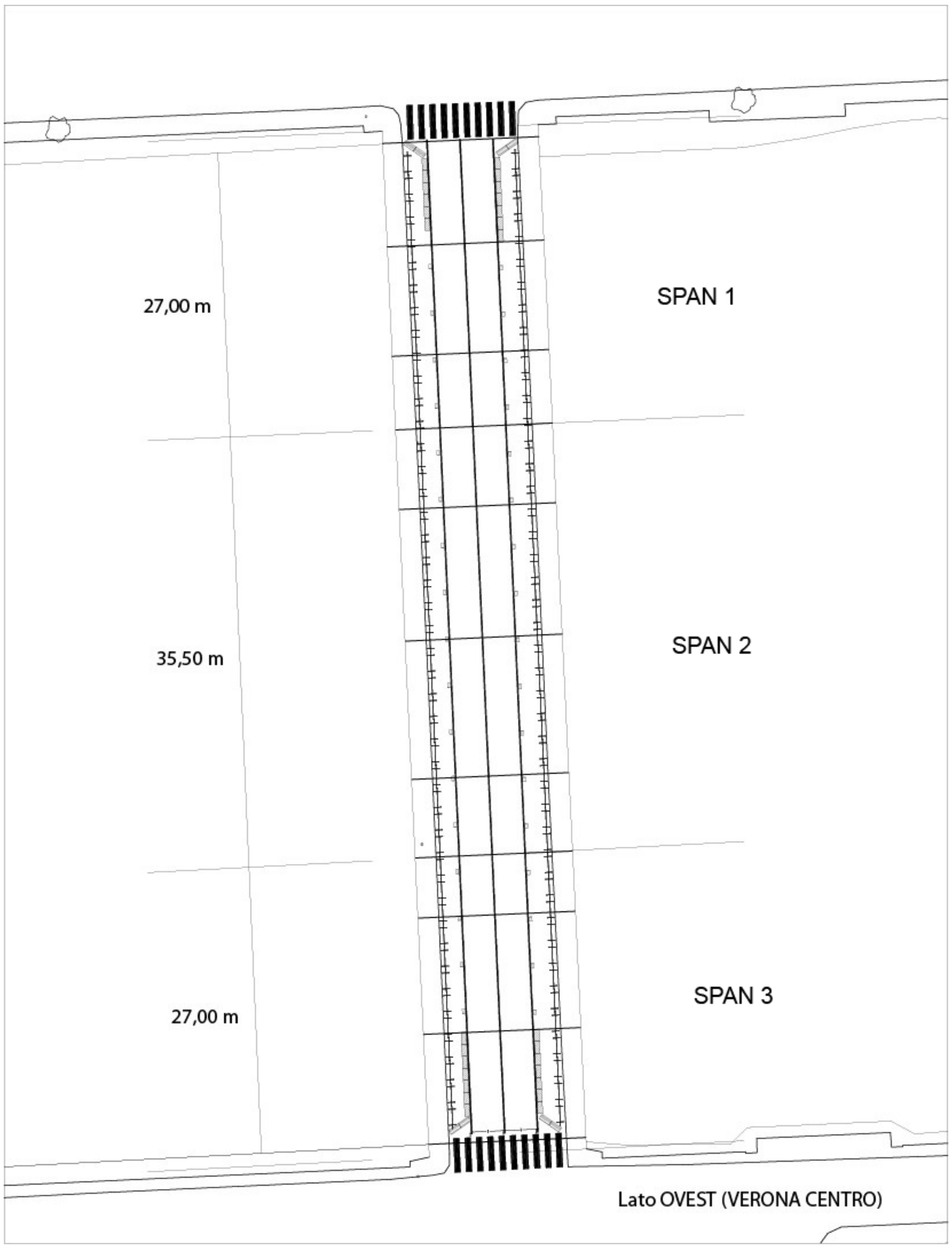


Figure 6.2: General plan of the bridge.

in the piers, in the section between the stack and the two adjacent transverse stiffening crosspieces, the first towards the middle of the bridge, the second in the side spans. The above intrados slab has a variable thickness: in correspondence of the stack is equal to 25 cm, while in correspondence of the first stringers near the pier is equal to 10 cm [8].

Deck slab

The slab of the deck is made of reinforced concrete, with a thickness of project of 18 cm, is supported by a pattern of beams in lattice, formed by the seven main longitudinal beams and fractionated in the transverse direction from the main stiffening crosspieces and joists secondary cross.
Following a series of inspections, has highlighted a mesh of armor composed of reinforcement rods of a diameter of 16 mm in the direction parallel to the main longitudinal beams, with bars arranged at regular intervals of 25 cm, while in the direction parallel to the crosspieces reinforcement bars of a diameter of 10 mm were detected, also arranged at intervals of 25 cm.

Main longitudinal beams

The slab of deck is structurally supported by seven main longitudinal beams of constant thickness of 40 cm, and height variable in proportion to the distance from the intermediate support and from the piers of the bridge. The longitudinal reinforcements are arranged on four layers of variable length in a way proportional to the size of the bending moment present along the main beams.
The brackets are made of steel bars of diameter of 16 mm distanced 35 cm from each other.

Main strengthening bars

The transverse main stiffening elements are characterized by a thickness of 25 cm and a height variable in relation to the height of the main longitudinal beams.
In the central span, the centerline stringer has a height of 1 meter, while the two adjacent ones have a height of 1.22 meters.

Transversal secondary joists

The static scheme of the bridge presents some small transverse reinforced concrete joists, arranged parallel to the main stiffening stringers. From the measurements made, they have a section with a thickness of 20 cm and a height equal to 32 cm. The above secondary cross joists are spaced with a pitch equal to 2 meters, except the first joist for each side from the two supports at the ends of the bridge, which are positioned at 1.85 meters from the support itself. The secondary joists are clamped with reinforcing steel with a diameter of 10 mm, with a pitch of 25 cm.

6.2 Inspection works

6.2.1 Materials testing

The knowledge of the characteristics of the materials that compose the bridge is crucial for subsequent analysis. The purpose of the determination of the mechanical characteristics of the materials was in fact to be able to calibrate a finite element model of the bridge, in order to perform dynamic identification. Following a series of inspections on the structure, an experimental investigation aimed at providing all the real characteristics of the materials was performed. In order to properly define the characteristics of the constituent materials, such as steel and concrete, 4 pieces of reinforcing bars were extracted in places where it would not aggravate the structural situation; 10 samples of concrete circular with a diameter of 100 mm were also collected. In addition, the sclerometric tests have been performed on site on the concrete forming the soffit of the longitudinal beams and stringers[8].

Compressive strength of the concrete

The analysis of the strength of the concrete on site is performed in accordance with Italian technical regulations, following the directions of the Circular of 2 February 2009, "Istruzioni per l'applicazione delle nuove Norme Tecniche per

le Costruzioni di cui al D.M. 14 gennaio 2008".
Being the tests carried out on the basis of samples taken from the structure, the law refer to UNI EN 12504-1, 12390-1, 12390-2, 12390-3.
In the compression tests the specimen is positioned between the two steel plates of a press, by imposing a constant increase of the compressive stress applied, until reaching the breaking load of the core sample. The breaking load is not found in correspondence of the disintegration of the specimen, but instead it is assumed in correspondence of the stabilization of the load. When the breaking load is reached, the characteristics cracks which affect the integrity of the concrete samples are in fact identified.
As a result of laboratory tests on the samples, average values of the compressive strength of the concrete for the two types of structural elements are determined:

Structural part	f_{ck} **[MPa]**
Longitudinal beams	21.9
Stringers	19.1

Table 6.1: Average compressive strength of the concrete

Modulus of elasticity of concrete

The elastic modulus, as defined in the UNI 6556, is the relationship between the tension σ and the corresponding strain ϵ measured in the direction of the stress. For materials, such as concrete, for which the stress-strain diagram is not linear, it is defined:

- Secant modulus of elasticity between two stresses the one determined by the slope of the secant to the stress-strain diagram between two considered stress values.
- Tangent Elastic modulus for a given voltage that determined by the slope of the tangent of the geometric stress-strain diagram in correspondence of that tension.

The samples selected for the determination of the elastic modulus are two core samples extracted from the longitudinal beams and two core samples extracted from the stringers.
Since the applied load is known, the applied stress $\sigma(t)$ and the corresponding deformation $\epsilon_i(t)$ of the ith extensometer at a generic instant t are first calculated using the following formulations:

$$\sigma(t) = \frac{P(t)}{\frac{\pi d^2}{4}} \tag{6.2.1}$$

$$\epsilon(t) = \frac{\Delta l_i(t)}{l} \tag{6.2.2}$$

The secant modulus of elasticity E (in compression) between the two stress values which limit each phase of loading-unloading is therefore determines, for each stabilized cycle, using the formula:

$$\frac{\Delta\sigma}{\Delta\epsilon} = \frac{\sigma_i - \sigma_{0i}}{\epsilon_i} \tag{6.2.3}$$

where:

- $\Delta\sigma$: Stress Range;
- σ_i: Maximum stress of the test cycle
- σ_{0i}: Starting stress of the test cycle
- $\Delta\epsilon$: Unit change in length corresponding to this interval measured during unloading.

For each considered structural element (longitudinal beams and stringers), a value of the concrete secant modulus of elasticity in compression E is calculated as the average of the respective samples from the corresponding element. The values are given in the following table:

Steel tensile strenght

Following the inspection on the structure some reinforcing bars have been extracted, on which was subsequently performed a tensile test in the laboratory

Structural part	**Modulus of elasticity E [MPa]**
Longitudinal beams	33040
Stringers	32562

Table 6.2: Average concrete secant modulus of elasticity in compression

in order to obtain the elastic modulus and the tensile strength.
From the test is determined by the stress-strain diagram of the material. It is possible to distinguish certain characteristic traits of the curve:

- A first linear elastic part;
- A short non-linear elastic part;
- A section in which the deformation grows without causing significant stress increases: this is the part where the yield strength of the steel is reached;
- A hardening part, in which small increases in stress determine an increase of the deformation;
- A final section that leads to rupture the specimen, with a clear necking in the transverse rupture section with subsequent decrease of the stress.

The elastic modulus E characterizes the behaviour of the material, representing the relationship between stress and strain in the case of uni-axial loading condition. It is measured in the initial portion of the linear elastic stress-strain curve and assumes a value equal to:

$$E = \frac{\Delta\sigma}{\Delta\epsilon} = 144065Mpa \tag{6.2.4}$$

The values of the project tensile strength for each element will be assumed equal to the mean value of the yielding stress obtained with the tensile tests, as reported in the following table:

Element	f_{yk} **[MPa]**
Reinforcing bars	360.21

Table 6.3: Project steel tensile strength

6.2.2 Overview of damage

Following a series of inspections on the structure in June 2011 [8], an overview of the state of deterioration of the bridge was outlined.
The higher level of damage of the structure is concentrated in correspondence of the external longitudinal beams, in the direction of the two side faces of the bridge. Moreover, the beams positioned downstream are, in all the spans, slightly more damaged than those in the side facing upstream. A localized state of shear cracking in the central part of the crosspieces was observed, while the terminal zones are healthy.

6.3 Structural analysis

The purpose of structural analysis is to interpret the characteristics of structures in order to simulate their response under various situations. Structural analysis may be divided in two major categories, depending on the way the structure is loaded, which are *static* and *dynamic* structural analysis.
Whether the loading acting on the structure is stationary over time, and its application point doesn't change, the structural analysis is considered to be static. Otherwise, if the loading is variable during the reference period, the analysis is said to be dynamic.
In the second case the analysis needs to consider inertial forces and therefore it must be able to describe accurately the mass distribution of the structure.

Finite Elements Method Finite Elements Method (FEM) is a very popular numerical approach that allows to describe a structure as an assembly of simple elements, in order to provide an approximated, algebraic, solution to the differential equations that constitute the continuous real system. A *mesh* is built through a discretization of the actual system using simple coded elements. For each type of elements a set of *shape functions* is pre-defined. The solution of the system, in terms of displacements, is calculated in characteristic points of each element and, by means of the shape functions, provides the solution for each point of the system.

It is necessary to keep in mind that when modeling a real, constructed structure, using a FE model, a lot of assumptions and hypothesis are, more or less explicitly, required. For example, each finite element requires a set of mechanical and geometrical properties, which must be meaningful (Is the material homogeneous? Is the hypothesis of small deformations acceptable? Is it possible to operate in conditions of linear elasticity? ...), and the type of element must be compatible with the behaviour of the considered part of the structure. The constraint conditions are equivalent to those specified by the preliminary designs or are they actually different?
These assumptions must be done very carefully since they heavily influence the reliability of the model of the structure.

6.3.1 Model preparation

A finite element model of the structure was developed using the Strand7 software (HSH Computing, Padua).
In this study case only one type of solver is used: *Natural Frequency Analysis*. The model is three-dimensional type. The main beams of the deck, the main cross beams, the secondary cross joists and the piles were modeled using beam elements, while the slabs of the deck were modeled using plate elements. The congruence between the upper slab and the lower slab is provided by rigid links. The two arches are obtained by assigning a variable section to the beam

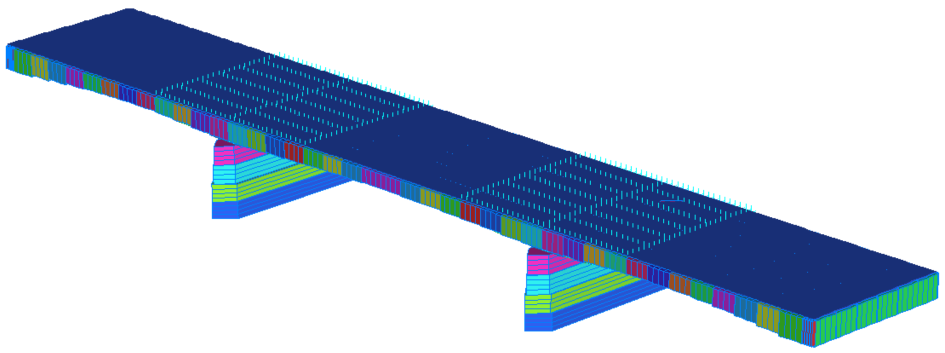


Figure 6.3: The FEM model of the bridge.

elements, while the thickness variation of the bottom slab in the longitudinal direction of the bridge was obtained by varying the thickness of the plate elements.

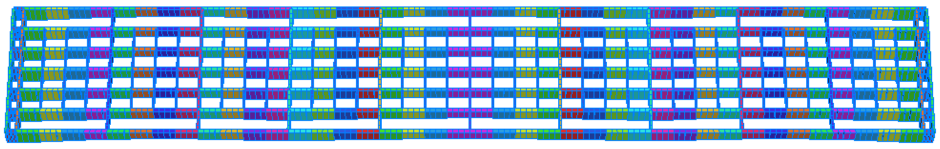


Figure 6.4: The lattice of deck girders, modelled with beam elements.

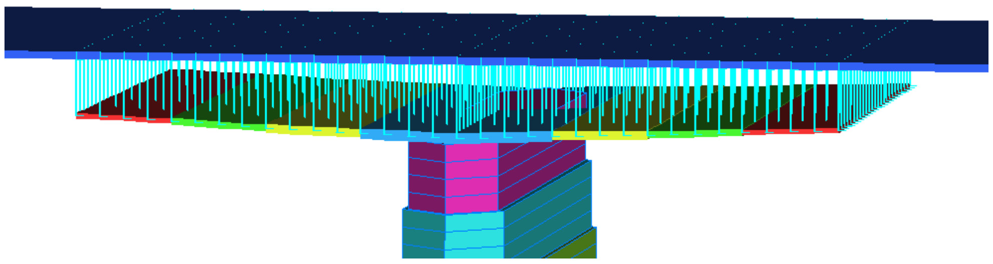


Figure 6.5: The rigid links that connect the bottom slab to the top.

The mechanical properties of the components of the model were evaluated starting from the laboratory tests carried out on the samples extracted from the bridge itself.

Dynamic analysis The purpose of the *Natural frequency analysis* is to provide an interpretation of the structural behaviour of the structure in order to allow an adequate choice for the placement of the measurement instruments. The structural vibration modes and the corresponding natural frequencies represent the structural response of the structure. The superposition of all the structural modes provides, in fact, the global dynamic behaviour of the structure.

6.4 Modal testing

The dynamic test is carried out in order to develop a system for dynamic identification and monitoring for the structure.

The previously introduced FEM model is considered. The model is necessary to define the position of the measurement points, by observing the vibration modes. The configuration of the measurement system of the structure on the most significant points is then carried out.
Subsequently, the measurements obtained from the structure are processed by means of methods based on Operational Modal Analysis. Initially a dynamic identification of the structure is carried out. Then the monitoring operations are described and the results obtained over a year of measurements are presented.
A comparison between the results provided by the various methods will indicate the reliability of the results.
Once OMA methods are validated, their results can provide a calibration for the FEM model. It will also be possible to observe any change in the modal parameters over time.

Overview of Operational Modal Analysis methods used The methods of operational modal analysis that have been used to identify the modal parameters of the structure have been presented previously in the thesis. An overview of the methods used is given below.

- The OMA method of most interest for this thesis is the *Enhanced Frequency Domain Decomposition* (EFDD) method, that was implemented in **AtOMA** as the *AutoEFDD* procedure. It provides an automatic extraction of the modal parameters of the structure, namely natural frequencies and damping ratios, without the need for any spatial configuration or *Peak-Picking* procedures. EFDD is a frequency domain method that is based on the Singular Value Decomposition of the Cross Power Spectral Density function to obtain the natural frequencies, and subsequently performs an Inverse Discrete Fourier transform of the SDoF Auto Spectral Density functions in order to extract the damping ratios.
- A second approach implemented in **AtOMA** to perform structural identification is the *Stochastic Subspace Identification* (SSI), that was implemented as the *AutoSSI* procedure. SSI is a time-domain method, which

considers the columns of the acquired signal as a base of vectors, while rows allow to obtain a sequence of estimates evaluated using a battery of Kalman filters. Such matrices, can be determined directly from the only knowledge of the output signals, without a priori knowledge of the matrices characterizing the model.

- A peak-picking FDD procedure is also used by means of the commercial software *ARTeMIS*. ARTeMIS allows to recognize an estimate of the mode shapes of the structure, which can be compared to the mode shapes given by the FE model.

The results given by these different methods are then confronted with each other.

6.4.1 Structural Identification of the Bridge

Operational modal analysis methods are based on measuring the response of the structure to environmental excitations, provided by accelerations, at significant points in the structure. To perform a FDD identification analysis using ARTeMIS, it is necessary to define multiple measurement setups so that a spatial estimate of the mode shapes can be performed.
In order to obtain a spatially complete and unambiguous description of the mode shapes, the correct choice of the number and positioning of the measuring instruments is crucial. From a FEM model, based on prior knowledge of the structure, the dynamic behavior of the bridge is obtained. The dynamic characteristics obtained from this analysis are not likely to match those of the actual structure, especially when the structure has undergone degradation and aging. The choice of measuring points is based on the minimization of the number of degrees of freedom of the model in order to make the eigenvalue problem associated with it more simple. This method is called *Guyan reduction.*

Selection of the measuring points using Guyan Reduction Model reduction is a key issue for the analysis of mechanical systems. The Partial Differential Equation which describes the behaviour of the elastic body is transformed in a second order Ordinary Differential Equation of the form:

$$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = Bu(t) \tag{6.4.1}$$

where M, C, K are the system matrices $Bu(t)$ is the load vector and x is the unknown vector with n DoF. Considering a FEM model of an actual civil structure, $n \in 10^5, 6 \cdot 10^6$, which leads to large dimension system matrices and thus to vast storage and simulation time needs. Especially, since each measuring instrument is meant to correspond to a degree of freedom of the structure, it is evident the need to reduce the size n. The general concept of model reduction is to find a low-dimensional subspace $T \in \mathcal{R}^{n\times n}$ in order to approximate the state vector $x = Tx_R + \varepsilon$ [29].
Guyan's method is the oldest reduction method, and is based on the notion of master/external and slave/internal DoFs.
Considering an undamped system:

$$M\ddot{x}(t) + Kx(t) = f \tag{6.4.2}$$

The m-set of Master DoFs is defined as the set of total DoFs that remain in the previous equation. Analogously the s-set contains all DoFs that will be eliminated:

$$m \cup s = n, \quad n = DOF_{total}, \quad m \cap s = \emptyset \tag{6.4.3}$$

By partitioning the system matrices into block matrices that depend explicitly on the m-set or s-set or a combination of them, the following re-ordered system is obtained:

$$\hat{M}\begin{pmatrix}\ddot{x}_m\\ \ddot{x}_s\end{pmatrix} + \hat{K}\begin{pmatrix}x_m\\ x_s\end{pmatrix} = \begin{pmatrix}f_m\\ f_s\end{pmatrix} \tag{6.4.4}$$

where:

$$\hat{M} = \begin{pmatrix}M_{mm} & M_{ms}\\ M_{sm} & M_{ss}\end{pmatrix}, \quad and \quad \hat{K} = \begin{pmatrix}K_{mm} & K_{ms}\\ K_{sm} & K_{ss}\end{pmatrix} \tag{6.4.5}$$

The second equation is solved for x_s and its is assumed that there is no force applied on the external DoFs, i.e $f_s = 0$. The transformation matrix for the static reduction is obtained by omitting the equivalent inertia terms:

$$\begin{pmatrix}x_m\\ x_s\end{pmatrix} = \begin{pmatrix}I\\ -K_{ss}^{-1}K_{sm}\end{pmatrix}x_m = T_{static}x_m \tag{6.4.6}$$

Guyan reduction is a good approximation for the lower eigenfrequencies respectively eigenvectors.
Applying a Laplace transformation on the undamped system differential equation, the equivalent system is given:

$$(-M\omega^2 + K)X(\omega) = B(\omega)X(\omega) = F(\omega) \tag{6.4.7}$$

which is then re-ordered into the block partitioned master/slave DoFs as defined previously, giving in that way the transformation matrix for the dynamic reduction:

$$\Rightarrow T_{dynamic} = \begin{pmatrix} I \\ -B(\omega)_{ss}^{-1}B(\omega)_{sm} \end{pmatrix} \tag{6.4.8}$$

where:

$$B(\omega)_{i,j} = -M_{i,j}\omega^2 + K_{i,j}, \quad i,j \in \{s,m\} \tag{6.4.9}$$

So far we have described the application of the Guyan reduction to a complex FEM model whose objective is to generate a reduced model that retains as much as possible the characteristics of the original one at low frequencies. In many ways the criteria for selecting the measurement points in a complex system is the same: we want to accurately measure the modes at lower frequency. So it is reasonable to postulate that the coordinates of a master FEM model can also be used as acquisition points in the dynamic tests. In any case it is necessary that the coordinates removed must be well connected in the structure.
So, we start with a finite element model, which will have many more DoFs than are actually measurable. Before the automatic procedure of selection is initiated, all the DoFs that can not be reasonably used as measuring points are removed, namely the rotational DoFs and the inaccessible points. The automatic selection is then started, and ends when the reduced model retains a number of master coordinates equal to the number established for acquisitions.
Once the *master* DoFs were identified, some accelerometers were positioned on the bridge in fixed positions for the duration of data acquisition in the site, and a set of sensors were put from time to time in different positions, in order to have multiple setups.
The setup cases are shown in the following figures [8].

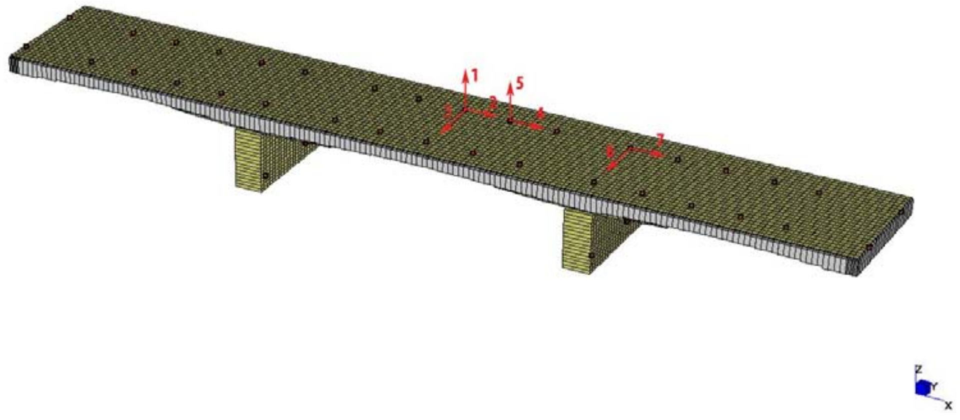


Figure 6.6: Setup 1.

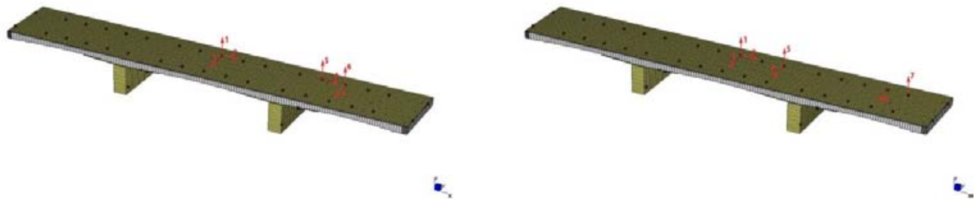


Figure 6.7: Setup 2 (left) and setup 3 (right).

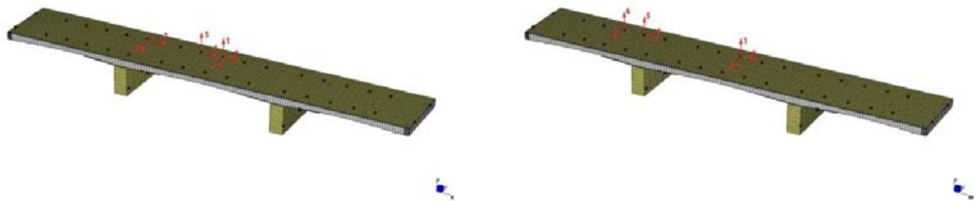


Figure 6.8: Setup 4 (left) and setup 5 (right).

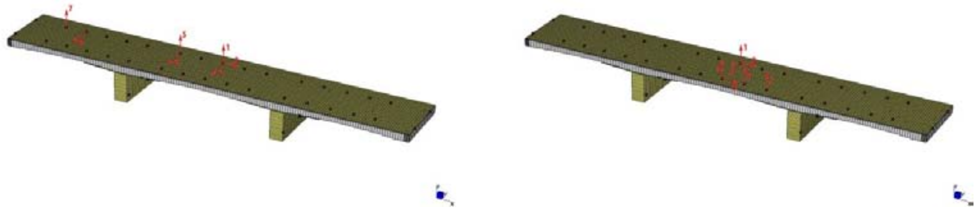


Figure 6.9: Setup 6 (left) and setup 7 (right).

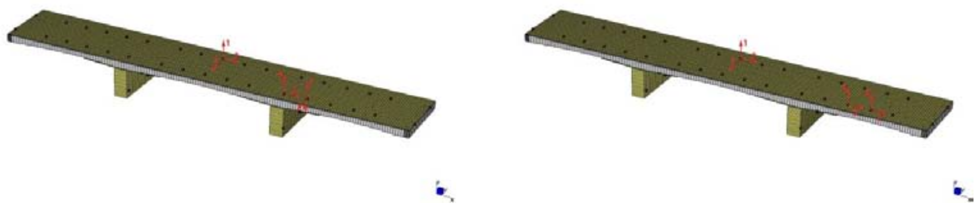


Figure 6.10: Setup 8 (left) and setup 9 (right).

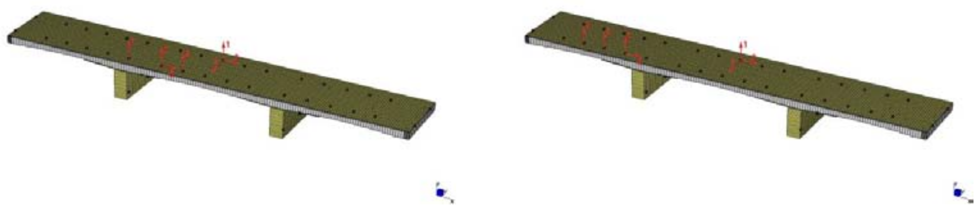


Figure 6.11: Setup 10 (left) and setup 11 (right).

Dynamic analysis: Artemis (FDD) vs. FEM

The close affinity between the results provided by the FEM model and those provided by FDD (using Artemis) can be observed by comparing the mode shapes for the first six modes, presented in the images below. In particular, from both evaluations, experimental and numerical, the first vibration mode is of bending type in the vertical direction, the second and the third are of bending type in the longitudinal direction, the mode 4 is torsional and involves only the central span, while the modes 5 and 6 are always of torsional type, but involving the two lateral spans. [8].

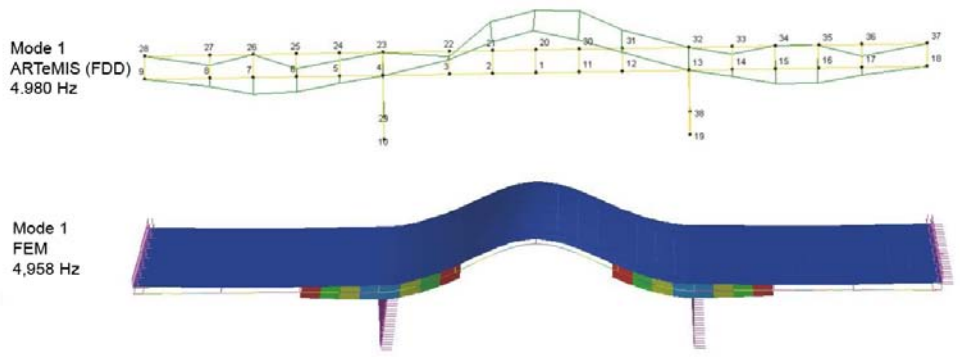


Figure 6.12: Mode 1.

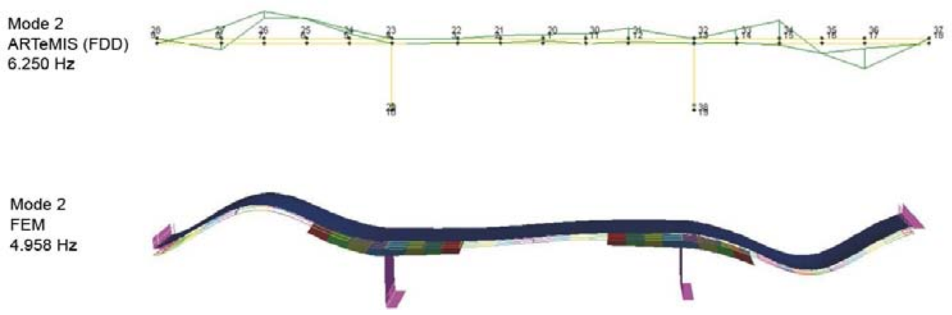


Figure 6.13: Mode 2.

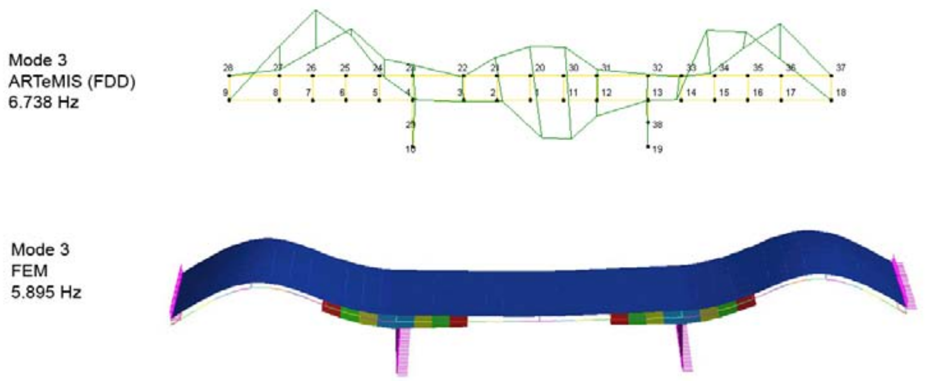


Figure 6.14: Mode 3.

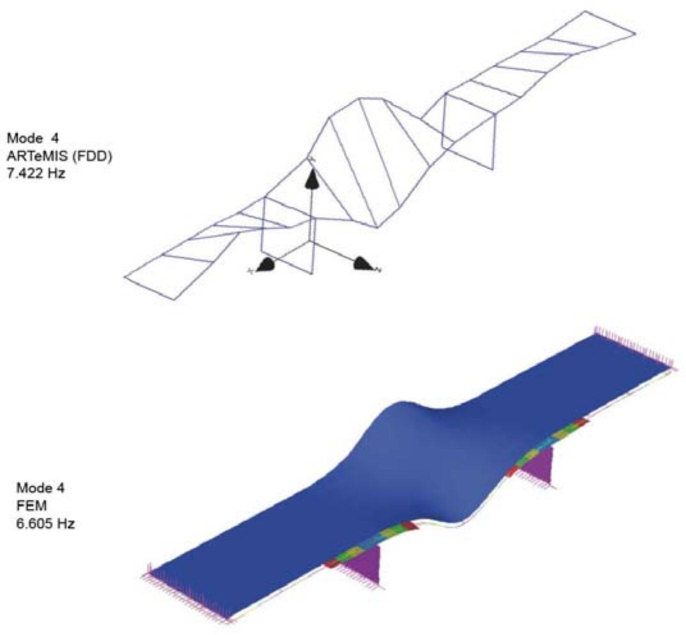


Figure 6.15: Mode 4.

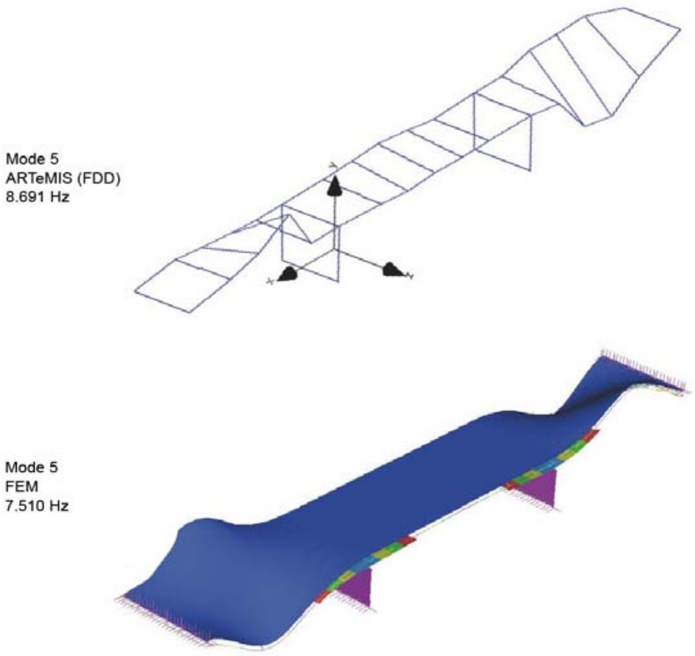


Figure 6.16: Mode 5.

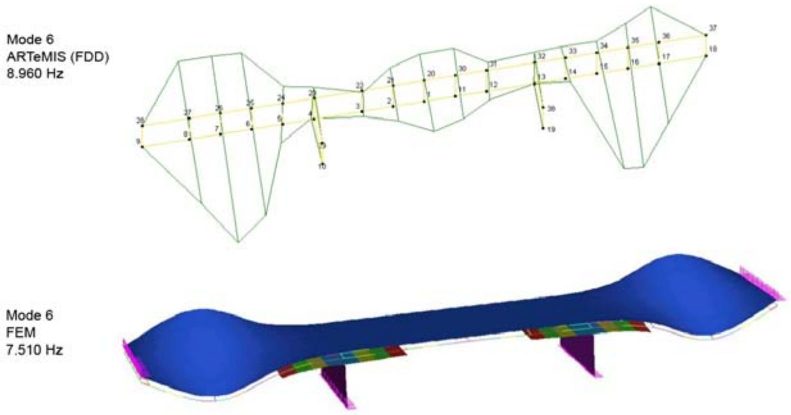


Figure 6.17: Mode 6.

6.5 Dynamic monitoring

The SHM system has been designed and installed on the Ponte Nuovo in order to monitor its mechanical behaviour. The final aim is the acquisition of the vibrational characteristics of the monument by means of acceleration transducers The acquired data are constantly related to the environmental parameters (temperature and relative humidity).

Insitutions involved	UNIPD
Period of implementation	May 2012 - active
Date of activation	20/5/2012
Dynamic System	6 Uni-axial Accelerometers Continuous Dynamic monitoring system
Static System	7 Strain gauges 16 Displacement transducers
State of the system	Active
Planned termination line	Undefined

Table 6.4: Details for SHM on Ponte Nuovo

The evaluation of the measured quantities, and in particular their changes over time, allows having useful indications in the definition of the structural behaviour and in the determination of the presence or occurrence of damage's phenomena.
Important information are currently recorded by the monitoring system, in relation to the dynamic response of the bridge, evaluating the accuracy of the adopted numerical models on the basis of the actual behaviour, also in case of possible seismic events.
The installation of the monitoring system was carried out in March 2012. The preliminary phase before the installation consisted in: (i) design of the system (hardware - types of sensors and acquisition units); (ii) development of appropriate software in relation to the selected monitoring strategy; (iii) choice of the system's layout and the points of structural control according to the outcomes of reference numerical models, the survey of the damage pattern and the results of dynamic identification tests of the structure. The system is composed of six uniaxial accelerometers (transducers acceleration), by sixteen linear potentiometers (displacement transducers) and from seven sensors strain gauges for reading the deformations (one of which is to "control" for the reading of the deformation due to the temperature and humidity) [34]. In Figure 6.18 is summarized the positioning of all the sensors and the control unit. The displacement transducers and sensors strain gauges were placed on all main beams at the center-line of the various spans, which realizes the max-

imum deformation. The acceleration sensors are installed at the middle of the bridge as a reference and then in the middle of the side spans, in order to grasp the global dynamic behavior of the structure and the change of the natural frequencies during earthquakes or heavy traffic. The installed dynamic monitoring system allows the analysis of a large amount of data taking into account two main aspects: (i) daily extraction of the fundamental modal parameters; (ii) registration and analysis of possible seismic events.
Significant effort is devoted to the analysis of recorded data. In particular the research activities are currently focused on the implementation in MATLAB environment of automatic and semi-automatic procedures for both static and dynamic data processing, applicable to many types of structural systems. As it was previously introduced, a monitoring system to observe static parameters (such as crack opening, displacements..) is in fact also installed on the structure, but it is not considered within this thesis.
The automatic procedure applied to static data elaborates a standard .txt file acquired by the static system and create automatically a series of graphs, representing the variation over time of the monitored parameters and correlating them with temperature and humidity variations. The algorithm allows also applying corrections on acquired data in order to remove errors caused by human interaction or system's malfunctioning [14], [31].
The program **AtOMA** allows to perform automatic dynamic monitoring of the structure without human intervention, by setting scheduled analysis.
Further developments will include automatic damage reporting and online availability of data on the current state of the structure.

6.5.1 Global results of the monitoring

The automatic procedure applied to dynamic data elaborates the acquisition files and automatically estimates and extracts modal parameters from measured vibrations. Four months of measurements were considered, consisting in 855 measurements files.
The damping ratio are affected by a clear uncertainty in the calculation for

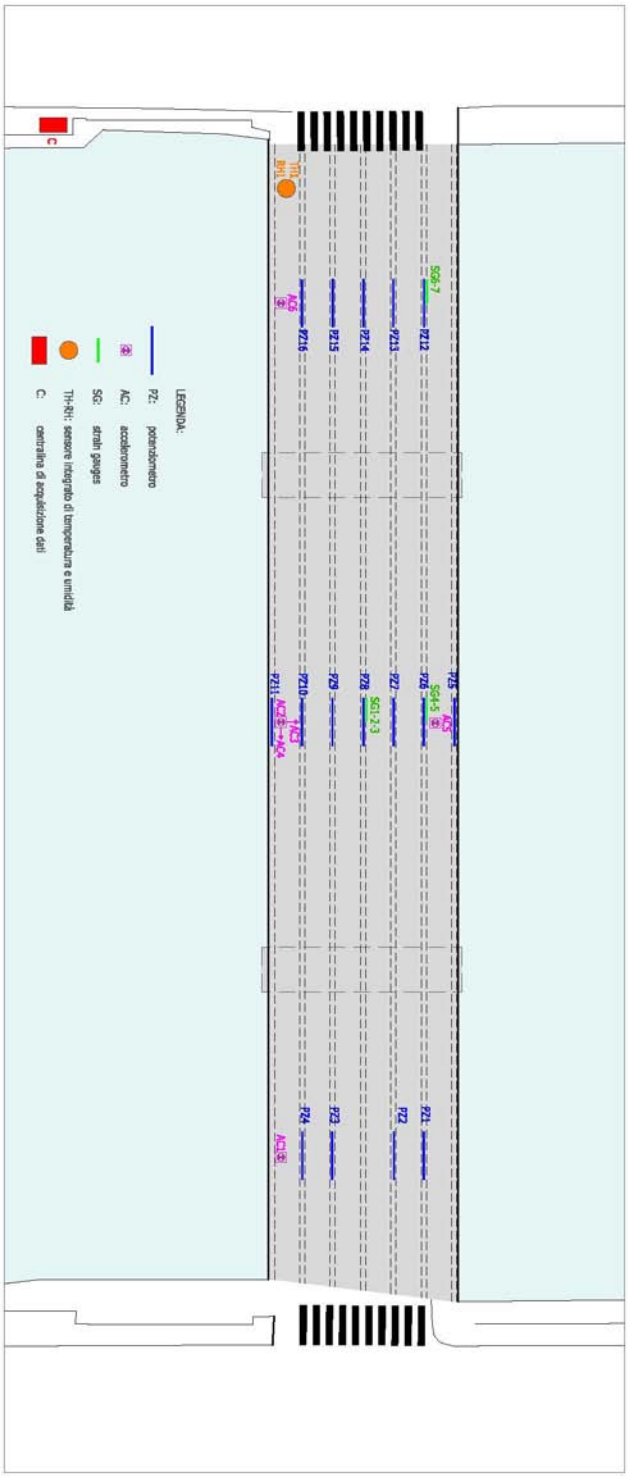


Figure 6.18: Schematic summary of the positioning of the instrumentation.

which they fluctuate within a certain range. The average trend, as well as the standard deviation remains almost constant over time.
The monitoring results are shown in Tables 6.6 and 6.7. These results have been obtained by excluding the values considered as statistical outliers (using the same method previously introduced for the box-plot diagram). It is very important to notice two parameters: the standard deviation and the success rate.
Considering the natural frequencies in Table 6.6, Modes 1, 2, 4, 5, and 6, exhibit a low standard deviation and a high success rate (the number of tests completed in total, excluding the outliers) and this indicates a good reliability for these results. It has been, on the contrary, not always possible to correctly identify mode 3, which shows a lower (but not insufficient) success rate. This is probably due to the positioning of the accelerometers.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
f_{max}	5,029	6,226	6,909	7,788	11,743	12,598
f_{min}	4,883	5,786	6,543	7,495	11,572	11,865
f_{mean}	4,954	6,090	6,809	7,560	11,700	12,051
std.dev	0,035	0,126	0,098	0,071	0,054	0,174
success%	95%	89%	60%	95%	90%	95%

Table 6.5: Frequency results of the monitoring using AutoEFDD.

These values can be compared to those obtained through the preliminary identification of the structure using *ARTeMIS Extractor*:

Mode	f_{mean} (Hz) (AutoEFDD)	f_{mean} (Hz) (ARTeMIS)	Relative error
1	4,954	4,980	0,5%
2	6,090	6,250	2,6%
3	6,809	6,738	1,0%
4	7,560	7,422	1,8%
-	-	8,691	-
-	-	8,960	-
5	11,700	11,600	0,9%
6	12,051	-	-

Table 6.6: Comparision between the frequencies obtained using AutoEFDD and ARTeMIS.

The structural modes at 8,691 Hz and at 8,960 Hz have not been identified by AutoEFDD, probably due to the limited number of channels available for the monitoring. The identification performed with Artemis took advantage, in fact, of a much higher number of measuring instruments.
Analogously, for Table 6.7 all calculated damping ratio parameters for the first 6 modes can be considered as structural, since their value is is minor than 10-20%. Mode 1 and mode 5 and 6 are the ones that exhibit less dispersed value, having a low standard deviation. Damping ratios calculated for mode 2, 3 and 4 are, on the contrary, more dispersed. The success rate for mode 3 is around 54%, which is due to the difficulty in identifying the structural mode (the success rate for the natural frequency identification for mode 3 is 60%).

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
ζ_{max}	2,393%	5,974%	4,782%	4,782%	2,519%	2,659%
ζ_{min}	0,577%	0,435%	0,277%	0,349%	0,270%	0,199%
ζ_{mean}	1,532%	2,994%	1,816%	2,367%	1,132%	1,123%
std.dev	0,375%	1,252%	1,049%	0,928%	0,470%	0,549%
success%	93%	86%	54%	96%	84%	92%

Table 6.7: Damping results of the monitoring using AutoEFDD.

The trend of the natural frequencies and damping ratios calculated using the monitoring tool **AtOMA**-AutoEFDD is shown in the following figures. In Figure 6.19 the trend over time of the natural frequencies calculated for Ponte Nuovo is plotted. The trend of natural frequencies shows an almost constant behaviour during the months of observation. The lower frequencies show a greater stability while the higher ones are more dispersed.

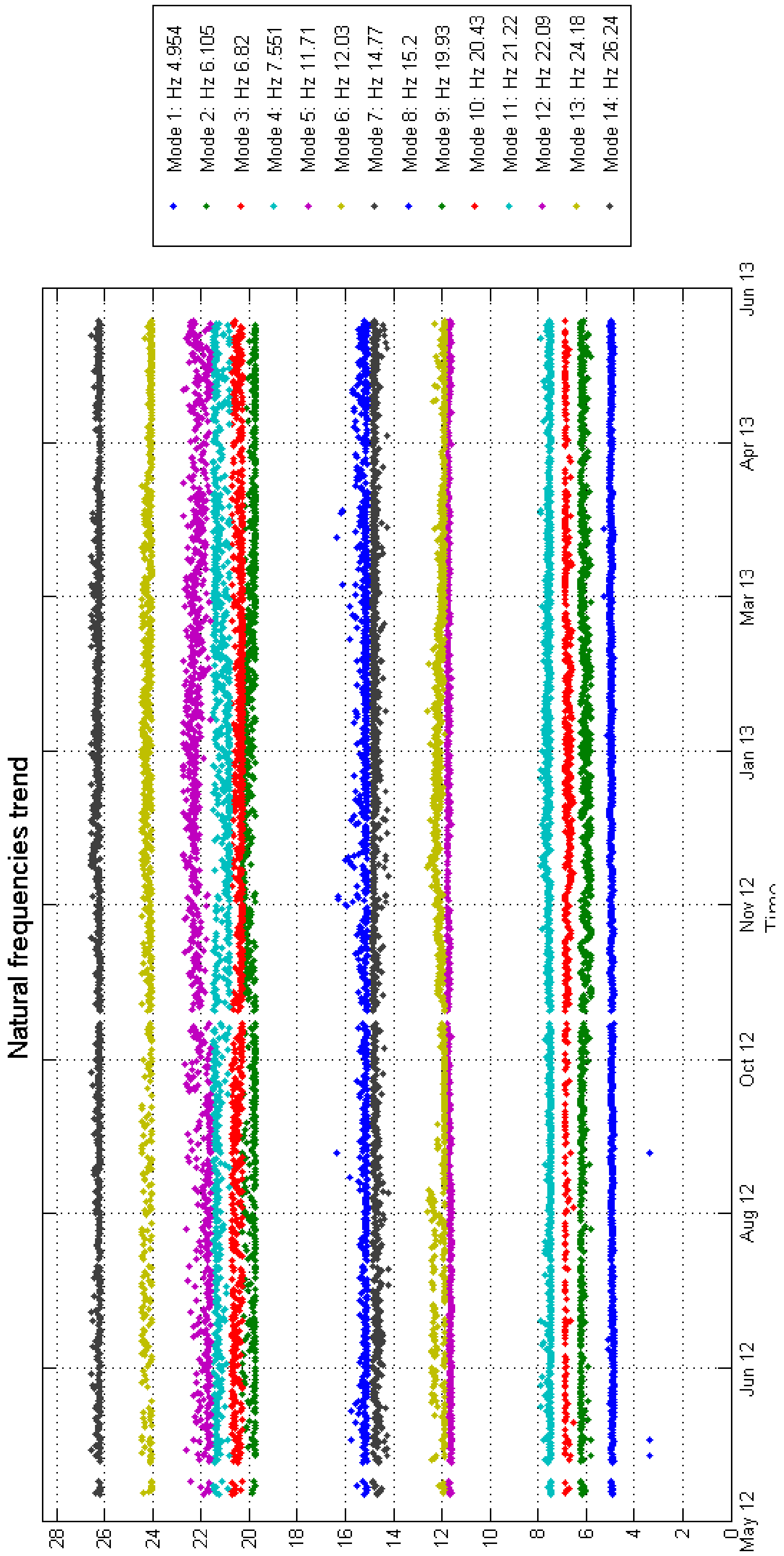


Figure 6.19: Trend over time of the natural frequencies for Ponte Nuovo (May. 12 - Jun. 13).

Figure 6.20 shows an overview of the modal parameters calculated during the whole monitoring. The structural modes with a low dispersion are considered correctly identified. From this figure it is not possible, however, to understand whether the dispersion is due to seasonal variations of the structure or if it is due to a weak structural mode.

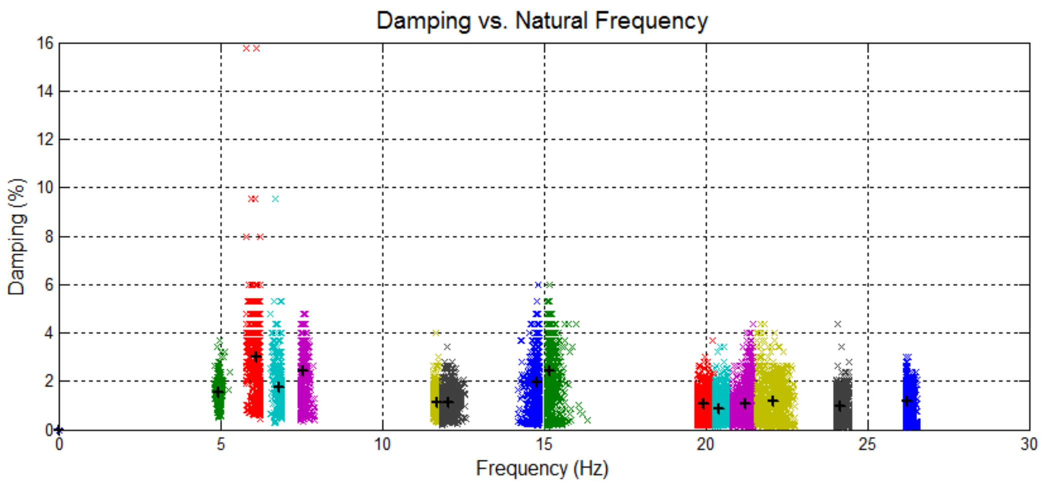


Figure 6.20: Damping ratios vs natural frequencies (May. 12 - Jun. 13).

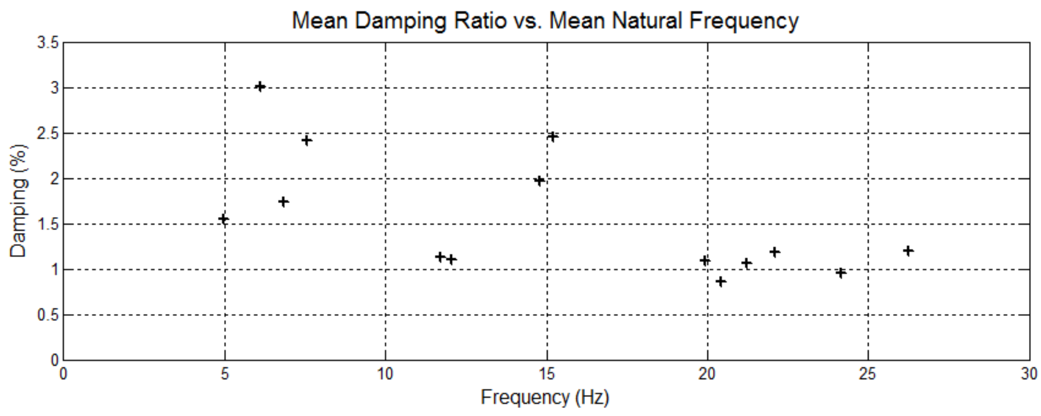


Figure 6.21: Mean values of the modal parameters (May. 12 - Jun. 13).

As previously introduced, the box-plot provides a statistical description of the distribution of the values. Since a box-plot with a narrow inter-quartile range (IQR) indicates a low dispersion of the results, modes 1, 4, 5, 6 appear to be well estimated.

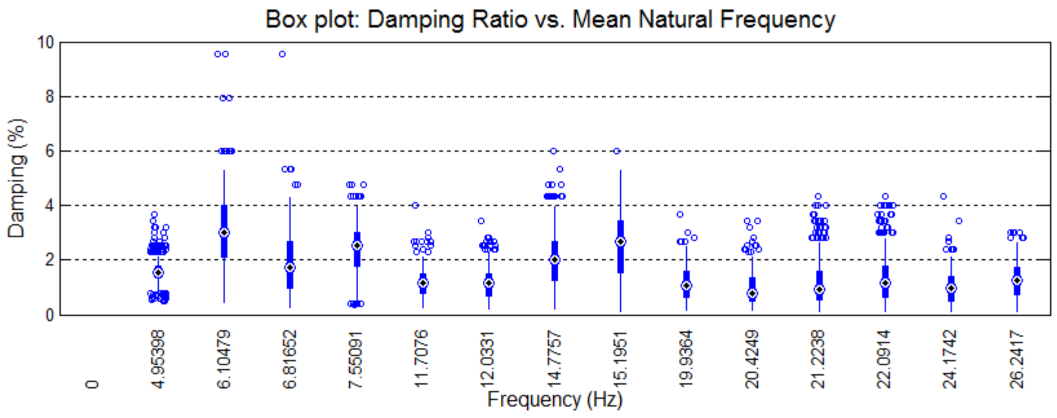


Figure 6.22: Box-plot of the damping ratios (May. 12 - Jun. 13).

Figures 6.23, 6.24 and 6.25 show the developing over time of the damping ratio parameter for each structural mode (from mode 1 to mode 6). A seasonal trend variation is visible for all the structural modes. Modes 1, 2, 3, 4, 6 show, on average, the highest damping ratio values in the winter period and then settle again on the previous values, while mode 5 exhibits lower values in the winter period.

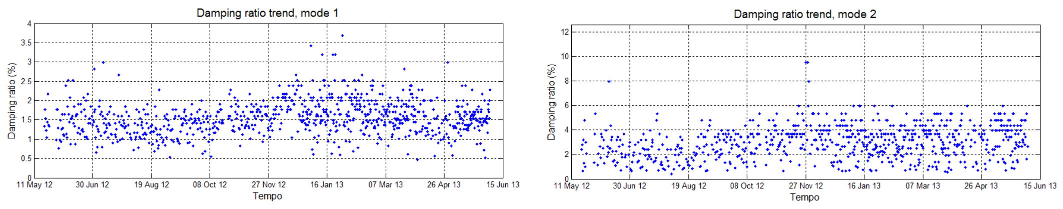


Figure 6.23: Trend over time of damping ratios for mode 1, 2 (May. 12 - Jun. 13).

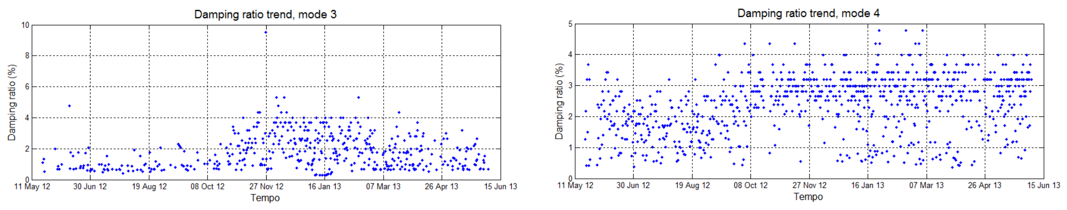


Figure 6.24: Trend over time of damping ratios for mode 3, 4 (May. 12 - Jun. 13).

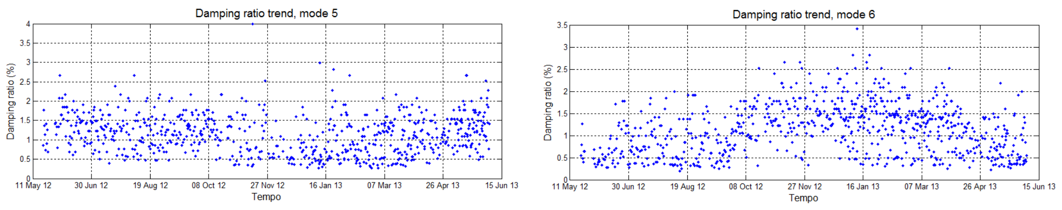


Figure 6.25: Trend over time of damping ratios for mode 5, 6 (May. 12 - Jun. 13).

6.5.2 Validation of the results

As previously introduced, the dynamic monitoring test is performed using **AtOMA**'s AutoEFDD procedure, which implements an Enhanced Frequency Decomposition method.

In order to validate the results, another method is used to extract the modal parameters of the structure: *Stochastic Subspace Identification*, through the **AtOMA**'s AutoSSI procedure. Once the results provided by AutoEFDD are available, the extracted structural modes are searched inside the results given by SSI. While high success rates mean that the structural mode is considered to be valid, low success rates do not automatically imply the invalidity of the result, since SSI itself is a method that requires a calibration. Moreover, due to the slowness of the SSI algorithm, it was not possible to carry out the analysis of all the files of monitoring, therefore only a small sample of the total has been considered.

The results, for the first 6 structural modes, given by AutoSSI procedure are displayed in Tables 6.8 and 6.9.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
f_{max}	5,009	6,315	7,186	8,242	11,866	12,546
f_{min}	4,908	5,736	6,643	7,281	11,425	11,995
f_{mean}	4,944	6,062	6,938	7,649	11,660	12,229
$std.dev$	0,027	0,192	0,138	0,352	0,110	0,129
$success\%$	97,1%	65,7%	82,9%	94,3%	71,4%	97,1%

Table 6.8: Frequency results of the monitoring using AutoSSI.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
ζ_{max}	2,472%	7,794%	6,415%	4,731%	2,832%	2,992%
ζ_{min}	0,806%	3,124%	2,435%	1,118%	0,658%	1,182%
ζ_{mean}	1,542%	4,898%	4,354%	2,486%	1,580%	2,014%
$std.dev$	0,377%	1,107%	1,146%	0,919%	0,606%	0,497%
$success\%$	100,0%	65,7%	82,9%	94,3%	71,4%	91,4%

Table 6.9: Damping results of the monitoring using AutoSSI.

The validity of the natural frequencies extracted using AutoEFDD is clear for most of the structural modes: modes 1, 3, 4, 6 exhibit in fact a high success rate. Modes 2 and 5 were not always identified by SSI, but the validation is overall positive, since they were correctly identified in more than 65% of the monitoring files considered.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
$f(EFDD)$	4,954	6,090	6,809	7,560	11,700	12,051
$Success\%(EFDD)$	95,3%	89,1%	59,9%	95,1%	89,7%	95,3%
$f(SSI)$	4,944	6,062	6,938	7,649	11,660	12,229
$Success\%(SSI)$	97,1%	65,7%	82,9%	94,3%	71,4%	97,1%
$Error\ \epsilon$	0,2%	0,5%	1,9%	1,2%	0,3%	1,5%

Table 6.10: Comparision of the natural frequencies extracted using AutoEFDD and AutoSSI.

The parameters of damping ratio calculated with SSI are of the same order of magnitude as those calculated by the method EFDD. The estimates of the damping ratio for modes 1, 4, 5 are comparable with those obtained using EFDD, while those calculated for modes 2, 3 and 6 present much higher values: these values are probably attributable to the reduced number of files on which the analysis was performed.

	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
$\zeta(EFDD)$	1,532%	2,994%	1,816%	2,367%	1,132%	1,123%
$Success\%(EFDD)$	92,9%	86,0%	53,7%	96,3%	84,1%	91,6%
$\zeta(SSI)$	1,542%	4,898%	4,354%	2,486%	1,580%	2,014%
$Success\%(SSI)$	100%	65,7%	82,9%	94,3%	71,4%	91,4%
$Error\ \epsilon$	0,6%	48,2%	82,3%	4,9%	33,1%	56,8%

Table 6.11: Comparision of the damping ratios extracted using AutoEFDD and AutoSSI.

In summary, the mean values of the modal parameters calculated using the two methods are shown in Tables 6.10 and 6.11. For each pair of values of the relative error was calculated as the ratio of the absolute error and the mean of the two values.

6.6 Conclusions

In this chapter an application regarding the Ponte Nuovo del Popolo in Verona was carried out. The purpose of this application was to consider an existing structure, on which previous dynamic structural identification tasks were performed, in order to observe the validity of the results given by **SIM/AtOMA** in comparison to those provided by other methods and commercial software.
Two structural modes previously identified with ARTeMIS using different monitoring setups [8] were not identified by the subsequent monitoring. In order to correctly identify these two modes it was probably necessary to install other measuring instruments in the two terminal spans of the bridge.
Furthermore, except for these two structural modes which were not identified, the results were proven to be reliable and the analysis was extended and completed over one year of measurements using AutoEFDD and AutoSSI procedures.
Overall, the validation of the results provided by the AutoEFDD procedure

can be considered satisfied, since the results given by the various methods are consistent to each other. While the natural frequencies identified using AutoEFDD and AutoSSI are perfectly corresponding, damping ratios show some variability between the two methods, but it is however considered to be acceptable.
The monitoring of the bridge displayed a constant behavior of the natural frequencies over a complete year, while the damping ratios exhibited some seasonal change. In fact, for most of the structural modes, the trend of the evaluated damping ratios exhibits an increment in the winter season and returns to its original values during spring.

Chapter 7

Conclusions

Structural identification (St-Id) of constructed systems has become, over the last few decades, a relevant engineering field. As seen in Chapter 2, St-Id rests its foundation on the concept that for a mathematical model is a description of reality, it must be validated on the basis of observations of the physical world. In this case, the interest of this thesis was focused on the observation of the dynamic characteristics of existing structures by means of various methods. A classic method for dynamic identification of a structure lies in modal analysis, which allows to extract the dynamic parameters of the structure, such as its natural frequencies, damping ratios, mode shapes, from experimental measurements.

In contrast to the classical experimental modal analysis, which requires the excitation of the structure by means of particular instruments, another, recently developed, type of analysis has been considered, the operational modal analysis (OMA). OMA observes the response of the structure subjected to the "natural" excitations, such as wind, vehicular traffic, seismic micro vibrations and so on. The advantages that this technique have been presented, among which the fact that it is no longer necessary to interrupt the functionality of the structure under examination and this allows the execution of monitoring operations over time and makes possible a variety of applications, such as Structural Health Monitoring. Furthermore, this method is the only one that provides a *global* dynamic characterization of the *real* structure: in fact FE models provide global information of a structure's model, not of the structure

itself, and static measurements provide real information but of local nature. The fact that it is not necessary to stimulate the structure with specific devices, makes the application of these methods highly advantageous from an economic point of view.
Various operational modal analysis methods available were illustrated, focusing on two methods in particular, Frequency Domain Decomposition and Stochastic Subspace Identification. In the context of this thesis, it was in fact developed an application that allows the performance of activities of dynamic identification by means of these two methods. This application is called *Structural Identification and Monitoring / Automatic tool for Operational Modal Analysis* (**SIM/AtOMA**).
The work done for this thesis consisted in expanding the implementation of the FDD method, previously started in the context of the Department of Civil, Construction and Environmental Engineering of Padua (DICEA), in order to enable the extraction of the damping ratio parameter, and subsequently carry out the implementation of the two methods (EFDD and SSI) within a program with a graphical interface, which can be used for the identification and automatic monitoring of existing structures.
The mode of operation of this program were presented in Chapters 4, 5 and 6. In chapter 4, the characteristics of the program were discussed from a user point of view. The interactions required to perform each task and the outputs were explained, thus all the parameters, buttons and Figures were documented in detail.
Furthermore, a tutorial for the user was provided in this chapter, which consisted in the dynamic identification of a bridge over river Mincio, situated in the municipality of Peschiera sul Garda. First a description of the structure was provided in order to contextualize the problem. Subsequently the characteristics of the monitoring system were discussed. Then the tasks performed to extract the modal parameters using the program were described in detail. The results provided by the two methods (AutoEFDD and AutoSSI) have proven to be consistent and are therefore considered valid to describe the dynamic behavior of the structure.
In chapter 5, the program has been explained from the point of view of pro-

gramming code that defines its behavior. All major functions have been described, indicating the meaning of the most important steps.
In chapter 6, an application of St-Id carried out using **SIM/AtOMA** was presented. The analysis of this bridge was based on a dynamic structural characterization carried out in the context of previous thesis and was developed further by means of the tools provided by the program **AtOMA**. This has allowed us to further confirm the validity of the methods used, as an operation of dynamic identification had been previously carried out by means of a commercial software which ARTeMIS. This previous analysis had shown the correlation between the structure's actual vibration modes and those exhibited by the FE model. The results given by AutoEFDD and AutoSSI are consistent with the previous analysis and have allowed to observe the development of the modal parameters of the structure for over a year of measurements.
By means of the two applications considered, was therefore demonstrated the ability of **AtOMA** to provide accurate results of the actual dynamic behavior of a structure.
St-Id is, however, a very vast engineering field, whose theoretical basis range from signal analysis to structural mechanics. A brief overview of the required theoretical knowledge necessary to define the operations performed in a St-Id analysis has been provided in Chapter 2.
SIM/AtOMA allows to perform structural dynamic identification of many types of structures with an easy, user friendly interface, which informs the user on the intermediate steps of the analysis in order to let him calibrate the parameters to achieve the best possible results. The results given by the AutoEFDD procedure can be validated inside the program itself by using AutoSSI procedure, which makes it an independent tool for dynamic identification.
At the present time, the on-site installation of the program **AtOMA**, makes it possible to carry out operations not only of dynamic identification, but also of automatic dynamic monitoring. By means of internet connections, real-time information on the state of the structure are available without the need to travel to the location or to transfer measurement files, which may have dimensions not suitable for transfer over the network.
However, there is much room for the further developments of the program.

For example, the enabling of the detection of any damage states, identified by changes in the modal parameters of the structure, will optimize the maintenance and safety of the structure by submitting via e-mail warning in case of damage. The development of a spatial mode shape recognition will allow to directly correlate the results of the experimental testing with the mode shapes given by a FE model, facilitating the process of model updating. The implementation of ARX methods will allow to eliminate the influence of temperature and humidity from the results (which was clearly visible in the damping results obtained in the monitoring of Ponte Nuovo, in Chapter 6).

References

[1] Allemang R.J.,Brown D.L., "Experimental modal analysis", Chapter 21.

[2] Andersen P., "Identification of Civil Engineering Structures using Vector ARMA Model", *Ph.D. Thesis. Aalborg University. Denmark*, 1997

[3] Andersen P., Brinker R., "Estimation of Modal Parameters and Their Uncertainties.", *Proceeding of the 17th IMAC*, 1999

[4] ASCE SEI Committee on Structural Identification of Constructed Systems "Structural Identification (St-Id) of Constructed Facilities", *American Society of Civil Engineers, Structural Engineering Institute*, 2011.

[5] Baker B., "How delta-sigma ADCs work", *Texas Instruments Incorporated,Analog Applications Journal*, 2011

[6] Bendat J.S., Piersol A.G., "Random Data: Analysis and Measurement Procedures", Fourth Edition, *Wiley Series in Probability and Statistics*, 2010

[7] Bilošová A., "Modal Testing", 2011.

[8] Bisson A., "Caratterizzazione strutturale ed identificazione dinamica di due ponti stradali", 2011.

[9] Brinker R., Andersen P., "Understanding Stochastic Subspace Identification", *Proceedings of the 24th International Modal Analysis Conference (IMAC), St. Louis, Missouri*, 2006

[10] Brinker R., Andersen P., Jacobsen N.J., "Automated Frequency Domain Decomposition for Operational Modal Analysis", *Proceedings of*

The 25th International Modal Analysis Conference (IMAC), Orlando, Florida, 2007

[11] Brinker R., Andersen P., Kirkegaard P.H., Ulfkjaer J.P., "Damage Detection in Laboratory Concrete Beams", *Proceedings of The 13th International Modal Analysis Conference (IMAC)*, 1995

[12] Brinker R., Ventura C., Andersen P., "Damping Estimation by Frequency Domain Decomposition", *Proceedings of the 19th International Modal Analysis Conference (IMAC), Kissimmee, USA*, 2001

[13] Brinker R., Zhang L.M., P. Anderson, "Modal Identification from Ambient Response using Frequency Domain Decomposition", *Procedures of the 18th IMAC, San Antonio, TX, USA*, 2000

[14] Caldon M., "Un algoritmo per l'analisi automatica dei dati di monitoraggi strutturali. Applicazione a tre casi studio", *Master Degree Thesis, University of Padova*, 2012

[15] Cauberghe B., "Applied frequency-domain system identification in the field of experimental and operational modal analysis", *PhD Thesis, Vrije Univeriteit Brussel* 2004.

[16] Cooley J.W., Tuckey J.W., "An Algoritm for the Machine Calculation of Complex Fourier Series", *Mathematics of computation* 1965.

[17] Cunha A., Caetano E, "Experimental Modal Analysis of Civil Engineering Structures", *Sound and Vibration*, June 2006.

[18] Doebling S.W., Farrar C.R., Prime M.B., Shevitz D.W., "Damage Identification and Health Monitoring of Structural and Mechanical Systems from Changes in Their Vibration Characteristics: A Literature Review", *Los Alamos National Laboratory*, 1996.

[19] Ewins D. J., "Modal Testing - Theory and Practice", *Research Studies Press*, 1984.

[20] Gade S., Møller N.B., Herlufsen H., Konstantin-Hansen H., "Frequency Domain Techniques for Operational Modal Analysis", *Brüel & Kjær*

Sound and Vibration Measurements A/S, *IMAC-XXIV: Conference & Exposition on Structural Dynamics - Looking Forward: Technologies for IMAC*, 2006

[21] Hall S.R., "The Effective Management and Use of Structural Health Data" *Proceedings of the 2nd International Workshop on Structural Health Monitoring*, 1999.

[22] He, J., Fu, Z.F., "Modal Analysis" *Butterworth-Heinemann*, 2001.

[23] Hoon S., Farrar C., Hemez F.M., Shunk D.D., Stinemates D.W., Nadler B.R., Czarnecki J.J., "A Review of Structural Health Monitoring Literature: 1996âĂŞ2001" *Los Alamos National Laboratory*, 2004.

[24] Islami K., "Enhanced system identification and automatic SHM of bridge structures", *Ph.D. Thesis, University of Padua, Italy*, 2013.

[25] Jacobsen N.-J., Andersen P., Brinker R., "Using Enhanced Frequency Domain Decomposition as a Robust Technique to Harmonic Excitation in Operational Modal Analysis", *Proceedings of The 2nd International Operational Modal Analysis Conference (IOMAC), Copenhagen, Denmark*, 2007.

[26] Jacobsen N.-J., Andersen P., Brinker R., "Applications of Frequency Domain Curve-fitting in the EFDD Technique", *Proceedings of the 26th International Modal Analysis Conference (IMAC) Orlando, Florida USA*, 2008.

[27] James G.H., Lauffer J.P., Nard A.R., "Modal Testing Using Natural Excitation", *Proceeding of the 10th IMAC. San Diego. CA. USA*, 1992

[28] Kessler S.S., "Structural Health Monitoring in Composite Materials Using Lamb Wave Methods", *Technology Laboratory for Advanced Composites Department of Aeronautics and Astronautics Massachusetts Institute of Technology*

[29] Koutsovalisis P., Beitelschmidt M., "Model Reduction of Large Elastic Systems, A Comparison Study on the Elastic Piston Rod", *12th World Congress in Mechanism and Machine Science, IFToMM*, 2007.

[30] Ljung L., "System Identification: Theory for the User", *Prentice Hall*, 1987.

[31] Lorenzoni F., "Integrated methodologies based on structural health monitoring for the protection of cultural heritage buildings", *PhD Thesis, IUAV*, 2013.

[32] Masjedian M.H., Keshmiri M., "A Review on Operational Modal Analysis Researches: Classification of Methods and Applications", *IOMAC 2009, 3rd International Operational Modal Analysis Conference*, 2009

[33] Moon F.L., Aktan A.E., "Impacts of Epistemic (Bias) Uncertainty on Structural Identification of Constructed (Civil) Systems", *The Shock and Vibration Digest, Vol. 38, No. 5*, 2006

[34] Modena C., Cappellin D., Islami K., Caldon M., "Relazione tecnica sulle attivitá di monitoraggio strutturale del Ponte Nuovo del Popolo a Verona, Maggio 2012 - Maggio 2013", *Dipartimento di Ingegneria Civile Edile e Ambintale, Universitá di Padova*, 2013

[35] Moore S.M, Lai J.C.S, Shankar K., "ARMAX Modal Parameter Identification in the Presence of Unmeasured Excitation-l: Theoretical Background.", *University of New South Wales. Canberra. Australia.*, 2006

[36] Piovesan D., "Dynamic system identification of bridges: development of automatic algorithms in the frequency domain for continuous monitoring.", *Univerisitá degli studi di Padova*, 2013.

[37] SM Ingegneria s.r.l, "Ponte sul fiume Mincio a Peschiera, Relazione di verifica sullo stato di fatto.", *Ponte sul fiume Mincio lungo l'autostrada A4 in Comune di Peschiera del Garda. Rilievo dello stato di danno, verifiche e proposte di interventi urgenti per l'adeguamento del ponte*, 2011.

[38] SM Ingegneria s.r.l, "Relazione interpretativa del monitoraggio delle deformazioni dinamiche", *Ponte sul fiume Mincio lungo l'autostrada A4 in Comune di Peschiera del Garda. Rilievo dello stato di danno, verifiche e proposte di interventi urgenti per l'adeguamento del ponte*, 2013.

[39] Van Overchee P., De Moor B., "Subspace identification for linear systems - Theory, Implementation, Applications", *Kluwer Academic Publishers* 1996.

[40] Ventura C.E., Tomas H., "Structural Assessment by Modal Analysis in Western Canada", *Procedures of the IMAC XV, Orlando, Florida* 1997.

[41] Zhang L., Brincker R., Andersen P., "An Overview of Operational Modal Analysis: Major Development and Issues", *Proceedings of the 1st International Operational Modal Analysis Conference (IOMAC), Copenhagen, Denmark*, 2005

[42] Zhang L.-M, Wang T., Tamura Y., "Frequency-spatial Domain Decomposition Technique with Application to Operational Modal Analysis of Civil Engineering Structures", *Proceedings of the 1st International Operational Modal Analysis Conference (IOMAC), Copenhagen, Denmark*, 2005

Appendix A

Complete Matlab code of the program

A.1 AutoEFDDgui.m

```
function varargout = AutoEFDDgui(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @AutoEFDDgui_OpeningFcn, ...
    'gui_OutputFcn',  @AutoEFDDgui_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before AutoEFDDgui is made visible.
```

```
function AutoEFDDgui_OpeningFcn(hObject, eventdata, handles,
    varargin)
handles.output = hObject;
guidata(hObject, handles);
setappdata(0,'usingtemplate',0);
setappdata(0,'scheduleron',0);
function varargout = AutoEFDDgui_OutputFcn(hObject, eventdata,
    handles)
varargout{1} = handles.output;



% --- Executes on button press in browsenewfdd.
function browsenewfdd_Callback(hObject, eventdata, handles)
folder_name = uigetdir('','Please select the folder with the data
    file');
handles.fileList = getAllFiles(folder_name, 500);
guidata(hObject,handles);
set(handles.folder,'String',folder_name);
set(handles.startfdd,'Enable','on');
set(handles.statusS,'Visible','on','String','<--','BackgroundColor
    ',[0.941 0.941 0.941]);
set(handles.phase1,'BackgroundColor',[1 1 0]);
set(handles.phase2,'BackgroundColor',[0.941 0.941 0.941]);
set(handles.phase3,'BackgroundColor',[0.941 0.941 0.941]);

% --- Executes on button press in startfdd.
function [handles]=startfdd_Callback(hObject, eventdata, handles)
set(handles.statusS,'Visible','on','String','Wait..','
    BackgroundColor',[1 0 0]);

L=str2double(get(handles.L,'String'));
fs=str2double(get(handles.fs,'String'));
NFFT_all=str2double(get(handles.NFFT,'String'));
NFFT=NFFT_all(get(handles.NFFT,'Value'));
clear NFFT_all;
folder_name=get(handles.folder,'String');
fileList = getAllFiles(folder_name, 500);



canali=get(handles.channels,'String');
```

```
canali_ind=strfind(canali,',');
v(1)=str2num(canali(1:canali_ind(1)-1));
for i=2:length(canali_ind)
    v(i)=str2num(canali(canali_ind(i-1)+1:canali_ind(i)-1));
end
v(i+1)=str2num(canali(canali_ind(i)+1:length(canali)));

%%% Select only Fast file
j = 1;
for i= 1:length(fileList)
    isFast = strfind(fileList(i,1),'_Fas');
    emptyCell = cellfun(@isempty,isFast(1));
    if ( emptyCell == 0 )
        FastList(j,1) = fileList(i,1);
        j = j+1;
    end
end
numfiles = j-1;ss
set(handles.status1,'String',[num2str(numfiles) ' dynamic files
   found']);
hstr0=[num2str(numfiles) ' dynamic files found' char(10) '
   Processing FDD. Please wait...'];
FastNameList = FastList;
FastList = cell2struct(FastList, 'name', numfiles);
%% Loop over in this Folder
h = waitbar(0,hstr0);

for ifast = 1 : size(FastList,1)
    waitbar((ifast-1)/size(FastList,1),h,hstr0);
    tic;
% Look at [34]
    giro=toc;
    Monitoraggi_Mancanti=size(FastList,1)-ifast;
    hstr0=['Now processing FDD. ' datestr(datenum(0,0,0,0,0,giro*
   Monitoraggi_Mancanti),'HH:MM:SS') ' left.'];

end
waitbar(1,h,'Done.');
```

```
if round(size(FastNameList,1)/3)>3
    setifast=round(size(FastNameList,1)/3);
else
    setifast=3;
end
set(findobj('Tag','ifast'),'String',num2str(setifast));
setappdata(0,'FDD_location','FDD.mat');
set(findall(handles.uipanel7,'-property','enable'),'enable','on');
set(findall(handles.uipanel5,'-property','enable'),'enable','off')
    ;
set(handles.frequencies,'Enable','on');
set(handles.statusF,'Visible','on','String','<--','BackgroundColor
    ',[0.941 0.941 0.941]);
set(handles.statusD,'Visible','off');
set(handles.statusS,'String','OK','BackgroundColor',[0 1 0]);
set(handles.phase1,'BackgroundColor',[0 1 0]);
set(handles.phase2,'BackgroundColor',[1 1 0]);
set(findobj('Tag','saveres'),'Enable','off');
set(findobj('Tag','savetemplate'),'Enable','off');
set(findobj('Tag','savefddfile'),'Enable','on');

close(h);
hhh=handles;
clear handles hObject;
save FDD;

load (getappdata(0,'FDD_location'),'FastNameList');
set(findobj('Tag','ifast0'),'String',FastNameList,'Value',1);

if getappdata(0,'usingtemplate')==1
    if getappdata(0,'scheduleron')==1 %spostamento files
        archive=[getappdata(0,'fastdir') '\archive\'];
        if ~exist(archive,'file') mkdir(archive); end
        for i=1:length(FastNameList)    movefile(FastNameList{i},
    archive,'f'); end
    end
    handles=frequencies_Callback([],[], hhh);
    handles=damping_Callback([],[], handles);
    saveres_Callback([],[], handles);
    setappdata(0,'mergetemplate',1);
    merge;

end

% --- Executes on button press in browsefdd.
function browsefdd_Callback(hObject, eventdata, handles)
[FileName,PathName,FilterIndex] = uigetfile('*.mat','Please select
    the FDD.mat file.');
fdd_name=[PathName FileName];
set(handles.fddfile,'String',fdd_name);
set(handles.phase1,'BackgroundColor',[1 1 0]);
set(handles.phase2,'BackgroundColor',[0.941 0.941 0.941]);
set(handles.phase3,'BackgroundColor',[0.941 0.941 0.941]);
set(findobj('Tag','saveres'),'Enable','off');
set(findobj('Tag','savetemplate'),'Enable','off');
set(findobj('Tag','validation'),'Enable','off');
set(findobj('Tag','dampingvsfrequencies'),'Enable','off');
set(findobj('Tag','meanvsfrequency'),'Enable','off');
set(findobj('Tag','boxplot'),'Enable','off');
set(findobj('Tag','dmode'),'Enable','off');

% --- Executes on button press in loadfdd.
function loadfdd_Callback(hObject, eventdata, handles)
setappdata(0,'FDD_location',get(handles.fddfile,'String'));

guidata(hObject,handles);
FDD=getappdata(0,'FDD_location');
load(FDD, 'FastNameList');
if size(FastNameList,1)<6 && getappdata(0,'usingtemplate')~=1
    errordlg('Required at least 6 files to compute a brand new
   analysis.');
    set(handles.statusS,'String','<--','BackgroundColor',[0.941
   0.941 0.941]);
    return;
end
if round(size(FastNameList,1)/3)>3
    setifast=round(size(FastNameList,1)/3);
else
    setifast=3;
end
```

```
set(handles.ifast,'String',num2str(setifast));
set(findall(handles.uipanel7,'-property','enable'),'enable','on');
set(findall(handles.uipanel5,'-property','enable'),'enable','off')
    ;
set(handles.frequencies,'Enable','on');
set(handles.statusF,'Visible','on','String','<--','BackgroundColor
    ',[0.941 0.941 0.941]);
set(handles.statusD,'Visible','off');
set(handles.statusS,'String','OK','BackgroundColor',[0 1 0]);
set(handles.phase1,'BackgroundColor',[0 1 0]);
set(handles.phase2,'BackgroundColor',[1 1 0]);
set(findobj('Tag','saveres'),'Enable','off');
set(findobj('Tag','savetemplate'),'Enable','off');
set(findobj('Tag','savefddfile'),'Enable','on');
set(handles.frequenzetempo,'Enable','off');
set(handles.svbutton,'Enable','off');

load (getappdata(0,'FDD_location'),'FastList');
FL=struct2cell(FastList);
set(findobj('Tag','ifast0'),'String',FL,'Value',1);
if getappdata(0,'usingtemplate')==1
     handles=frequencies_Callback([],[], handles);
     handles=damping_Callback([],hObject, handles);
     saveres_Callback([],[], handles);
     setappdata(0,'mergetemplate',1);
     merge;
end
% --- Executes on button press in frequencies.
function [handles] = frequencies_Callback(hObject, eventdata,
    handles)
set(handles.statusF,'String','Wait..','BackgroundColor',[1 0 0]);

FDD=getappdata(0,'FDD_location');
load(FDD);
gg=2;

%% Calcolo del MAC tra matrice UUrif con un'altra UU scelta a
    parità di NFFT
% Inserimento dati
```

```
radio=get(handles.newresultsradio,'Value');
if  radio == 1      pp  = 1;
else                pp  = -1;
end

%Look at [34].

clearvars -except handles


if getappdata(0,'usingtemplate')==1
    load (getappdata(0,'savefile')); %carico il savefile del
   template
    load (getappdata(0,'indici')); %carico il file indici del
   template
else
    load savefile
    load indici
end
load(getappdata(0,'FDD_location'));
...
Result(~Result)=nan;
modelist=findobj('Tag','modelist');
for i=2:size(Result,2)
    listSTR{i-1}=['Mode ' num2str(i-1) ': ' num2str(trimmean(
   Result(:,i),20),4) ' Hz'];
end
set(modelist,'String',listSTR);

if getappdata(0,'usingtemplate')~=1
    set(findobj('Tag','svbutton'),'Enable','on');
end
setappdata(0,'Result_frequency',Result);
set(findall(findobj('Tag','uipanel5'),'-property','enable'),'
   enable','on');

set(findobj('Tag','frequenzetempo'),'Enable','on');
set(findobj('Tag','damping'),'Enable','on');
set(findobj('Tag','modetext'),'String',[num2str(size(Result,2)-1)
   ' structural modes found.']);
```

```
set(findobj('Tag','statusF'),'String','OK','BackgroundColor',[0 1
    0]);
set(findobj('Tag','statusD'),'Visible','on','String','<--','
    BackgroundColor',[0.941 0.941 0.941]);
set(findobj('Tag','saveres'),'Enable','off');
set(findobj('Tag','savetemplate'),'Enable','off');
set(findobj('Tag','validation'),'Enable','off');
set(findobj('Tag','dampingvsfrequencies'),'Enable','off');
set(findobj('Tag','meanvsfrequency'),'Enable','off');
set(findobj('Tag','boxplot'),'Enable','off');
set(findobj('Tag','dmode'),'Enable','off');

% --- Executes on button press in radionuovo.
function radionuovo_Callback(hObject, eventdata, handles)
set(handles.existingdatapanel,'Visible','off');
set(handles.newdatapanel,'Visible','on');
set(handles.ssipanel,'Visible','off');

set(handles.loadfdd,'Enable','off');
set(handles.startfdd,'Enable','on');
set(handles.radiofdd,'Value',0);
set(handles.radionuovo,'Value',1);
set(handles.ssiradio,'Value',0);
set(handles.browsenewfdd,'Enable','on');
set(handles.browsefdd,'Enable','off');

% --- Executes on button press in radiofdd.
function radiofdd_Callback(hObject, eventdata, handles)
set(handles.existingdatapanel,'Visible','on');
set(handles.newdatapanel,'Visible','off');
set(handles.ssipanel,'Visible','off');
set(handles.loadfdd,'Enable','on');
set(handles.startfdd,'Enable','off');
set(handles.radionuovo,'Value',0);
set(handles.radiofdd,'Value',1);
set(handles.ssiradio,'Value',0);
set(handles.browsenewfdd,'Enable','off');
set(handles.browsefdd,'Enable','on');

% --- Executes on button press in svbutton.
```

```
function svbutton_Callback(hObject, eventdata, handles)
FDD=getappdata(0,'FDD_location');
load(FDD);

ifast0  = get(handles.ifast0,'Value');
ifast   = str2double(get(handles.ifast,'String'));
ifast   = ifast0+ifast -1;
fmTOT   = zeros(floor(size(FastList)/(ifast-ifast0+1)),50);
appAX   = zeros(50,300,floor(size(FastList)/(ifast-ifast0+1)));
ZZ      = appAX;
YY      = appAX;
indiciTOT=zeros(floor(size(FastList)/(ifast-ifast0+1)),50);

UUrif(:,:)=Umodi(:,:,ifast0);
for j=ifast0+1:ifast
    for i=1:NFFT/5+1
        UUrif2=UUrif(i,:)*UUrif(i,:)';
        P=(UUrif(i,:)*Umodi(i,:,j)').^2;
        ui=Umodi(i,:,j)*Umodi(i,:,j)';
        MACuu(j-ifast0,i)=P/(UUrif2*ui);
    end
end
STD=std(MACuu);
miniSTD=STD(1:NFFT/5); % 40Hz: valore scelto per avere una mSTD "
    pulita"
mSTD=sum(miniSTD)/(NFFT/5);
for n=1:NFFT/5+1
    MACuuMEDIO(n)=sum(MACuu(:,n)')/(ifast-ifast0);
end
DER=abs(diff(MACuuMEDIO(:)));
miniDER=DER(1:NFFT/5); % 40Hz: valore scelto per avere una mDER "
    pulita"
mDER=sum(miniDER)/(NFFT/5);

% Grafici MAC medio e SV
x=0:NFFT;
y=x-x+str2double(get(handles.mac,'String'));

figure1=figure('Color',[1 1 1]); % (0.4*fs)*NFFT/fs = 0.4*NFFT
set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
```

```
movegui(figure1 , 'center');

subplot(2,1,1);
[AX,H1,H2]=plotyy(f1(1:NFFT/5),MACuuMEDIO(1:NFFT/5), f1(1:NFFT/5),
    SS1(1:NFFT/5,:,ifast),'plot'); hold on
set(get(AX(1),'Ylabel'),'String','MAC','Color','k'); %scritte y1
set(AX(1),'YLim',[0 1],'YTick',[0:0.1:1],'XTick',[0:1:0.4*fs]);
set(AX(1),'YColor','b');
set(get(AX(2),'Ylabel'),'String','SV(i)','Color','k'); % scritte
    y2
set(AX(2),'YLim',[-220 -20],'YTick',[-220:20:-20],'XTick'
     ,[0:1:0.4*fs]);
plot(x,y,'-r','linewidth',1);
set(get(gca,'Xlabel'),'String','Frequency (Hz)','Color','k');
title('\fontsize{12}Singular values - reference file');
subplot(2,1,2);
for i=ifast0:ifast
     [AX,H1,H2]=plotyy(f1(1:NFFT/5),MACuuMEDIO(1:NFFT/5),f1(1:NFFT
    /5),SS1(1:NFFT/5,1,i),'plot'); hold on
     set(AX(2),'YTick',[],'XTick',[]);
     set(AX(2),'YLim',[-220 -20]);
     set(AX(1),'YLim',[0 1]);
end
set(get(AX(1),'Ylabel'),'String','MAC','Color','k'); %scritte y1
set(AX(1),'YTick',[0:0.1:1],'XTick',[0:1:0.4*fs]);
set(AX(1),'YColor','b');
set(get(AX(2),'Ylabel'),'String','SV1','Color','k'); % scritte y2
set(AX(2),'YTick',[-220:20:-20]);
plot(x,y,'-r','linewidth',1);
set(get(gca,'Xlabel'),'String','Frequency (Hz)','Color','k');
title('\fontsize{12}First singular value - all files');


%
    ------------------------------------------------------------------------

function [handles]=load_Callback(hObject, eventdata, handles)
if getappdata(0,'scheduleron')~=1
     button = questdlg('Are you sure you want to start a new
    project from template? Any unsaved progress will be lost.','
```

```
     title','Yes','No',2);
     if strcmp(button,'No')==1
         return;
     end

end
if ~exist('../template','file')
    mkdir('../template');
end
if ~exist('../tmp','file')
    mkdir('../tmp');
end

if getappdata(0,'scheduleron')==1
    FileName=getappdata(0,'schfilename');
    PathName=getappdata(0,'schpathname');

else
    [FileName,PathName] = uigetfile('../template/*.efdd','Select
    the template file to use.');
end
setappdata(0,'usingtemplate',1);

newfn='current.zip';     tmp='../tmp/';
copyfile([PathName FileName],[tmp newfn],'f');
unzip([tmp newfn],tmp);
delete([tmp newfn]);
templatename=regexprep(FileName,'.efdd','.mat');
load([tmp templatename]);

if ~strcmp(templatetype,'AUTOEFDD')
    errordlg('The selected file is not an AutoEFDD template.');
    return;
end

setappdata(0,'savefile',[tmp 'savefile.mat']);
setappdata(0,'indici',[tmp 'indici.mat']);
setappdata(0,'templatename',templatename);

set(handles.newresultsradio,'Value',0);
set(handles.NFFT,'String',NFFT,'Value',1,'Enable','off');
set(handles.L,'String',L,'Enable','off');
set(handles.fs,'String',fs,'Enable','off');
set(handles.channels,'String',channels,'Enable','off');
set(handles.mac,'String',mac,'Enable','off');
set(handles.n,'String',n,'Enable','off');
set(handles.mac_damping,'String',mac_damping,'Enable','off');
set(handles.max_corr,'String',max_corr,'Enable','off');
set(handles.min_corr,'String',min_corr,'Enable','off');
set(handles.loadfdd,'Enable','off');
set(handles.ifast,'Enable','off');
set(handles.ifast0,'Enable','off');
setappdata(0,'author',author);
setappdata(0,'tempdate',tempdate);
setappdata(0,'description',description);
loadtemplate;


%
    ----------------------------------------------------------------------

function merge_Callback(hObject, eventdata, handles)
merge;



% --- Executes on button press in frequenzetempo.
function frequenzetempo_Callback(hObject, eventdata, handles)
Result=getappdata(0,'Result_frequency');
plotfreq(Result,0);

% --- Executes on button press in validation.
function validation_Callback(hObject, eventdata, handles)
validate_dampingGUI;

% --- Executes on button press in dampingvsfrequencies.
function dampingvsfrequencies_Callback(hObject, eventdata, handles
    )
load DAMPING_RESULTS;
```

```
f_vs_d (newf20, d_results0, f_mean, d_mean); %Plot delle frequenze
    vs i damping rilevati



% --- Executes on button press in damping.
function [handles]=damping_Callback(hObject, eventdata, handles)
load PERDAMPING
set(handles.statusD,'String','Wait..','BackgroundColor',[1 0 0]);
% Scelta dei parametri per la ricerca dei damping
sogliaMAC = str2num(get(handles.mac_damping,'String'));
max_corr = str2num(get(handles.max_corr,'String'));
min_corr = str2num(get(handles.min_corr,'String'));
Fdd_loc=getappdata(0,'FDD_location');

intervallo=0.4;
lastfm=size(newf2,2);
d_results=zeros(size(newf2,1),size(newf2,2));
d_std=zeros(size(newf2,1),size(newf2,2));
if exist('damping_temp','file')
    ok=0;
    while ok==0
        [status, message, messageid]=rmdir('damping_temp','s');
        ok=status;
        pause(2);
    end
end
mkdir('damping_temp');
size(SS0,3)
for ifast3=1:size(SS0,3);
    Segnale=SS0(1:round(intervallo*NFFT+1),1,ifast3);
    indiceF=ones(lastfm,1);
    damping=zeros(lastfm,1);

    j=1;
    i=1;

    while j<=lastfm && i < intervallo*NFFT+1
        i=i+1;
        if newf2(ifast3,j)==0
            j=j+1;
        else
            if (f1(i)>=newf2(ifast3,j) && f1(i-1)<=newf2(ifast3,j)
)
                indiceF (j) = i;
                j=j+1;
            end
        end
    end

    rangemax=100;
    rangeL=zeros(length(indiceF),1);
    rangeR=zeros(length(indiceF),1);
    k=1;

    for Fj=1:length(indiceF) %Fj indice del modo considerato
        if indiceF(Fj)==1||indiceF(Fj)==0
            rangeR(Fj)=0;
            rangeL(Fj)=0;
        else

            Urif=Umodi(indiceF(Fj),:,ifast3);
            i=1;
            MAC=1;
            while i<rangemax && MAC>=sogliaMAC && i < indiceF(Fj)
%i indice della frequenza
                Uint=Umodi(indiceF(Fj)-i,:,ifast3);
                MAC=(Urif*Uint').^2/((Uint*Uint')*(Urif*Urif'));
                if( MAC < sogliaMAC)
                    rangeL(Fj)=i-1;
                end
                i=i+1;
            end
            i=1;
            MAC=1;
            while i<rangemax && MAC>=sogliaMAC && i < indiceF(Fj)
%i indice della frequenza
                Uint=Umodi(indiceF(Fj)+i,:,ifast3);
                MAC=(Urif*Uint').^2/((Uint*Uint')*(Urif*Urif'));
                if( MAC < sogliaMAC)
                    rangeR(Fj)=i-1;
                end
                i=i+1;
            end
            %Devo mantenere la campana centrata sulla frequenza di
     risonanza.
            maxDrange=0.5; %permetto al massimo un 20% di
    differenza tra range a sn e destra
            if rangeR(Fj)> rangeL(Fj)*(1+maxDrange);
                rangeR(Fj)=rangeL(Fj)*(1+maxDrange);
            end
            if rangeL(Fj)> rangeR(Fj)*(1+maxDrange)
                rangeL(Fj)=rangeR(Fj)*(1+maxDrange);
            end

        end
    end
    FF=newf2(ifast3,:);
    % Costruisco la campana da trasformare nel tempo azzerando
    tutto quello che sta attorno al range
    Bell=zeros(length(Segnale),length(indiceF)); %ciascuna colonna
     corrisponde ad un modo

    for Fj=1:length(indiceF)
        for i=1:length(Segnale)
            if i >= indiceF(Fj)-rangeL(Fj) && i <= indiceF(Fj)+
    rangeR(Fj)
                Bell(i,Fj)=Segnale(i);
            end
        end
    end
    %passaggio nel dominio del tempo con FFT Inversa
    iT=(0:1/fs:200); %dominio temporale con passo 1/fs.
    for Fj=1:lastfm
        %Antitrasformata delle campane
        TBell = real(ifft(Bell(:,Fj),NFFT));
        TBell0(:,Fj) = TBell(1:length(TBell)/2)./TBell(1);
        MyTBell=TBell0(:,Fj);

        % Costruisco la curva dei valori massimi
        j=1;
        MaxTBell(j,Fj)=MyTBell(1);
        MaxT(j,Fj)=iT(1);
        j=j+1;
        for i=2:length(MyTBell)-1

            if MyTBell(i)>MyTBell(i-1)&&MyTBell(i)>MyTBell(i+1)
                MaxTBell(j,Fj)=MyTBell(i);
                MaxT(j,Fj)=iT(i);
                j=j+1;
            end
        end

        % Costruisco la curva dei valori minimi
        j=1;
        minTBell(j,Fj)=MyTBell(1);
        minT(j,Fj)=iT(1);
        j=j+1;
        for i=2:length(MyTBell)-1

            if MyTBell(i)<MyTBell(i-1)&&MyTBell(i)<MyTBell(i+1)
                minTBell(j,Fj)=MyTBell(i);
                minT(j,Fj)=iT(i);
                j=j+1;
            end
        end
    end
    N=0.10*NFFT/fs;
    RettaRif=0;
    for Fj=1:lastfm
        if indiceF(Fj)==1 || (rangeL(Fj)==0 && rangeR(Fj)==0) || (
rangeR(Fj)+rangeL(Fj)<N)
            FT(Fj)=NaN;
            damping(Fj)=NaN;
            damp_std(Fj)=NaN;
        else

            minTlag(Fj)=find (MaxTBell(:,Fj)< max_corr,1,'first');
            maxTlag(Fj)=find (MaxTBell(:,Fj)<min_corr,1,'first')
-1;
```

```
          minTlagmin(Fj)=find (abs(minTBell(:,Fj))< max_corr,1,'
first');
          maxTlagmin(Fj)=find (abs(minTBell(:,Fj))<min_corr,1,'
first')-1;
          if maxTlag(Fj)<minTlag(Fj) maxTlag(Fj)=minTlag(Fj);
end
          %stima delle frequenze
          minTindex=find(abs(TBell0(:,Fj))<max_corr,1,'first');
          maxTindex=find(abs(TBell0(:,Fj))>min_corr,1,'last');
          MyTBell=TBell0(minTindex:maxTindex,Fj);
       %considero il solo tratto 'pulito'
          Frequency_D = length( find(diff(MyTBell>0)~=0))+1;
  %il numero di zero crossing
          FT(Fj)=Frequency_D/(2*(iT(maxTindex)-iT(minTindex)));
  %il vettore contentente le frequenze di risonanza stimata nel
 dominio del tempo

          %stima del damping
          k=maxTlag(Fj)-minTlag(Fj)+maxTlagmin(Fj)-minTlag(Fj);
%il numero di picchi positivi
          delta(Fj)=2/k*log(max_corr/abs(min_corr));
          damping(Fj)=1/sqrt(1+(2*pi/delta(Fj))^2);

          Rnorm=zeros(maxTlag(Fj)-minTlag(Fj)+1,1);
          RettaRif(1,Fj)=log(MaxTBell(minTlag(Fj),Fj));
          RettaRif(maxTlag(Fj)-minTlag(Fj)+1,Fj)=log(MaxTBell(
maxTlag(Fj),Fj));
          Rnorm(1)=0; Rnorm(maxTlag(Fj)-minTlag(Fj)+1)=0;
          for rr=2:(maxTlag(Fj)-minTlag(Fj))
              RettaRif(rr,Fj)=RettaRif(1,Fj)-(RettaRif(1,Fj)-
RettaRif(length(Rnorm),Fj))/(MaxT(maxTlag(Fj),Fj)-MaxT(minTlag(
Fj),Fj))*(MaxT(minTlag(Fj)+rr-1,Fj)-MaxT(minTlag(Fj),Fj));
              Rnorm(rr)=log(MaxTBell(minTlag(Fj)+rr-1,Fj))-
RettaRif(rr,Fj);
          end
          %Se prende tutto (o quasi) il dominio temporale
probabilmente è
          %dovuto a componente armonica e ignoro il damping.
Questa parte
```

```
            %va eliminata nel momento in cui si implementa un
    controllo per
            %le componenti armoniche.
            if length(MyTBell)>0.7*length(TBell0(:,Fj))
                damping(Fj)=NaN;
            end

            dstd=(sum(Rnorm.^2)/length(Rnorm))^0.8;
            if dstd==0
                damp_std(Fj)=NaN;
            else
                damp_std(Fj)=1/dstd;
            end

        end
    end

    filename=['damping_temp/validazione' num2str(ifast3) '.mat'];
    save (filename,'Segnale','FF','FT','damping','indiceF','f1','
    Bell','damp_std','minTlag','maxTlag','MaxT','max_corr','
    min_corr','MaxTBell','iT','TBell0','newf2','RettaRif');
    clear RettaRif;

    d_results(ifast3,:)=damping;
    d_std(ifast3,:)=damp_std;

end

d_results0=zeros(size(d_results));
newf20=zeros(size(newf2));
for ii=1:lastfm
    iii=1;
    for ifast3=1:size(SS0,3); %pulizia dei vettori per usare box
    plot e trim mean
        if ~isnan(d_results(ifast3,ii))
            d_results0(iii,ii)=d_results(ifast3,ii);
            newf20(iii,ii)=newf2(ifast3,ii);
            iii=iii+1;
        end
    end
    indz=find(d_results0(:,ii)>0,1,'last');
    d_mean(ii)=trimmean(d_results0(1:indz,ii),20);
    f_mean(ii)=trimmean(newf20(1:indz,ii),20);
end
load(Fdd_loc, 'Result');
d_results2=[Result d_results];
save DAMPING_RESULTS d_results2 d_mean f_mean newf2 d_results0
    newf20
setappdata(0,'Result_damping',d_results2);
setappdata(0,'date_files',Result);
set(handles.validation,'Enable','on');
set(handles.dampingvsfrequencies,'Enable','on');
set(handles.meanvsfrequency,'Enable','on');
set(handles.boxplot,'Enable','on');
set(handles.saveres,'Enable','on');

strm{1}='Damping vs time';
for i=2:size(d_results2,2)
    strm{i}=['Mode ' num2str(i-1)];
    strl{i-1}=['Mode ' num2str(i-1,4) ': ' num2str(f_mean(i-1),4)
    ' Hz, Damping: ' num2str(d_mean(i-1)*100) '%'];
end
set(handles.modelist,'String',strl);
set(handles.dmode,'String',strm);
set(handles.dmode,'Enable','on');
set(handles.savetemplate,'Enable','on');
set(handles.statusD,'String','OK','BackgroundColor',[0 1 0]);
set(handles.phase2,'BackgroundColor',[0 1 0]);
set(handles.phase3,'BackgroundColor',[1 1 0]);
guidata(gcbo,handles);

% --- Executes on button press in meanvsfrequency.
function meanvsfrequency_Callback(hObject, eventdata, handles)
load DAMPING_RESULTS;
f_mean_vs_d_mean(f_mean,d_mean); %Plot del damping medio vs la
    frequenza media del modo



% --- Executes on button press in boxplot.
function boxplot_Callback(hObject, eventdata, handles)
```

```
load DAMPING_RESULTS;
f_mean_vs_d(d_results0 , f_mean, d_mean); %Plot del damping
    rispetto alla frequenza del modo

% --- Executes on button press in saveres.
function saveres_Callback(hObject, eventdata, handles)
Result_frequency=getappdata(0,'Result_frequency');
Result_damping=getappdata(0,'Result_damping');
if getappdata(0,'usingtemplate')==1 | getappdata(0,'savingtemplate
    ')==1
    foldername=['../Results/' getappdata(0,'templatename')];
    if ~exist(foldername,'file')
        mkdir(foldername);
    end
    elenco=getallfiles(foldername,1);
    ind=0;
    for i=1:length(elenco)
        elemento=elenco{i};
        ind(i)=str2double(elemento((findstr(elemento,'Result_')+7)
    :(findstr(elemento,'.mat')-1)));
    end
    indmax=max(ind);
    if indmax>=1
        indice=indmax+1;
    else
        indice=1;
    end
    templatename=getappdata(0,'templatename');
    setappdata(0,'resultsfolder',foldername);
    save([foldername '/Result_' num2str(indice) '.mat'],'
    Result_frequency','Result_damping','templatename');
else
    uisave({'Result_frequency', 'Result_damping'},'Result_');
end
set(handles.phase3,'BackgroundColor',[0 1 0]);


hh=helpdlg('Results successfully saved.');
pause(3);
if ishandle(hh) close(hh); end
```

```
function dmode_Callback(hObject, eventdata, handles)
load DAMPING_RESULTS;
ind=get(hObject,'Value');

if ind>size(d_results2,2) || ind <= 0
    errdlg('Indice non valido.');
else
    if ind>1
        figure1 = figure ('Color',[1 1 1]);
        % Create axes
        axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on'
    ,...
            'Position',[0.110705596107056 0.186544342507645
   0.834549878345499 0.685015290519878]);
        set(figure1, 'Units','centimeters', 'Position',[0 0 50
   20]/2);
        movegui(figure1, 'center');
        box(axes1,'on');
        hold(axes1,'all');

        plot(d_results2(:,1),d_results2(:,ind).*100,'linestyle','
   none','LineWidth',1.2,'Marker','.','Markersize',5); hold on;
   grid on;
        datetick('x','dd mmm yy','keeplimits','keepticks');
        xlabel('Time','FontSize',12); ylabel('[%]','FontSize',12);
        ylabel('Damping Ratio (%)','FontSize',12);
        title(['\fontsize{14}Damping Ratio Trend, Mode ' num2str(
   ind-1)]);
    end

end

% --- Executes on button press in viewres.
function viewres_Callback(hObject, eventdata, handles)
viewresults;

% --- Executes on button press in savetemplate.
function savetemplate_Callback(hObject, eventdata, handles)
NFFT=get(handles.NFFT,'String');
```

```
NFFT=NFFT{get(handles.NFFT,'Value')};

L=get(handles.L,'String');
fs=get(handles.fs,'String');
channels=get(handles.channels,'String');
mac=get(handles.mac,'String');
n=get(handles.n,'String');
mac_damping=get(handles.mac_damping,'String');
max_corr=get(handles.max_corr,'String');
min_corr=get(handles.min_corr,'String');
templatetype='AUTOEFDD';

st=savetemplate;
waitfor(st);
selectfilename=getappdata(0,'tempname');
description=getappdata(0,'tempdesc');
author=getappdata(0,'author');
image=getappdata(0,'tempimage');
tempdate=datestr(now, 'dd/mm/yy HH:MM');

templatename=strrep(selectfilename,' ','_');
if ~exist('..\template','file')
    mkdir('..\template');
end

selectfolder=['..\template\' templatename];
mkdir(selectfolder);
savefile=[selectfolder '\Template_' templatename '.mat'];

save(savefile,'templatetype','tempdate','NFFT','L','fs','channels'
    ,'mac','n','mac_damping','max_corr','min_corr','templatename','
    description','author');
copyfile('indici.mat',[selectfolder '\indici.mat'],'f');
copyfile('savefile.mat',[selectfolder '\savefile.mat'],'f');
copyfile(image,[selectfolder '\tempimage.jpg'],'f');
zip(['..\template\Template_' templatename '.zip'],'*',['..\
    template\' templatename]);
movefile(['..\template\Template_' templatename '.zip'],['..\
    template\Template_' templatename '.efdd'],'f');
```

```
ok=0;
while ok==0
    [ok, message, messageid]=rmdir(selectfolder,'s');
    if ok==0
        pause(2);
    end
end
setappdata(0,'templatename',templatename);
setappdata(0,'savingtemplate',1);

helpdlg(['The template file: Template_' templatename '.efdd has
   been successfully saved in the template folder.'])


%
   ------------------------------------------------------------------------

function view_Callback(hObject, eventdata, handles)
viewresults;

%
   ------------------------------------------------------------------------

function new_Callback(hObject, eventdata, handles)
button = questdlg('Are you sure you want to start a new project?
   Any unsaved progress will be lost.','title','Yes','No',2);
if strcmp(button,'Yes')==1
    curr=findobj('Tag','autoefdd');
    close(curr);
    AutoEFDDgui;
    setappdata(0,'usingtemplate',0);
    setappdata(0,'savingtemplate',0);
    setappdata(0,'scheduleron',0);
end


%
   ------------------------------------------------------------------------

function exit_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to exit (see GCBO)
% eventdata  reserved - to be defined in a future version of
    MATLAB
% handles    structure with handles and user data (see GUIDATA)
button = questdlg('Are you sure you want to quit? Any unsaved
    progress will be lost.','title','Yes','No',2);
if strcmp(button,'Yes')==1
     curr=findobj('Tag','autoefdd');
     close(curr);
end


% --- Executes on button press in savefddfile.
function savefddfile_Callback(hObject, eventdata, handles)
[FileName,PathName] = uiputfile('FDD_*.mat');
copyfile(getappdata(0,'FDD_location'),[PathName FileName],'f');




%
    --------------------------------------------------------------------

function aboutmenu_Callback(hObject, eventdata, handles)
aboutfig;


%
    --------------------------------------------------------------------

function newschedule_Callback(hObject, eventdata, handles)
newschedule;

%
    --------------------------------------------------------------------

function schedulelauncher_Callback(hObject, eventdata, handles)
[FileName,PathName] = uigetfile('..\schedule\*.schedule','Select
    the scheduler you want to launch.');
schname=[PathName FileName];
if ~exist('..\tmp\','file') mkdir ('..\tmp\'); end
```

```
newfile=['..\tmp\' strrep(FileName,'.schedule','.mat')];
copyfile(schname,newfile,'f');
continua=1;
while continua==1
    clearvars -except handles continua newfile;
    load(newfile);
    setappdata(0,'scheduleron',1);
    setappdata(0,'schfilename',schfilename);
    setappdata(0,'schpathname',schpathname);
    set(handles.folder,'String',fastdir);
    orari=datenum(orari,'HH:MM');
    orari=datenum(0,0,0,hour(orari),minute(orari),0);
    adesso=clock;
    curr_time=datenum(0,0,0,adesso(4),adesso(5),0);

    for i=1:length(orari)
        if orari(i)<=curr_time
            oraritmp(i)=etime(datevec(orari(i)+datenum
   (0,0,1,0,0,0)),datevec(curr_time));
        else
            oraritmp(i)=etime(datevec(orari(i)),datevec(curr_time)
   );
        end

    end
    oraritmp=sort(oraritmp);
    prossimo=datestr((datenum(0,0,0,0,0,oraritmp(1))+curr_time),'
   HH:MM');
    t = timer('TimerFcn', 'setappdata(0,''execute'',1);',...
        'StartDelay',oraritmp(1),'TasksToExecute',1);
    start(t)
    hhh = helpdlg(['Next scheduled analysis is set at ' prossimo '
   .' char(10) 'Do you want to interrupt the schedule?'],'Schedule
    Launcher');
    while strcmp(get(t,'Running'),'on')
        pause(10)
        if ~ishandle(hhh)
            %       helpdlg('Scheduler Launcher has been stopped
   .');
            stop(t);
```

```
                delete(t);
                setappdata(0,'execute',0);
                continua=0;
           end
      end
      if ishandle(hhh) close(hhh); end
      if getappdata(0,'execute')==1
           [handles]=load_Callback([],[],handles);
           [handles]=startfdd_Callback([],[],handles);
      end


      delete(t);
      rmappdata(0,'execute');
end

setappdata(0,'scheduleron',0);


% --- Executes on button press in ssiradio.
function ssiradio_Callback(hObject, eventdata, handles)
set(handles.ssiradio,'Value',1);
set(handles.radionuovo,'Value',0);
set(handles.radiofdd,'Value',0);

set(handles.existingdatapanel,'Visible','off');
set(handles.newdatapanel,'Visible','off');
set(handles.ssipanel,'Visible','on');
set(handles.ssibrowse,'Enable','on');
set(findall(handles.uipanel7,'-property','enable'),'enable','off')
     ;


% --- Executes on button press in ssibrowse.
function ssibrowse_Callback(hObject, eventdata, handles)
folder_name = uigetdir('','Select data folder');
if folder_name(1)~=0
     set(handles.ssidir,'String',folder_name);
     set(handles.ssistart,'Enable','on');
end
```

```
% --- Executes on button press in ssistart.
function ssistart_Callback(hObject, eventdata, handles)
folder_name=get(handles.ssidir,'String');

%SSI: look at [23]

    if size(appr,1)<12
        Result_frequency(ifast,1:13) =[kn appr(1:end,1)' zeros
   (1,12-size(appr,1))];
        Result_damping(ifast,1:13) = [kn (appr(1:end,2)')./100
   zeros(1,12-size(appr,1))];
    else
        Result_frequency(ifast,1:13) =[kn appr(1:12,1)'];
        Result_damping(ifast,1:13) =[kn (appr(1:12,2)')./100];
    end
    %     Result_damping(ifast,1:13) =[kn  (appr(:,2)')./100];
    save ssitemp Result_frequency Result_damping;





% --- Executes on button press in ssivalidate.
function ssivalidate_Callback(hObject, eventdata, handles)

[FileName,PathName,FilterIndex] = uigetfile('*.mat','Select the
   FDD results file');
filenameFDD=[PathName FileName];
load(filenameFDD);
for i=2:size(Result_frequency,2)
    j=1;
    fmean=0;
    for k=1:size(Result_frequency,1)
        if ~isnan(Result_frequency(k,i))
            fmean(j)=Result_frequency(k,i);
            j=j+1;
        end
    end
    freq(i-1)=trimmean(fmean,20);
    f_fddsucc(i-1)=length(fmean)/length(Result_frequency(:,i));
end
```

```
for i=2:size(Result_damping,2)
    j=1;
    dmean=0;
    for k=1:size(Result_damping,1)
        if ~isnan(Result_damping(k,i))
            dmean(j)=Result_damping(k,i);
            j=j+1;
        end
    end
    damp(i-1)=trimmean(dmean,20);
    d_fddsucc(i-1)=length(dmean)/length(Result_damping(:,i));
end

if length(freq)>6
    freq=freq(1:6);
end
if length(damp)>6
    damp=damp(1:6);
end
clear Result_frequency Result_damping;
[FileName,PathName,FilterIndex] = uigetfile('*.mat','Select the
    SSI results file');
filenameSSI=[PathName FileName];
load(filenameSSI);
% freq=[4.954 6.090 6.809 7.560 11.7 12.051]; %Ponte Nuovo
% freq=[2.128 4.067 4.547 6.146 9.72 11.56];
varia=0.7;
fnew=zeros(size(Result_frequency));
dnew=zeros(size(Result_frequency));
fnew(:,1)=Result_frequency(:,1);
dnew(:,1)=Result_frequency(:,1);
for i=2:size(Result_frequency,2)
    for j=1:size(Result_frequency,1)
        for k=1:length(freq)
            if k==1
                varia0=varia;
                if varia < (freq(k+1)-freq(k))/2
                    varia1=varia;
                else
                    varia1=(freq(k+1)-freq(k))/2;
                end
            elseif k==length(freq)
                varia1=varia;
                if varia < (freq(k)-freq(k-1))/2
                    varia0=varia;
                else
                    varia0=(freq(k)-freq(k-1))/2;
                end
            else
                if varia < (freq(k)-freq(k-1))/2
                    varia0=varia;
                else
                    varia0=(freq(k)-freq(k-1))/2;
                end
                if varia < (freq(k+1)-freq(k))/2
                    varia1=varia;
                else
                    varia1=(freq(k+1)-freq(k))/2;
                end
            end
            if Result_frequency(j,i)>freq(k)-varia0 &&
    Result_frequency(j,i)<freq(k)+varia1
                fnew(j,k+1)=Result_frequency(j,i);
                dnew(j,k+1)=Result_damping(j,i);
                break;
            end
        end
    end
end
clear Result_damping Result_frequency;
[~,col0]=find(dnew,1,'last');
[~,col]=find(fnew,1,'last');
Result_damping=dnew(:,1:col0);
Result_frequency=fnew(:,1:col);
Result_damping(Result_damping==0)=NaN;
Result_frequency(Result_frequency==0)=NaN;
uisave({'Result_frequency', 'Result_damping'},'CleanResult_');

%Validation results
%freq
```

```
for i=2:size(Result_frequency,2)
    col=Result_frequency(:,i);
    col=col(~isnan(col));
    f_iqr=iqr(col);
    l_inf=prctile(col,25)-1.5*f_iqr;
    l_sup=prctile(col,75)+1.5*f_iqr;
    col(col<l_inf | col>l_sup)=[];
    fmax(i-1)=max(col);
    fmin(i-1)=min(col);
    fmean(i-1)=mean(col);
    fstd(i-1)=std(col);
    fsuccess(i-1)=length(col)/length(Result_frequency(:,i));
    ferror(i-1)=(abs(fmean(i-1)-freq(i-1))/mean([freq(i-1) fmean(i
   -1)]));
end
fmean=fmean(1:length(fmax));
%damp
for i=2:size(Result_damping,2)
    col=Result_damping(:,i);
    col=col(~isnan(col));
    d_iqr=iqr(col);
    l_inf=prctile(col,25)-1.5*d_iqr;
    l_sup=prctile(col,75)+1.5*d_iqr;
    col(col<l_inf | col>l_sup)=[];
    dmax(i-1)=max(col);
    dmin(i-1)=min(col);
    dmean(i-1)=mean(col');
    dstd(i-1)=std(col);
    dsuccess(i-1)=length(col)/length(Result_damping(:,i));
    derror(i-1)=(abs(dmean(i-1)-damp(i-1))/mean([damp(i-1) dmean(i
   -1)]));
end
dmean=dmean(1:length(dmax));
fres=[[0 1:length(fmax)] ;[0 fmax];[0 fmin];[0 fmean];[0 fstd];[0
    fsuccess]];
dres=[[0 1:length(dmax)] ; [0 dmax]; [0 dmin]; [0 dmean]; [0 dstd
    ]; [0 dsuccess]];
f_label={'Mode' 'Fmax' 'Fmin' 'Fmean' 'std' 'success rate'};
d_label={'Mode' 'Dmax' 'Dmin' 'Dmean' 'std' 'success rate'};
[file path]= uiputfile('Validation.xls','Save validation file');
```

```
ofile = [path file];
xlswrite(ofile,fres,1);
xlswrite(ofile,f_label',1);
xlswrite(ofile,dres,2);
xlswrite(ofile,d_label',2);



f=figure('Name','AutoEFDD vs AutoSSI validation: Natural
    Frequencies',...
     'Color',[1 1 1]);
if length(fmean)<length(freq)
     fmean=[fmean zeros(1,length(freq)-length(fmean))];
     fsuccess=[fsuccess zeros(1,length(freq)-length(fsuccess))];
     ferror=[ferror zeros(1,length(freq)-length(ferror))];
end
dat =[1:length(freq);freq;f_fddsucc.*100;fmean;fsuccess.*100;
    ferror.*100];
rowname= {'Mode','Fmean(EFDD)','Success Rate(EFDD)','Fmean(SSI)','
    Success rate(SSI)','Relative error'};
t = uitable('Units','normalized','Position',...
     [0.057 0.312 0.894 0.445], 'Data', dat,...
     'RowName', rowname,...
     'ColumnName',[]);
fd = figure('Name','AutoEFDD vs AutoSSI validation: Damping Ratios
    ',...
     'Color',[1 1 1]);
if length(dmean)<length(damp)
     dmean=[dmean zeros(1,length(damp)-length(dmean))];
     dsuccess=[dsuccess zeros(1,length(damp)-length(dsuccess))];
     derror=[derror zeros(1,length(damp)-length(derror))];
end
datd =[1:length(damp);damp.*100;d_fddsucc.*100;dmean.*100;dsuccess
    .*100;derror.*100];
rowname= {'Mode','Dmean(EFDD)','Success Rate(EFDD)','Dmean(SSI)','
    Success rate(SSI)','Relative error'};
td = uitable('Units','normalized','Position',...
     [0.057 0.312 0.894 0.445], 'Data', datd,...
     'RowName', rowname,...
     'ColumnName',[]);
```

code/AutoEFDDgui.m

A.2 viewresults.m

```
function varargout = viewresults(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',        mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @viewresults_OpeningFcn, ...
                   'gui_OutputFcn',  @viewresults_OutputFcn, ...
                   'gui_LayoutFcn',  []  , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before viewresults is made visible.
function viewresults_OpeningFcn(hObject, eventdata, handles,
   varargin)
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

if isstr(getappdata(0,'CurrentResults'))
    set(handles.load,'Visible','off');
    load_Callback(hObject, eventdata, handles);
```

```
end
% UIWAIT makes viewresults wait for user response (see UIRESUME)
% uiwait(handles.viewpanel);


% --- Outputs from this function are returned to the command line.
function varargout = viewresults_OutputFcn(hObject, eventdata,
    handles)
varargout{1} = handles.output;


% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
curr=findobj('Tag','viewpanel');
setappdata(0,'CurrentResults',[]);
close(curr);

% --- Executes on button press in load.
function load_Callback(hObject, eventdata, handles)
set(handles.trend,'Enable','on');
set(handles.excelf,'Enable','on');
set(handles.exceld,'Enable','on');
set(handles.dvsf,'Enable','on');
set(handles.mdvsf,'Enable','on');
set(handles.boxplot,'Enable','on');
set(handles.dvst,'Enable','on');

if isstr(getappdata(0,'CurrentResults'))
     setappdata(0,'ResultsFile',getappdata(0,'CurrentResults'));
else
     [FileName,PathName,FilterIndex] = uigetfile('../Results/','
    Result_ or Sequence_');
     setappdata(0,'ResultsFile',[PathName FileName]);
end
set(handles.status,'String',['Now viewing: ' getappdata(0,'
    ResultsFile')]);

load(getappdata(0,'ResultsFile'));
for ii=2:size(Result_damping,2)
     iii=1;
    for ifast3=1:size(Result_damping,1); %pulizia dei vettori per
    usare box plot e trim mean
        if ~isnan(Result_damping(ifast3,ii))
            d_results0(iii,ii)=Result_damping(ifast3,ii);
            newf20(iii,ii)=Result_frequency(ifast3,ii);
            iii=iii+1;
        end
    end
    indz=find(d_results0(:,ii)>0,1,'last');
    d_mean(ii)=trimmean(d_results0(1:indz,ii),20);
    f_mean(ii)=trimmean(newf20(1:indz,ii),20);
end
setappdata(0,'d_results0',d_results0);
setappdata(0,'d_mean',d_mean);
setappdata(0,'f_mean',f_mean);
setappdata(0,'newf20',newf20);

strm{1}='Damping ratio trend';
for i=2:size(Result_damping,2)
    strm{i}=['Mode ' num2str(i-1)];
end
set(handles.dvst,'String',strm);



% --- Executes on button press in exceld.
function exceld_Callback(hObject, eventdata, handles)
load(getappdata(0,'ResultsFile'));
[FileName,PathName] =uiputfile('*.xls','Save as excel');
xlswrite([PathName FileName],Result_damping);

% --- Executes on button press in dvsf.
function dvsf_Callback(hObject, eventdata, handles)
f_vs_d (getappdata(0,'newf20'), getappdata(0,'d_results0'),
    getappdata(0,'f_mean'), getappdata(0,'d_mean')); %Plot delle
    frequenze vs i damping rilevati


% --- Executes on button press in mdvsf.
function mdvsf_Callback(hObject, eventdata, handles)
```

```
f_mean_vs_d_mean(getappdata(0,'f_mean'), getappdata(0,'d_mean'));
    %Plot del damping medio vs la frequenza media del modo



% --- Executes on button press in boxplot.
function boxplot_Callback(hObject, eventdata, handles)
f_mean_vs_d(getappdata(0,'d_results0'), getappdata(0,'f_mean'),
    getappdata(0,'d_mean')); %Plot del damping rispetto alla
    frequenza del modo



% --- Executes on selection change in dvst.
function dvst_Callback(hObject, eventdata, handles)
ind=get(hObject,'Value');
load(getappdata(0,'ResultsFile'));



     if ind>size(Result_damping,2) || ind <= 0
          errdlg('Index out of bounds.');
     else
               if ind>1
              figure1 = figure ('Color',[1 1 1]);
               % Create axes
               axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on
    ',...
               'Position',[0.110705596107056 0.186544342507645
    0.834549878345499 0.685015290519878]);
               set(figure1, 'Units','centimeters', 'Position',[0 0 50
     20]/2);
               movegui(figure1, 'center');
               box(axes1,'on');
               hold(axes1,'all');
              plot(Result_damping(:,1),Result_damping(:,ind).*100,'
    linestyle','none','LineWidth',1.2,'Marker','.','Markersize',5);
     hold on; grid on;
              datetick('x','dd mmm yy','keeplimits','keepticks');
              xlabel('Tempo','FontSize',12); ylabel('[%]','FontSize'
    ,12);
              ylabel('Damping ratio (%)','FontSize',12);
```

```
            title(['\fontsize{14}Damping ratio trend, mode '
    num2str(ind-1)]);
            end

     end

% --- Executes during object creation, after setting all
    properties.
function dvst_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in trend.
function trend_Callback(hObject, eventdata, handles)
load(getappdata(0,'ResultsFile'));

plotfreq(Result_frequency,0);

% --- Executes on button press in excelf.
function excelf_Callback(hObject, eventdata, handles)
load(getappdata(0,'ResultsFile'));
[FileName,PathName] =uiputfile('*.xls','Save as excel');
xlswrite([PathName FileName],Result_frequency);



% --- Executes when user attempts to close viewpanel.
function viewpanel_CloseRequestFcn(hObject, eventdata, handles)
setappdata(0,'CurrentResults',[]);
delete(hObject);
```

code/viewresults.m

A.3 aboutfig.m

```
function varargout = aboutfig(varargin)
% Begin initialization code - DO NOT EDIT
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @aboutfig_OpeningFcn, ...
                   'gui_OutputFcn',  @aboutfig_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before aboutfig is made visible.
function aboutfig_OpeningFcn(hObject, eventdata, handles, varargin
   )
handles.output = hObject;
[X, map] = imread('unipd.jpeg');
ax1=findobj('Tag','unipd');
axes(ax1);
image(X);
colormap(map);
axis off          % Remove axis ticks and numbers
axis image
set(ax1,'Tag','unipd');
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = aboutfig_OutputFcn(hObject, eventdata,
   handles)
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
```

```
close(findobj('Tag','aboutfigure'));
```

code/aboutfig.m

A.4 validatedampingGUI.m

```
function varargout = validate_dampingGUI(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
    @validate_dampingGUI_OpeningFcn, ...
                   'gui_OutputFcn',
    @validate_dampingGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before validate_dampingGUI is made visible.
function validate_dampingGUI_OpeningFcn(hObject, eventdata,
    handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
Result=getappdata(0,'date_files');
set(handles.date,'String',datestr(Result));

% --- Outputs from this function are returned to the command line.
```

```
function varargout = validate_dampingGUI_OutputFcn(hObject,
    eventdata, handles)
varargout{1} = handles.output;


% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
validate=findobj('Tag','validate');
close(validate);

% --- Executes on selection change in date.
function date_Callback(hObject, eventdata, handles)
contents = cellstr(get(hObject,'String'));
selected=contents{get(hObject,'Value')};
isselected=strfind(contents,selected);
i=1;
while i<=length(isselected)
     if cell2mat(isselected(i))==1
         ind=i;
         i=length(isselected)+1;
     else
         i=i+1;
     end
end

handles.fileval=['damping_temp/validazione' num2str(ind) '.mat'];
guidata(hObject, handles);
load (handles.fileval, 'damping', 'newf2');
for i=1:length(damping)
    STR{i}= [ num2str(i) ': Frequency= ' num2str(newf2(ind,i)) '
   Hz, Damping= ' num2str(damping(i).*100) ' %'];
end
set(handles.mode, 'String', STR);
set (handles.mode, 'Enable', 'on');
set (handles.spectral, 'Enable', 'on');
set (handles.normalized, 'Enable', 'on');
set (handles.logaritmic, 'Enable', 'on');
set (handles.ff, 'Enable', 'on');
```

```
% --- Executes on button press in spectral.
function spectral_Callback(hObject, eventdata, handles)
load (handles.fileval);
contents = cellstr(get(handles.mode,'String'));
selected=contents{get(handles.mode,'Value')};
index=str2num(selected(1:(strfind(selected,':')-1)));
Fj=index;
figure1=figure('Color',[1 1 1]);
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on',...
     'Position',[0.110705596107056 0.186544342507645
    0.834549878345499 0.685015290519878]);
set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
movegui(figure1, 'center');
plot1=plot(f1(1:length(Segnale)),20*log10(Segnale),f1(1:length(
    Segnale)),20*log10(Bell(:,Fj)));
set(plot1(2),'Color',[1 0 0]);
xlabel('Frequency [Hz]','FontSize',12);
ylabel('SV1 [dB]','FontSize',12);
title('\fontsize{14}Identification of auto spectrum');



% --- Executes on button press in logaritmic.
function logaritmic_Callback(hObject, eventdata, handles)
load (handles.fileval);
contents = cellstr(get(handles.mode,'String'));
selected=contents{get(handles.mode,'Value')};
index=str2num(selected(1:(strfind(selected,':')-1)));
Fj=index;
figure1 = figure ('Color',[1 1 1]);
set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
movegui(figure1, 'center');
% Create axes
     axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on',...
           'Position',[0.110705596107056 0.186544342507645
    0.834549878345499 0.685015290519878]);
box(axes1,'on');
hold(axes1,'all');
title('\fontsize{14}Validation of Damping Ratio Estimate');
```

```
scatter(MaxT(minTlag(Fj):maxTlag(Fj),Fj),log(MaxTBell(minTlag(Fj):
    maxTlag(Fj),Fj)),'MarkerEdgeColor',[0 0.498039215803146 0],'
    Marker','+'); hold on
plot(MaxT(minTlag(Fj):maxTlag(Fj),Fj),RettaRif(1:(maxTlag(Fj)−
    minTlag(Fj)+1),Fj));
ylabel('\fontsize{12}Log of Absolute Extremu Value');
xlabel('\fontsize{12}Time Lag (sec)');


% −−− Executes on button press in ff.
function ff_Callback(hObject, eventdata, handles)
load (handles.fileval);

contents = cellstr(get(handles.mode,'String'));
selected=contents{get(handles.mode,'Value')};
index=str2num(selected(1:(strfind(selected,':')−1)));
Fj=index;
figure1 = figure ('Color',[1 1 1]);
% Create axes
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on',...
'Position',[0.110705596107056 0.186544342507645 0.834549878345499
    0.685015290519878]);
set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
movegui(figure1, 'center');
box(axes1,'on');
hold(axes1,'all');
title({'\fontsize{14}Frequency (F) vs Frequency (T)'});
scatter(FF,FT);
ylabel({'\fontsize{12}Natural frequency estimated in time domain (
    Hz)'});
xlabel({'\fontsize{12}Natural frequency estimated in frequency
    domain (Hz)'});

% −−− Executes on button press in normalized.
function normalized_Callback(hObject, eventdata, handles)
load (handles.fileval);

contents = cellstr(get(handles.mode,'String'));
selected=contents{get(handles.mode,'Value')};
index=str2num(selected(1:(strfind(selected,':')−1)));
```

```
Fj=index;
Rectx=[MaxT(minTlag(Fj),Fj) MaxT(maxTlag(Fj),Fj) MaxT(maxTlag(Fj),
    Fj) MaxT(minTlag(Fj),Fj) MaxT(minTlag(Fj),Fj)];
Recty=[max_corr max_corr -max_corr -max_corr max_corr];

figure1 = figure ('Color',[1 1 1]);
set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
movegui(figure1, 'center');
    % Create axes
     axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on',...
           'Position',[0.110705596107056 0.186544342507645
    0.834549878345499 0.685015290519878]);
     box(axes1,'on');
     hold(axes1,'all');
     title('\fontsize{14}Normalized Correlation Function of
    Singular Value Spectral Bell');
     plot(Rectx,Recty,MaxT(minTlag(Fj):maxTlag(Fj),Fj),MaxTBell(
    minTlag(Fj):maxTlag(Fj),Fj),iT(1:length(TBell0)),TBell0(1:
    length(TBell0),Fj),'LineWidth',1,'Color',[1 0 0],...
          'DisplayName','funzione di autocorrelazione normalizzata')
    ;
     xlabel({'\fontsize{12}Time Lag [s]'});
     ylabel({'\fontsize{12}Normalized Correlation'});
```

code/validatedampingGUI.m

A.5 fmeanvsd.m

```
function f_mean_vs_d (d_results, f_mean, d_mean)

figure1 = figure ('Color',[1 1 1]);
% Create axes
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on',...
     'Position',[0.110705596107056 0.186544342507645
    0.834549878345499 0.685015290519878]);
set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
movegui(figure1, 'center');
box(axes1,'on');
hold(axes1,'all');
```

```
title({'\fontsize{14}Box plot: Damping Ratio vs. Mean Natural
    Frequency '});
d_results(d_results==0) = nan;
boxplot (d_results*100,f_mean,'plotstyle','compact','whisker',1);
    hold on;
set(axes1,'ylim',[0,10]);
ylabel({'\fontsize{12}Damping (%)'});
xlabel({'\fontsize{12}Frequency (Hz)'});
```

code/fmeanvsd.m

A.6 fmeanvsdmean.m

```
function f_mean_vs_d_mean(f_mean,d_mean)

figure1 = figure ('Color',[1 1 1]);
% Create axes
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on',...
     'Position',[0.110705596107056 0.186544342507645
    0.834549878345499 0.685015290519878]);
set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
movegui(figure1, 'center');
box(axes1,'on');
hold(axes1,'all');
title({'\fontsize{14}Mean Damping Ratio vs. Mean Natural Frequency
    '});
scatter(f_mean,d_mean.*100,'MarkerFaceColor',[0 0 0],'
    MarkerEdgeColor',[0 0 0],...
     'Marker','+',...
     'LineWidth',2);
ylabel({'\fontsize{12}Damping (%)'});
xlabel({'\fontsize{12}Frequency (Hz)'});
```

code/fmeanvsdmean.m

A.7 fvsd.m

```
function f_vs_d (newf2, d_results, f_mean, d_mean)

figure1 = figure ('Color',[1 1 1]);
% Create axes
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on',...
     'Position',[0.110705596107056 0.186544342507645
    0.834549878345499 0.685015290519878]);
set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
movegui(figure1, 'center');
box(axes1,'on');
hold(axes1,'all');
title({'\fontsize{14}Damping vs. Natural Frequency'});
for i=1:size(d_results,2)
    scatter (newf2(:,i),d_results(:,i)*100); hold on;
end
scatter(f_mean,d_mean.*100,'MarkerFaceColor',[0 0 0],'
    MarkerEdgeColor',[0 0 0],...
     'Marker','+',...
     'LineWidth',2);
hold off;

ylabel({'\fontsize{12}Damping (%)'});
xlabel({'\fontsize{12}Frequency (Hz)'});
```

code/fvsd.m

A.8 loadtemplate.m

```
function varargout = loadtemplate(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
     'gui_Singleton',  gui_Singleton, ...
     'gui_OpeningFcn', @loadtemplate_OpeningFcn, ...
     'gui_OutputFcn',  @loadtemplate_OutputFcn, ...
     'gui_LayoutFcn',  [] , ...
     'gui_Callback',   []);
```

```
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end



% --- Executes just before loadtemplate is made visible.
function loadtemplate_OpeningFcn(hObject, eventdata, handles,
    varargin)

handles.output = hObject;
guidata(hObject, handles);
set(handles.author,'String',['Author: ' getappdata(0,'author')]);
set(handles.date,'String',['Date: ' getappdata(0,'tempdate')]);
set(handles.description,'String',['Description: ' getappdata(0,'
    description')]);
[X, map] = imread('../tmp/tempimage.jpg');
ax1=findobj('Tag','image');
axes(ax1);
image(X);
colormap(map);
axis off          % Remove axis ticks and numbers
axis image
set(ax1,'Tag','image');
pause(10);
close(findobj('Tag','loadtemp'));



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
close(findobj('Tag','loadtemp'));
```

code/loadtemplate.m

A.9 merge.m

```
function varargout = merge(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @merge_OpeningFcn, ...
    'gui_OutputFcn',  @merge_OutputFcn, ...
    'gui_LayoutFcn',  []  , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before merge is made visible.
function merge_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for merge
handles.output = hObject;
guidata(hObject, handles);
% Update handles structure

setappdata(0,'path','');
if getappdata(0,'mergetemplate')==1
    set(handles.sortdate,'Value',1);
    set(handles.sortuser,'Value',0);
    set(handles.files,'String',getallfiles(getappdata(0,'
    resultsfolder'),1));
    [handles]=merge_Callback(hObject, eventdata, handles);
end
```

```
% --- Outputs from this function are returned to the command line.
function varargout = merge_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

[FileName,PathName,FilterIndex] = uigetfile('*.mat','Result_? o
    Sequence_?',getappdata(0,'path'));
setappdata(0,'path',PathName);
res_name=[PathName FileName];
if FileName~=0
     Files=get(handles.files ,'String');
     Files{length(Files)+1}=res_name;
     set(handles.files ,'Value',length(Files));
     set(handles.files ,'String',Files);
     if(length(Files)>0)
         set (handles.remove,'Enable','on');
         set (handles.moveup,'Enable','on');
         set (handles.movedown,'Enable','on');
         set (handles.merge,'Enable','on');
     end
end


% --- Executes during object creation, after setting all
    properties.
function files_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in remove.
function remove_Callback(hObject, eventdata, handles)
```

```
content=get(handles.files,'String');
ind=get(handles.files,'Value');
content(ind)=[];
set(handles.files,'Value',length(content));
set(handles.files,'String',content);
if(length(content)==0)
    set (handles.remove,'Enable','off');
    set (handles.moveup,'Enable','off');
    set (handles.movedown,'Enable','off');
    set (handles.merge,'Enable','off');
end

% --- Executes on button press in moveup.
function moveup_Callback(hObject, eventdata, handles)
content=get(handles.files,'String');
ind=get(handles.files,'Value');

if ind>1
    tmp=content{ind-1};
    content{ind-1}=content{ind};
    content{ind}=tmp;
    set (handles.files,'String',content);
    set (handles.files,'Value',ind-1);
end

% --- Executes on button press in movedown.
function movedown_Callback(hObject, eventdata, handles)
content=get(handles.files,'String');
ind=get(handles.files,'Value');

if ind<length(content)
    tmp=content{ind+1};
    content{ind+1}=content{ind};
    content{ind}=tmp;
    set (handles.files,'String',content);
    set (handles.files,'Value',ind+1);
end

% --- Executes on button press in merge.
function [handles]=merge_Callback(hObject, eventdata, handles)
```

```
files=get (handles.files ,'String');
len=0;
dupl=0;
index=1;
deleteduplicate=get(handles.duplicate ,'Value');

for i=1:length(files)

    %se i files non contengono le variabili Result_frequency e
    Result_damping non sono files di risultati
     vars=who('-file', files{i});
     isok=cell2mat(strfind(vars,'Result_frequency'))+cell2mat(
    strfind(vars,'Result_damping'));
     if isempty(isok) || isok~=2
         errordlg(['The file ' files{i} ' is not a result file.']);
         return
     end
     load (files{i});
    %se i files presentano un numero diverso di modi non possono
    essere uniti.
     if i>1
         if size(Result_frequency ,2)~=modi
             errordlg('The selected result files are not compatible
    . Different number of modes.');
             return
         end
     end
     modi=size(Result_frequency ,2);

    %rimozione duplicati
     if i>1 && deleteduplicate==1
         sizeRes=size(Result_frequency ,1);
         for k=1:sizeRes
             if find(Sequence_f(:,1)==Result_frequency(sizeRes-k
    +1,1),1,'First')>0
                 Result_frequency(sizeRes-k+1,:)=[];
                 Result_damping(sizeRes-k+1,:)=[];
                 dupl=dupl+1;
             end
         end
     end
     %merge
     len=size(Result_frequency,1)+len;
     Sequence_f(index:len,:)=Result_frequency(1:size(
    Result_frequency,1),:);
     Sequence_d(index:len,:)=Result_damping(1:size(Result_frequency
    ,1),:);
     index=index+size(Result_frequency,1);
     clear Result_frequency Result_damping;
end
%ordinamento per data
if get(handles.sortdate,'Value')==1
     Sequence_f=sortrows(Sequence_f,1);
     Sequence_d=sortrows(Sequence_d,1);
end

Result_frequency=Sequence_f;
Result_damping=Sequence_d;

dialogstr=[num2str(length(files)) ' results files merged. '
    num2str(size(Result_frequency,1)) ' total rows. '];
if deleteduplicate==1
     dialogstr=[dialogstr num2str(dupl) ' duplicate entries deleted
    .'];
end
hd=helpdlg(dialogstr,'Merge details');

if getappdata(0,'mergetemplate')==1
     save([getappdata(0,'resultsfolder') '/current_sequence.mat'],'
    Result_frequency','Result_damping');
     setappdata(0,'CurrentRes',[getappdata(0,'resultsfolder') '/
    current_sequence.mat']);

else
     [FileName,PathName] =uiputfile([getappdata(0,'path') '
    Sequence_'],'Save merged file');
     save([PathName FileName],'Result_frequency','Result_damping');
     setappdata(0,'CurrentRes',[PathName FileName]);
end
set(handles.view,'Enable','on');
```

```
setappdata(0,'mergetemplate',0);
pause(3);
if ishandle(hd) close(hd); end



% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)
curr=findobj('Tag','mergepanel');
delete(curr);

% --- Executes on button press in sortdate.
function sortdate_Callback(hObject, eventdata, handles)
set (handles.sortuser,'Value',0);



% --- Executes on button press in view.
function view_Callback(hObject, eventdata, handles)
setappdata(0,'CurrentResults',getappdata(0,'CurrentRes'));
viewresults;
```

code/merge.m

A.10 newschedule.m

```
function varargout = newschedule(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
     'gui_Singleton', gui_Singleton, ...
     'gui_OpeningFcn', @newschedule_OpeningFcn, ...
     'gui_OutputFcn', @newschedule_OutputFcn, ...
     'gui_LayoutFcn', [] , ...
     'gui_Callback',  []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before newschedule is made visible.
function newschedule_OpeningFcn(hObject, eventdata, handles,
    varargin)
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);




% --- Outputs from this function are returned to the command line.
function varargout = newschedule_OutputFcn(hObject, eventdata,
    handles)
varargout{1} = handles.output;


% --- Executes on button press in browse.
function browse_Callback(hObject, eventdata, handles)
[FileName,PathName] = uigetfile('..\template\*.efdd','Select the
    template');
setappdata(0,'schfilename',FileName);
setappdata(0,'schpathname',PathName);


% --- Executes during object creation, after setting all
    properties.
function mm_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in addalarm.
```

```
function addalarm_Callback(hObject, eventdata, handles)
when=datenum(0,0,0,str2double(get(handles.hh,'String')),str2double
    (get(handles.mm,'String')),0);
oranuova=datestr(when,'HH:MM');

if oranuova~=0
     ore=get(handles.alarms,'String');
     ore{length(ore)+1}=oranuova;
     set(handles.alarms,'Value',length(ore));
     set(handles.alarms,'String',ore);
     if(length(ore)>0)
         set (handles.salva,'Enable','on');
         set (handles.delete,'Enable','on');
     end
end


% --- Executes during object creation, after setting all
    properties.
function alarms_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in salva.
function salva_Callback(hObject, eventdata, handles)
schfilename=getappdata(0,'schfilename');
schpathname=getappdata(0,'schpathname');
fastdir=getappdata(0,'fastdir');
orari=get(handles.alarms,'String');
nome=get(handles.name,'String');
if isempty(nome)
     errordlg('Please set a name for the schedule.');
     return;
end
if isempty(schfilename)
     errordlg('Please select a template file.');
     return;
```

```
end
if isempty(fastdir)
    errordlg('Please select the default folder in which
   acceleration data are stored.');
    return;
end
if ~exist('..\schedule','file')
    mkdir('..\schedule');
end
filename=['..\schedule\sch_' nome '.mat'];
save(filename,'schfilename','schpathname','orari','nome','fastdir'
   );
movefile(filename,['..\schedule\sch_' nome '.schedule'],'f');
ww=helpdlg('Schedule successfully create. Launch it with Schedule
   Launcher.');
pause(5);
if ishandle(ww) close(ww); end

% --- Executes on button press in delete.
function delete_Callback(hObject, eventdata, handles)
content=get(handles.alarms,'String');
ind=get(handles.alarms,'Value');

content(ind)=[];

set(handles.alarms,'Value',length(content));
set(handles.alarms,'String',content);
if(length(content)==0)
    set (handles.salva,'Enable','off');
    set (handles.delete,'Enable','off');
end



% --- Executes on button press in browsef.
function browsef_Callback(hObject, eventdata, handles)
folder_name = uigetdir('../','Select the folder in which new data
   will be stored.');
setappdata(0,'fastdir',folder_name);
```

code/newschedule.m

A.11 plotfreq.m

```
function plotfreq(Result,fraxes)
Result0 = Result;
Result0(find(isnan(Result0))) = 0;
for j=1:size(Result,2)
    media(j) = sum(Result0(:,j))/sum(sign(Result0(:,j)));
end

if fraxes~=0
    axes(fraxes);

else
    figure1=figure('Color',[1 1 1]);
    set(figure1, 'Units','centimeters', 'Position',[0 0 50 20]/2);
    movegui(figure1, 'center');
end



maxf=max(max(Result(:,2:end)));
for i = 2:size(Result,2)
    %         a = num2str(media(i));
    fr=num2str(trimmean(Result(:,i),20),4);
    set_legend(i-1) = cellstr(['Mode ' num2str(i-1) ': Hz ' fr]);
end

plot(Result(:,1),Result(:,2:end),'linestyle','none','LineWidth'
    ,1.2,'Marker','.','Markersize',5); hold on; grid on;
set(gca,'FontSize',10,'YTick',[0:2:maxf+2],'YLim',[0 maxf+2]);

if fraxes==0
    set(gca,'Position',[0.13 0.105714285714286 0.633513513513514
    0.811428571428571]);
    legend(set_legend,'Position',[0.809909909909913
    0.0714285714285714 0.167567567567568 0.879706566220238]);
else
    set(gca,'Tag','frequenze');
end

datetick('x','mmm yy','keeplimits','keepticks');
```

```
xlabel('Time','FontSize',12); ylabel('[Hz]','FontSize',12);
ylabel('Natural frequencies (Hz)','FontSize',12);
title(['\fontsize{14}Natural frequencies trend']);
```

code/plotfreq.m

A.12 savetemplate.m

```
function varargout = savetemplate(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
     'gui_Singleton',  gui_Singleton, ...
     'gui_OpeningFcn', @savetemplate_OpeningFcn, ...
     'gui_OutputFcn',  @savetemplate_OutputFcn, ...
     'gui_LayoutFcn',  [] , ...
     'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before savetemplate is made visible.
function savetemplate_OpeningFcn(hObject, eventdata, handles,
    varargin)
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = savetemplate_OutputFcn(hObject, eventdata,
    handles)
varargout{1} = handles.output;

% --- Executes on button press in image.
function image_Callback(hObject, eventdata, handles)
[FileName,PathName,FilterIndex] = uigetfile('*.jpg','Select
    template image');
setappdata(0,'tempimage',[PathName FileName]);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
setappdata(0,'tempname',get(handles.name,'String'));
setappdata(0,'tempdesc',get(handles.desc,'String'));
setappdata(0,'author',get(handles.author,'String'));
close(findobj('Tag','savetemp'));
```

code/savetemplate.m

Ringraziamenti

Les jeux sont faits! Dopo piú di duecento pagine scritte in inglese (sperando di non aver sfigurato piú del dovuto), questa pagina di ringraziamenti é per me, e probabilmente anche per chi legge, un benvenuto ritorno alla mia lingua madre.
Questa tesi ha impegnato in modo predominante gli ultimi sette mesi della mia vita ed in parte ha rappresentato per me una sfida, dato che la maggior parte dei temi toccati era per me nuova. Voglio quindi ringraziare in primo luogo i miei due correlatori, Kleidi e Mauro, che mi hanno aiutato attivamente in tutto questo tempo.
La tesi rappresenta, comunque, il termine di un percorso molto piú ampio che ho iniziato ormai parecchi anni fa, grazie al quale ho avuto modo di conoscere e apprezzare molte persone. Voglio approfittare di questo spazio per ricordarne un po', Marco, Laura, Carlo, Alberto e Alessandro che sono stati i migliori compagni di gruppo che potessi volere, Antonio e Massimo e Francesco, con cui ho condiviso la disperazione soprattutto dei primi esami. Vorrei nominare tutti ma un foglio non basterebbe.
Voglio ringraziare Elisa e i miei amici, in particolare Dario, Tommaso, Sara, e mia sorella Veronica, che hanno avuto un'enorme pazienza con me in questi mesi di tesi, e mi hanno ricordato di tanto in tanto che esiste una vita al di-fuori dello studio / lavoro.
Soprattutto, voglio ringraziare i miei genitori, Tiziana e Stefano, per avermi dato l'opportunitá di arrivare a questo punto e per avermi sostenuto, non senza sacrifici, moralmente e materialmente in tutti questi anni.