Politecnico di Torino

Master of Science in Civil Engineering

Academic Year 2018/2019

Bayesian inference of structural model parameters in an uncertainty quantification framework

MSc thesis

Supervisors:

Prof. Rosario Ceravolo

Ph.D. Gaetano Miraglia

Ass. Prof. Giuseppe Abbiati

Candidate:

Alessio Faraci

December 2019

If you torture the data long enough,
it will confess to anything.

Ronald H. Coase

A mio padre,
fondamenta sulle quali ho gettato
i miei principi di uomo.

A mia madre,
la sola al mondo che sa,
del mio cuore,
ciò che è stato sempre.

Abstract

In structural engineering, modeling fulfills a key role to simulate the behaviour of structures, but even very detailed models may fail to represent critical mechanisms. The uniqueness and uncertainties associated with civil structures make the prediction of the actual mechanical characteristics and the structural performance, a difficult task. Reliable estimates require calibration of system parameters based on measured experimental response data. To date, several different approaches have been adopted in literature. Generally, these ones try to minimize the difference between the model output and the experimental data. However, inverse problems (such as the estimate of mechanical parameters) when treated deterministically, are typically ill-conditioned and often ill-posed, since the values of parameters used to predict the structural behaviour are uncertain owing to simplifying and approximate assumptions on model. As consequence, these modeling uncertainties suggest that a single optimal parameter vector is not sufficient to specify the structural model, but rather a family of all plausible values of the model parameters consistent with observations needs to be identified.

A common accepted approach to deal with model uncertainties and experimental errors is to consider the identification problem from a statistical perspective. In the last decade, Bayesian model updating techniques became the standard tool for the identification of nonlinear dynamical systems. These techniques provide a robust and rigorous framework due to their ability to account model uncertainties and other sources of errors intrinsic in any system identification method as result of noise-measurements, as well as the partial model capacity to replicate the real physics of the system of interest.

This understanding has resulted in the need to model a discrepancy term to connect model prediction to the observations. The goal of this thesis work is to provide a fuller treatment of the posterior uncertainty linked to the discrepancy. Specifically, the Bayesian inference has been applied to the system identification of nonlinear systems not only to provide the most plausible model parameters and their probability distribution in uncertainty framework, but also to estimate the probability of the model discrepancy, i.e. a probabilistic estimate of the influence of the effects of measurement error and model inaccuracy on the prediction of system parameters. This Bayesian approach is illustrated first for parameters estimation of a basic nonlinear model using simulated data, and then applied to an experimental case study for the system identification of a planar masonry facade system with the application of the hybrid simulation/testing procedure for seismic response history.

The effectiveness of the proposed Bayesian inference of system identification in uncertainty framework lies in providing probabilistic information of the estimated parameters and on their error, which can be useful at the moment of making decisions with respect to the selection of parameters and/or the assessment of mathematical models that simulate the nonlinear behaviour experienced by the system.

Keywords: Bayesian inference, uncertainty quantification, surrogate modelling, polynomial chaos expansion, Kriging, hybrid simulations.

Contents

1 Introduction **1**
1.1 Structural Identification 1
1.1.1 Type of model employed in St-Id 2
1.1.2 Structural Identification stages 3
1.1.3 St-Id of nonlinear hysteretic systems 7

2 Bayesian inference for model calibration **11**
2.1 Inverse problems 11
2.2 Bayesian inference 12
2.2.1 Predictions 14
2.3 Bayesian inference for model calibration 15
2.3.1 Inverse solution 17
2.3.2 Model predictions 17
2.4 Markov Chain Monte Carlo simulations 18
2.4.1 Affine invariant ensemble algorithm (AIES) 19
2.4.2 Convergence of MCMC simulations 20
2.5 Surrogate modeling 20
2.5.1 Polynomial chaos expansions 21
2.5.2 Kriging 22
2.5.3 Surrogate modelling for stochastic dynamical systems . . . 24
2.5.4 Nonlinear Autoregressive with eXogenous input model . . . 24
2.5.5 PC-NARX 26
2.5.6 Kriging-NARX 30

3 Model validation and numerical benchmarks **31**
3.1 Bouc-Wen-Baber-Noori model of hysteresis 31
3.2 Model validation 33
3.3 Simulation of experimental record data 35
3.4 Calibration not accounting discrepancy 36
3.5 Calibration accounting random variable discrepancy 39
3.5.1 Reference model 40
3.5.2 NARX model 44
3.5.3 PC-NARX and Kriging-NARX surrogate models 45
3.5.4 Bayesian inversion using PC-NARX metamodel 48
3.5.5 Bayesian inversion with Kriging-NARX 51

4 Hybrid simulation tests of the masonry facade case study **55**
4.1 Hybrid simulations 55
4.1.1 Architecture of the hybrid simulator 56
4.2 Description of the case study 58
4.2.1 Materials 58
4.2.2 Hybrid simulations 60

5 NARX model for the masonry facade case study **69**
5.1 Cohesive Zone Models for mortar joints in masonry 69
5.2 FE model parameterization 71
5.3 NARX model 76

83

Bibliography **113**

List of Tables

3.1 Bouc-Wen-Baber-Noori parameters 36
3.2 BWBN parameters ranges 37
3.3 BWBN identified parameters 37
3.4 Prior distributions of the BWBN model parameters 39
3.5 Posterior marginals 43
3.6 Posterior marginals (PC-NARX) 48
3.7 Posterior marginals (Kriging-NARX) 51

4.1 Material specifications of clay brick Swiss K-Modul 15/19 [28] . . . 59
4.2 Compression test results of clay bricks [28] 60
4.3 Test results of mortar samples [28] 60
4.4 Compression test results of masonry specimens [28] 62
4.5 Material properties [28] 62
4.6 Test program [53] 64

5.1 McGarry model parameter distributions 72

List of Figures

2.1 Computation of LARS-based PC-NARX model, [40] 29

3.1 SDF Bouc-Wen model of hysteresis 31
3.2 Montenegro (1979) ground motion record 35
3.3 Model validation 35
3.4 Simulated experimental data 36
3.5 Pattern Search algorithm 38
3.6 Interior Point algorithm 38
3.7 Linear regressions for prior moments estimates 39
3.8 Prior samples 41
3.9 Trace plots and acceptance rate of the chains 42
3.10 Posterior samples 43
3.11 Model response 44
3.12 NARX free-run-reconstruction of the entire ED 45
3.13 NARX free-run-reconstruction for the experiment $k = 2$ 46
3.14 NARX coefficients cross-validation PCEs vs Kriging 46
3.15 PC-NARX - Kriging-NARX validation 47
3.16 PC-NARX and Kriging-NARX prediction on validation set 47
3.17 Trace plots and acceptance rate of the chains (PC-NARX) 49
3.18 Posterior samples (PC-NARX) 50
3.19 Model response (PC-NARX) 50
3.20 Trace plots and acceptance rate of the chains 52
3.21 fig: Posterior samples (Kriging-NARX) 53
3.22 Model response (Kriging-NARX) 53

4.1 Hybrid Simulation loop [28] 56
4.2 Architecture of the hybrid simulator [28] 57
4.3 Masonry wall 59
4.4 Bending and compressive strength testing [28] 61
4.5 Test setup for compression test of masonry specimens [28] 61
4.6 Architecture of the PSD-HS setup [28] 63
4.7 Test setup for DIC analysis [28] 63
4.8 1979 Montenegro earthquake 64
4.9 Restoring forces hysteresis loops measured during Test #5 65
4.10 Displacement responses during Test #5 66
4.11 Von Mises strain field measured via DIC during Test #5 67
4.12 Overview of the wall specimen after Test #5 68

5.1 Validation implementation 72
5.2 FE discretization 73
5.3 Model parameterization 74
5.4 Model parameterization on filtered data-set 75
5.5 NARX$^{(k)}$ free-run-reconstruction of the entire ED 77
5.6 NARX$^{(l)}$ free-run-reconstruction of the entire ED 78
5.7 NARX$^{(m)}$ free-run-reconstruction of the entire ED 78
5.8 NARX$^{(}k$) free-run-reconstruction of N44 in x-direction 79
5.9 NARX$^{(}l$) free-run-reconstruction of N44 in y-direction 79
5.10 NARX$^{(}m$) free-run-reconstruction of N55 in y-direction 79

Chapter 1

Introduction

Nowadays civil engineers are routinely using finite element (FE) models to simulate the behaviour of structures for practical applications. However, there are several examples that show even very detailed models may miss critical mechanisms. The uniqueness and uncertainties associated with civil structures render their actual mechanical characteristics and performance parameters extremely difficult to predict. Reliable estimates of the performance and vulnerability of structural systems require calibration and validation based on actual observations and measured experimental data. This understanding has resulted in the need to improve model predictions using experimental response data, and has fueled on the development of Structural Identification (St-Id).

There are several scenarios, which may justify the identification of a field-calibrated analytical model for simulating an actual structure: (1) design and verification; (2) documentation of as-is structural characteristics as baseline for assessing any future changes due to aging and deterioration; (3) evaluation of possible causes and mitigation of deterioration, damage, and/or other types of performance deficiencies; (4) health and performance monitoring for maintenance management; (5) structural intervention, modification, retrofit or hardening.

The aim of St-Id is to bridge the gap between the model and the real system. As direct consequence, St-Id has the potential to reduce the need for excessive conservatism in the face of uncertainty, and to expand the assessment of structural performance.

From this prospective, St-Id has attracted the attention of numerous researchers worldwide over the last several decades. It is the goal of this section to benchmark and provide an overview of these developments, which constitute the current state-of-the-art.

1.1 Structural Identification

According to [16], Structural Identification (St-Id) can be defined as: *"the parametric correlation of structural response characteristics predicted by a mathematical model with analogous quantities derived from experimental measurements"*.

St-Id has its origins in systems engineering during the late 1950's. The advent of the computer permitted extensive simulation and evaluation of systems and components. These advances contributed to the development of the system

identification (Sys-Id) concept, defined as the estimation of a system based on the correlation of inputs and outputs. St-Id is a transformation and application of Sys-Id to civil structural systems.

The concept of parameter identifiability is defined as the ability to determine the unknown parameters of a system from input-output measurements. This is known as the *identification problem*. Properly speaking, the goal of the identification problem is to make precise for other purposes, such as prediction, design, control etc., a certain given incomplete mathematical description of the system by employing appropriate a priori and experimental information [24].

Clearly, the identification problem is an inverse problem and, as such, is associated with a number of difficulties of an analytical and numerical nature.

1.1.1 Type of model employed in St-Id

The following sections provide a description of St-Id research and developments by the type of model employed.

Physics-Based Models

Physics-based (PB) models are formulated to explicitly address the boundary and continuity conditions, equilibrium and kinematics of the constructed system of interest. In PB approaches the identified model can be used to explicitly simulate behaviour under various critical loading conditions. Such models can diagnose the causes of changes in behaviour as well as identify how such changes may impact the performance of the overall system. While several researchers have investigated the use of nonlinear models [24], [74], [5], [48], [49], [34], [75], currently the most commonly employed PB St-Id approach relies on linear matrix structural analysis or FE models. In general terms PB models can be considered predictive as they rely heavily on the generalized laws of statics, mechanics, dynamics, etc. This basis, which does not require response data from the constructed system, allows such models to be useful in a priori sense.

Non-Physics-Based Models

Researchers have also investigated the use of many different types of non-physics-based (NPB) models for St-Id, including Artificial Neural Networks (ANNs) [66], [47], [12], [56], wavelet decomposition [30], [35], [55], auto-regressive moving average vector (ARMAV) models [4], [10], [72], state space models [7], [38], [3], [23], [27], [58], and Empirical Mode Decomposition (EMD) in conjunction with the Hilbert-Huang Transform. The main advantage of these techniques is that they are data-driven, i.e., the construction of NPB models is solely dependent on the data provided. NPB models are descriptive in nature. They are not based on specific generalized laws but are derived principally from various means of data modeling, reduction and interpretation. As such, these models are not appropriate for a priori use. However, once NPB models are trained through the use of response data, they may be considered predictive as they are then capable of estimating future response through forecasting identified patterns and thus identifying when the

system has changed. This data driven nature makes them attractive for modeling complex phenomena, automation, real-time St-Id, continuous monitoring, and minimizing errors due to user interaction. But it is equally important to recognize that they can only identify whether a change in behaviour that corresponds to the data recording process has occurred and cannot (in the absence of PB techniques) identify the cause of the change of its affect on overall performance. More importantly, until many decades of data with sufficient density and bandwidth is captured and analyzed, it will not be possible to definitively identify and differentiate between "normal" and "abnormal". Mitigation of measurement errors remains a significant and often unrecognized problem.

1.1.2 Structural Identification stages

Following [16], St-Id of Constructed Systems can be organized in six steps:

STEP 1 - Objectives, Observation and Conceptualization

The first step of St-Id involves becoming familiar with the issue that is driving the application as well as the structure itself.

Our current knowledge on behaviour and performance of constructed systems is greatly incomplete. It follows that in order to properly guide a St-Id application it is critical that potential uncertainties have to be identified at the first stage. If the structure is not properly conceptualized in its current state it is likely that some potentially significant behaviour mechanisms that have uncertainty associated with them may be under appreciated. This may lead to poor model construction and/or incomplete experimental design, which will in turn influence each step of the process. In such cases, the St-Id will at best result in inconclusive results.

The data, information and knowledge that are available about a system that will be identified would serve as important constraints and drivers for the analytical modeling, measurements and controlled experiments, and model-calibration.

STEP 2 - A Priori Modeling

The development of an a priori model within the structural identification process serves to help conceptualize a structure, identify key structural responses that will aid in the selection of appropriate experimental approaches. The effect of modeling assumptions should be examined through the comparison of several modeling approaches and through sensitivity analyses. Depending on the objectives of the St-Id, the a priori model may also serve as the model calibrated through parameter identification.

Most a priori models are based on assumptions of linearity and stationarity because, in the absence of response data from the specific constructed system, it is difficult to justify the complications associated with nonlinear constitutive relations.

The most pertinent distinction between the numerous PB modeling approaches lies in its ability to identify key mechanism and provide an expected range of response to allow an efficient and robust experimental program to be designed and carried out. (Structural Models, Finite Element Models)

The most common PB models employed as a priori models are:

STEP 3 - Experimentation

A fundamental component of the Structural Identification (St-Id) process is the experimental process leading to "data" in various forms and at various levels of refinement, that are used in the analysis tools to decode the performance of a structure. St-Id uses the results from static and dynamic measurements as a first step towards developing more reliable conceptual or numerical models. These models are used to evaluate and predict structural performance, and to support operational and maintenance decisions. From this prospective, experimental methods and technologies serve as the quantitative link to the constructed system of interest. As such, this step is indispensable. The fundamental challenge in experimenting with actual constructed systems is to acquire the most meaningful data, and minimize the uncertainty inherent in the data to facilitate its effective interpretation. This challenge requires more than the minimization of random and bias errors caused by the sensors themselves, which can be mitigated by employing established best practices for both sensor calibration and installation.

STEP 4 - Data processing and data interpretation

Unfortunately, in contrast with many other engineering areas, sensors in structural engineering rarely measure causes directly; causes must be inferred from measured effects. Even when causes can be measured in complex structures, it will never be possible to measure directly every possible phenomenon of interest at every location. Thus, without appropriate methods for data interpretation, structural identification cannot provide useful engineering support.

Step 4 of the St-Id process involves the processing and interpretation of data. In general terms data processing activities aim to make the acquired data more appropriate for interpretation. This is typically achieved through cleansing the data of blatant and subtle errors, improving the quality of the data, and then compressing and/or transforming the data to better support interpretation.

The second stage of Step 4 is concerned with data interpretation. There are two main types of data interpretation and they are distinguished by the use or absence of a physics-based behaviour model.

Non-parametric models are defined as non-physics-based numerical models that in some cases allow data condensation and reconstruction using a limited number of parameters. This approach does not require the development and use of a behaviour model of the structure. Therefore, they are much less onerous to implement. Consequently, they have potential to be used on a large number of structures. For this type of model, the structural identification process is generally a parametric curve-fit of mathematical functions to the measured data. Although the functions reproduce to a certain level of accuracy the measured data, the parameters themselves do not have any direct physical interpretation. The primary goal of this approach is to detect anomalies in behaviour. Anomalies are detected as a difference in measurements with respect to measurements recorded during an initial period. More specifically, this approach involves examining changes over a certain period during the life of a structure. The methodology is data driven in the

sense that the evolution of the data is estimated without information of physical processes. So the second stage of Step 4, direct data interpretation, involves fitting mathematical models (also referred to as a non- physics-based model), such as Artificial Neural Networks, Auto-Regressive Models, state space models etc., to the processed data. These models are not formulated with any consideration of the underlying physics of the constructed system, rather they aim to accurately capture and replicate the patterns associated with the data. In this manner, they are most concerned about identifying when the constructed system behaviour has changed rather than identifying the underlying cause of the change. This approach has advantages of require minimal user interaction and being able to address large data sets, and, as a result, is a powerful tool for continuous monitoring of structures.

Examples of the data-driven models include autoregressive models (AR) (and variants such as ARMA, ARX and ARMAX models) and the rational polynomial model. The following paragraphs focus on techniques used for direct data processing and interpretation (without the need for physics-based or parametric models), and on methods applied to constructed systems. The following approaches are organized by their primary function: anomaly detection and data processing, data reduction and representation, and feature extraction.

Data Reduction and Representation. Since it is difficult to characterize data in a high dimensional space, it is often necessary to extract low dimensional features for data analysis. Anomaly detection and data processing may be further augmented by a variety of frequency, time, and time-frequency approaches that enable data to be reduced by a fixed number of parameters. This enables the dominant modes or components to be easily recognized and thus aids in data storage and signal reconstruction, particularly in the reduction of noise. Many of these methods, and the compact representations they offer, further allow the dynamics of the system to be characterized, including potential damage.

Autoregressive Methods. In light of the aforementioned resolution issues, feature extraction is often conducted strictly in the time domain using a set of algebraic and temporal relationships among outputs, and in some cases inputs, of systems. Such relationships are useful for predicting values of sensor measurements from measurements of other sensors. Predicted values are then compared with the measured values from those sensors. A temporal redundancy is obtained observing how the differential or difference relationships among different sensor outputs and inputs evolve with the time. A simple relationship for characterizing a system is a polynomial mapping between system inputs (when available) and outputs. One such representation referred to as an autoregressive-moving average with noise, characterizes the system as a weighted polynomial of past outputs (autoregressive - AR) and past and present inputs (moving averages - MA). The output is a linear combination of the input history and the past outputs. The input series is a causal moving average (MA) feed-through process, and the series involving weighted past output values is an autoregressive (AR) process. AR (single and multi-variate), ARMA and ARX representations have all been used to represent measured responses of structures. In situations where the behaviour of the structure varies, it is possible to calculate coefficients incrementally. This approach is also useful

for assessing whether significant information regarding new events can be obtained through observing the autoregressive model. Through observing changes in coefficients, unusual events such as sudden foundation settlement, ground movement, excessive traffic loading and failure of post-tensioning cables, can be revealed. In fact, evaluations of the model coefficients and residual errors against baseline values for the structure are also capable of detecting anomalies associated with even minor levels of damage.

Data Mining. Data mining is a field of research concerned with finding patterns in data for both understanding and prediction purposes. Data mining algorithms are especially useful when dealing with amounts of data that are so considerable that human processing is infeasible. This is often the situation in structural identification tasks, as visualizing distributions of models in multi-dimensional parameter spaces is difficult for engineers without suitable computing tools.

Feature Selection and Extraction. Feature selection is a method used to reduce the number of features (parameters) in order to facilitate data interpretation. Irrelevant features may have negative effects on a prediction task. Moreover, the computational complexity of a classification algorithm may suffer from excessive dimensionality caused by several features. When a data set has too many irrelevant variables and only a few examples, over-fitting is likely to occur. In addition, data are usually better characterized using fewer variables. Feature selection is an effective method for supporting system identification since it identifies parameters that explain predictions of candidate models.

STEP 5 - Calibration of Models

Step 5 of the St-Id process involves the selection and calibration of physics-based models. These models, in contrast to the non-physics-based models used for direct data interpretation, are formulated to explicitly recognize the underlying physics of the constructed system. Owing to the uncertainties, it is clear that simply developing a finite element model with typical engineering assumptions and idealizations may not be sufficient. Rather it is recommended that several different modeling strategies be employed and compared to ensure the model selected for calibration is appropriate. The model calibration process typically involves optimizing a set of model parameters to minimize the difference between the initial model and the experimental results. Approaches to this model calibration (also known as model updating) can be classified based on how they select the parameters to identify, the formulation of their objective functions (to minimize), the optimization approach they employ (e.g., gradient-based or non-gradient-based), and whether or not they explicitly address uncertainties, among others.

Once the model form and space has been determined, an appropriate technique must be selected to identify the parameters of that model. Once structural parameters have been identified from experimental data, they can be used to calibrate numerical models so that their response predictions correlate well with the measured response of the physical system. Differences between in-situ and predicted structural parameters and responses may arise from simplifications employed in

the modeling process, e.g., in the representations of the boundary and support conditions, connectivity between various structural elements, unknown material properties and constitutive relationships (particularly those associated with soil and concrete), and energy dissipation (damping) mechanisms as well as measurement errors. The calibration process involves selecting a small number of model parameters that have uncertainty so their values cannot be known a priori. Once these parameters are selected, various procedures are used to find their values for which the measurements best match the model predictions. This process then naturally enables updating the analytical models such that they more accurately predict the observed response of the in situ structural system. However, despite the advances in finite element modeling, model calibrations of full-scale structures can easily be in error by as much as 50%, indicating that validation of a particular behaviour model is a non-trivial exercise [16]. This process can be particularly challenging due to the degree of freedom mismatch, as the number of response measurement locations is significantly less than the number of degrees of freedom in the finite element model. This mismatch often makes it difficult to precisely identify the portions of the model that cause the discrepancies between measured and predicted response. No matter the approach used, the result of a successful calibration effort is a model suitable to provide owners and managers of that infrastructure with the information necessary for decision making related to rehabilitation and maintenance. A calibrated model enables a more pro-active maintenance that can be substantially more economical than delayed responses to deterioration.

STEP 6 - Models for Decision-Making

The use of the models developed and calibrated (physics-based) or trained (non-physics-based) through the St-Id process is essential to influence the decision-making process. Properly leveraging a calibrated analytical model through scenario analysis, parametric studies, or what-if simulations, in order to influence decisions is related to improving the performance of a design at different limit-states, or to evaluating the future performances of an existing as-constructed system, influencing decisions is a crucial part of the St-Id process. In the future, especially as an increasing portion of civil engineering expenditures relate to renewal of existing constructed systems, simulation-based management of our constructed environment will be essential, and will rely on reliable applications of St-Id.

1.1.3 St-Id of nonlinear hysteretic systems

The phenomenon of hysteresis is displayed in many systems of engineering interest. Examples include systems in structural dynamics being stressed beyond their elastic limit (nonlinear hysteretic behaviour is seen commonly in structures experiencing strong ground earthquake excitation), aircraft structures subjected to acoustic or aerodynamic loads [5]. Because of the hysteretic nature of the restoring force in such situations (hysteresis is a highly nonlinear phenomenon), the nonlinear force cannot be expressed in the form of an algebraic function involving the instantaneous values of the state variables of the system.

Hysteresis is typical of a class of functions which are multi-valued. Consequently, much effort has been devoted by numerous researchers to develop models

of hysteretic restoring forces and techniques to identify such systems. The identification of hysteretic-type nonlinearities is of great importance in the design of earthquake-resistant structures.

Early notions on identification of nonlinear hysteretic systems are contained in [5]: in 1982, Andronikou et al. presented the application of an adaptive random search algorithm to the identification of parameters in single-degree-of-freedom (SDOF) systems containing hysteretic nonlinearities through which the hysteretic restoring force was represented by a bilinear model with three unknown parameters identified using sinusoidal inputs. In the same year, S. F. Masri et al. [45] presented a non-parametric identification technique for chain-like multidegree-of-freedom nonlinear dynamic systems that used information about the state variables of nonlinear systems to express the system characteristics in terms of two-dimensional orthogonal functions to identify dynamic systems with arbitrary nonlinearities.

Benedettini et al. [7] investigated nonparametric models defined by two different descriptions: the first, in which the restoring force is a function of displacement and velocity; and the second, in which the incremental force is a function of force and velocity. It is shown in their work the ability of the second variable space to better reproduce the behaviour of hysteretic oscillators. The approximation of the real restoring function is done in terms of orthogonal (Chebyshev) polynomials and nonorthogonal polynomials by assuming as state variables the force itself and velocity; while mixed parametric and nonparametric model were used in the case of important hardening and viscous damping.

S. F. Masri et al. [46] extended this approach by proposing a polynomial base approximation of the restoring force as a function of velocity, displacement and the excitation.

In the frame of nonparametric approaches, Pei et al. [58] used a special type of neural network, which showed good performances in the identification of hysteretic systems.

When a structural system subject to earthquake loading exhibits degradation or behaves as time variant, we must consider instantaneous and possibly online estimation techniques to perform a nonlinear identification. On-line identification of degrading and pinching hysteretic systems is quite a challenging problem because of its complexity. The method generally followed for the on-line identification of hysteretic systems under arbitrary dynamic environments is the use of adaptive estimation approaches [20], [19], [64]. The availability of such an identification approach is crucial for the on-line control and monitoring of nonlinear structural systems to be actively controlled. A recently developed technique, the unscented Kalman filter (UKF) which is capable of handling any functional nonlinearity, is applied in [71], [21] and [37], to the on-line parametric system identification of hysteretic differential models with degradation and pinching. Simulation results show that the UKF is efficient and effective for the real-time state estimation and parameter identification of highly nonlinear hysteretic systems.

A technique for the structural identification of hysteretic oscillators that are characterized by degradation in stiffness has been proposed in [17] in which it is considered the possibility to replace the expression of the time derivative of the restoring force with a polynomial approximation, characterized by time-varying

coefficients. The instantaneous estimation, based on optimization techniques, is made possible through the temporal localization of frequency components, i.e., the representation in the joint time–frequency domain.

A classification of the identification methods for nonlinear systems can be done also on the domain in which the identification is performed: frequency, time, or joint time-frequency domain, e.g. in 2012, Bursi et al. [14] consider the instantaneous-based identification technique applied to nonlinear systems, which relies on a time-frequency approach. The entailing formulation determines model parameters that minimize an error function between time-frequency representations of measured and simulated signals, respectively. To identify the hysteretic behaviour of a nonlinear steel-concrete composite structure, authors combine the instantaneous-based identification technique with a parametric method to provide instantaneous estimates of parameters. The instantaneous estimate of the parameters can be then used to check the consistency of a given model by assessing the stability of the parameters value in time [18].

As aforementioned hysteresis and nonlinear behaviour have been the subject of numerous previous studies, including the development of models for bilinear hysteresis, yielding structures, degrading systems and other hysteretic systems and structures. Many models have been used for capturing nonlinear dynamical systems, including single-valued models, distributed element models, modal models.

Several different approaches have been adopted for identification of nonlinear systems. The approaches include stochastic linearization techniques, Bayesian models [36], nonparametric methods using polynomial basis functions, enhanced response sensitivity approach [39], optimization algorithms and neural networks. The Bouc–Wen, in particular, has seen its parameters estimated via nonlinear optimization schemes, Bayesian state estimate with bootstrap filters, adaptive on-line methods, and applications of the extended Kalman filter (EKF) and the unscented Kalman filter (UKF) [15], [37]. Recent developments in nonlinear identification include the use of state-space models, auto-regressive (AR) models, nonlinear regression models and cellular automata nested neural networks.

Within the realm of nonlinear identification, on-line identification schemes are of paramount importance because they allow for the incorporation of flexible controller strategies that adapt with the structure, as structures that behave nonlinearly may only exhibit their governing response properties when excited by strong motions. While there have been several developments in on-line nonlinear identification, one of more versatile methods involves the use of adaptive neural networks. Specifically, an adaptive approach that utilizes Volterra/Wiener neural networks (VWNNs) has been shown to be a highly effective estimator of nonlinear responses. The Volterra/Wiener neural network (VWNN) has been shown to be an effective tool for on-line estimation of nonlinear restoring forces and responses. However, the power of the VWNN for on-line identification has not been fully harnessed due to the high sensitivity of its parameters. A probabilistic approach in examining the effects of the VWNN's parameters on the robustness and stability of its estimation capabilities is presented in [12].

Chapter 2

Bayesian inference for model calibration

This chapter covers the core of this thesis work. It concerns the theory behind statistical approaches towards inverse problems are dealt with, focusing on one of the most accepted probabilistic approaches for uncertainty quantification (UQ): the Bayesian inference method. The benefits of this method are its rigorous treatment of uncertainties by the explicit use of probability for representing, propagating and updating uncertainty on model parameters, and its reliability in inverse problems. Of course, the theory developed is based on Bayes's theorem and conditional probabilities. In this framework, the a priori information is the probability distribution of the model variables over the *"model space"*. This distribution captures the epistemic uncertainty of the unknowns before the data are analyzed. Section 2.2 explains how this *a priori probability distribution* is updated into an *a posteriori probability distribution* by combining the prior knowledge and the evidence of actual observations to capture the remaining uncertainty once the data have been processed. Whereas in Section 2.3 it is pointed out how to apply Bayesian inference in the context of model calibration, and how the practical computation of this posterior distribution generally requires the use of sampling methods. For this purpose, Monte Carlo sampling algorithms are employed and discussed in Section 2.4. As well it is called attention to their low convergence rate jointly to computational prohibitive time-consuming issues (Section 2.4.2). These constitute crucial issues that need to be managed. To this aim, finally, in Section 2.5 surrogate models based on Polynomial Chaos Expansions (PCE) and Kriging (Gaussian Process) are introduced jointly with Nonlinear Autoregressive with eXogenous input models (NARX) for stochastic dynamical systems.

2.1 Inverse problems

Physical theories usually model the laws governing the real world as systems of differential equations in order to let us to make predictions on the outcome. A typical feature of these sets of equations is *causality*: forward conditions depend on the previous ones [33]. This class of problems constitutes the so called *forward*

problems or *direct problems*. On the contrary, an *inverse problem* arises, as seen in Chapter 1, when the actual result of some experimental measurement data is indirectly used to infer the values of the unknown parameters describing the system, whose values cannot be directly measured. Thus, the aim is to propagate evidence about acquired data backwards to get an insight on the model parameters [67]. It is clear from that as this class of problems is endowed with *non-causality* which remarkably contributes to their instability [33].

While direct problems have a unique solution, inverse problems do not. Because of this, any available a priori information about model parameters, as well as the representation of the data uncertainties, play a role of great significance.

Deterministic parameter estimations of a numerical model find a set of optimal parameters that best fit the observed data. They are nothing but optimization problems of minimizing the discrepancy between computed and measured data. However, such approaches bump into some common problems; indeed they are: prone to be ill-posed and ill-conditioned, extremely susceptible to errors, and affected by uniqueness and stability issues [22]. These factors coupled with the uncertainty derived from the acquisition of data measurements, result in uncertainty on model prediction. Accounting for these uncertainties is crucial and necessary, and naturally leads to consider the problem from a probabilistic perspective.

The philosophy behind probabilistic methods is to restate the inverse problem purely in the statistical form: the objective is to acquire knowledge, assess the uncertainty, and draw conclusions about the quantities that cannot be observed starting from data at hand and available prior informations. This procedure leads to remove the ill-posedness by extending the solution in the wider space of probability distributions. As consequence, the solution of an inverse problem produces not a single estimate of the unknowns but a probability distribution from which producing estimates that have, evidently, different probabilities. Quoting Kaipio and Somersalo [33]: *"the proper question to ask is not: what is the value of this variable? But rather: what is our information about this variable?"*.

The advantage of applying an uncertainty quantification (UQ) framework is that allows assessing both the effect of uncertainty on the model parameters and the uncertainties on derived quantities, e.g. response predictions, and evaluating the robustness of the model against uncertainty.

2.2 Bayesian inference

Bayesian inference is *"the process of fitting a probability model to a set of data and summarizing the result by a probability distribution on the parameters of the model and on unobserved quantities such as predictions for new observations"*, Gelman et al. [26]. In the context of Bayesian statistics, all model parameters are supposed being random variables. This randomness describes, by means of joint probability distributions, the degree of uncertainty related to their realizations, namely the *a priori* information. This a priori probability is then updated into an *a posteriori* probability distribution (i.e. a conditional probability distribution of the unknown quantities), by combining the prior knowledge and evidence of actual observations using Bayes's theorem and conditional probabilities. With this in mind, the so-

lution of an inverse problem coincides with the posterior probability distribution, i.e. the probability distribution associated to a certain variable of interest, when all available informations have been included in the model. This posterior distribution reflects the degree of confidence, or belief, about that quantity once measurement has been performed.

As general notation, let $\mathcal{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_N\}^\intercal$ denote the observed data set. This one is supposed to be made of independent realizations of a random vector $\boldsymbol{Y}$, where the $\boldsymbol{y}_i$ variables are called the *random outcomes* with probability density function (PDF) $\pi(\boldsymbol{y})$. The shape of the latter, in parametric statistical models, is assumed being parameterized. Let then $\boldsymbol{\theta} = \{\theta_1, ..., \theta_M\}$ denote the vector of these unknown parameters (or *hyperparameters*) of interest. Therefore:

$$\boldsymbol{Y} \sim \pi(\boldsymbol{y}|\boldsymbol{\theta}) \tag{2.1}$$

generally these symbols represent multivariate quantities.

The inference goal is to estimate the hyperparameters $\boldsymbol{\theta}$ of the PDF above from the observed data. These ones are modeled as a random vector following the *prior distribution* $\pi(\boldsymbol{\theta})$ which reflects the grade of knowledge about $\boldsymbol{\theta}$ before any observation of $\boldsymbol{Y}$:

$$\boldsymbol{\Theta} \sim \pi(\boldsymbol{\theta}) \tag{2.2}$$

Bayesian statistics draws conclusions making probability statements about the parameters $\boldsymbol{\theta}$ given $\boldsymbol{y}$, using *Bayes' theorem*[1]:

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{\pi(\boldsymbol{\theta})\,\pi(\boldsymbol{y}|\boldsymbol{\theta})}{\pi(\boldsymbol{y})} \tag{2.3}$$

where the posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ of the hyperparameters is obtained by conditioning the *joint probability distribution*, i.e. the product between the prior $\pi(\boldsymbol{\theta})$ and the *sampling distribution* (or *data distribution*) $\pi(\boldsymbol{y}|\boldsymbol{\theta})$, with the known value of the data $\pi(\boldsymbol{y})$. The latter is given by the sum over all possible values of $\boldsymbol{\theta}$:

$$\pi(\boldsymbol{y}) = \sum_{\theta} \pi(\boldsymbol{\theta})\,\pi(\boldsymbol{y}|\boldsymbol{\theta}) \tag{2.4}$$

or, if $\boldsymbol{\theta}$ is continuous, by the integral:

$$\pi(\boldsymbol{y}) = \int \pi(\boldsymbol{\theta})\,\pi(\boldsymbol{y}|\boldsymbol{\theta})\,d\boldsymbol{\theta} \tag{2.5}$$

[1] ⟨ Bayes' theorem gives the probability that some hypothesis, say H, is true given an event E. To calculate this, the prior probability before got the event E, i.e. the probability of the hypothesis was true $\mathbb{P}(H)$, needs to be multiplied by the probability of the event given the hypothesis true $\mathbb{P}(E|H)$, and divided by the total probability of the event occurring $\mathbb{P}(E)$, i.e.: $\mathbb{P}(H|E) = \frac{\mathbb{P}(H)\,\mathbb{P}(E|H)}{\mathbb{P}(E)}$ ⟩.

Considering now the fact that, for a fixed $\boldsymbol{y}$, $\pi(\boldsymbol{y})$ does not depend on $\boldsymbol{\theta}$, an equivalent form of equation (2.3) can be written omitting the factor $\pi(\boldsymbol{y})$:

$$\pi(\boldsymbol{\theta}|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\,\pi(\boldsymbol{y}|\boldsymbol{\theta}) \tag{2.6}$$

where $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ is taken here as function of $\boldsymbol{\theta}$ and not of $\boldsymbol{y}$.

Once chosen a subjective model for the prior probability distribution, Bayes' rule (2.6) states that the data $\boldsymbol{y}$ affects the posterior inference only through the term $\pi(\boldsymbol{y}|\boldsymbol{\theta})$. When this term is taken as function of $\boldsymbol{\theta}$ for independent realizations of $\boldsymbol{y}$, is called the *likelihood function* $\mathcal{L}$, a function that simply expresses the probability of observing the data $\mathcal{Y}$ given $\boldsymbol{\theta}$:

$$\mathcal{L}(\boldsymbol{\theta};\mathcal{Y}) := \prod_{i=1}^{N} \pi(\boldsymbol{y}_i|\boldsymbol{\theta}) \tag{2.7}$$

In this way Bayesian inference obeys to the *likelihood principle*: for a given sample of data, any two probability models $\pi(\boldsymbol{y}|\boldsymbol{\theta})$ that share the same likelihood, yield the same inference for $\boldsymbol{\theta}$ [26].

Substituting equation (2.7) in (2.3):

$$\pi(\boldsymbol{\theta}|\mathcal{Y}) = \frac{\pi(\boldsymbol{\theta})\,\mathcal{L}(\boldsymbol{\theta};\mathcal{Y})}{Z} \tag{2.8}$$

where Z stands for a normalizing factor, named *evidence* or *marginal likelihood*, that ensures that this distribution integrates to 1, and is defined by the integral:

$$Z = \int_{\mathcal{D}_\theta} \pi(\boldsymbol{\theta})\,\mathcal{L}(\boldsymbol{\theta};\mathcal{Y})\,d\boldsymbol{\theta} \tag{2.9}$$

This simply equation (2.8) represents the core of Bayesian inference: the posterior distribution $\pi(\boldsymbol{\theta}|\mathcal{Y})$ contains all the information inferred about the hyperparameters by combining the prior knowledge $\pi(\boldsymbol{\theta})$ on hyperparameters and the observed data $\mathcal{Y}$. In this sense, it can be seen as an "update" of the prior knowledge on $\boldsymbol{\theta}$.

Except for specific distributions of parameters and for particular choices on the prior for which an analytical solutions exist (*conjugate distributions*) [26], the practical computation of posterior distributions requires the use of sampling methods, i.e. a resolution of the a posteriori probability distribution in terms of "samples". For this purpose, Monte Carlo sampling algorithms are employed (SECTION 2.4).

2.2.1 Predictions

In the UQ framework, it is useful not to uniquely find out the posterior distribution $\pi(\boldsymbol{\theta}|\mathcal{Y})$, but also to propose the *"best distribution"* for the assessment of $\boldsymbol{Y}$ by selecting a *point estimator* $\hat{\boldsymbol{\theta}}_0$, i.e. a particular value selected from the posterior

distribution. Common choices of the latter are the posterior mean (mean of equation (2.8)), and the posterior mode, a.k.a. MAP (i.e. maximum a posteriori, that is the mode of equation (2.8)). However, following this approach, the estimation of uncertainty associated to the parameters is neglected [67].

To get through this, and to make inferences about an unknown observation (*predictive inferences*), the uncertainty on $\boldsymbol{\theta}$ can be incorporated into the prior and posterior assessment of $\boldsymbol{Y}$ by *predictive distributions*:

$$\pi'_{pred}(\boldsymbol{y}) := \int_{\mathcal{D}_\theta} \pi(\boldsymbol{y}|\boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, d\boldsymbol{\theta} \tag{2.10}$$

$$\pi''_{pred}(\boldsymbol{y}^\star|\mathcal{Y}) := \int_{\mathcal{D}_\theta} \pi(\boldsymbol{y}^\star|\boldsymbol{\theta})\, \pi(\boldsymbol{\theta}|\mathcal{Y})\, d\boldsymbol{\theta} = \frac{1}{Z} \int_{\mathcal{D}_\theta} \pi(\boldsymbol{y}^\star|\boldsymbol{\theta})\, \mathcal{L}(\boldsymbol{\theta};\mathcal{Y})\, \pi(\boldsymbol{\theta})\, d\boldsymbol{\theta} \tag{2.11}$$

Equation (2.10) is the *prior predictive distribution*, that is the marginal distribution of the unknown but observable $\boldsymbol{y}$, i.e. before the data $\mathcal{Y}$ are considered (prior because it is not conditional on a previous observation, predictive because it is the distribution for a quantity that is observable [26]). After the observation of $\mathcal{Y}$, a prediction of an unknown but observable, say it $\boldsymbol{y}^\star$, can be made by the *posterior predictive distribution* (2.11), that is conditional on the observed data $\mathcal{Y}$ (posterior because it is conditional on the observed $\mathcal{Y}$, predictive because it is a prediction for an observable $\boldsymbol{y}^\star$ [26]).

2.3 Bayesian inference for model calibration

In engineering, analysis of physical systems are performed by means of computational models. A computational *forward* model $\mathcal{M}$ is usually defined as a function that maps a set of model input parameters governing the system $\boldsymbol{x}$ to predict certain output quantities of interest (QoI) $\tilde{\boldsymbol{y}}$:

$$\mathcal{M} : \boldsymbol{x} \in \mathcal{D}_\theta \subset \mathbb{R}^M \longmapsto \tilde{\boldsymbol{y}} = \mathcal{M}(\boldsymbol{x}) \in \mathbb{R}^N \tag{2.12}$$

Computational models are commonly established based on equations governing the system itself (e.g. mechanics, dynamics), or on numerical methods. In the context of computational modeling and uncertainty quantification, *Bayesian model calibration* aims at identifying the input parameters $\boldsymbol{x}$ of $\mathcal{M}$ that allow to recover the observations $\mathcal{Y}$. Specifically for this thesis work, the final focus is to identify unknown properties and key components of complex systems, based on their observed response to controlled external loads in laboratory experiments.

In the statements of Bayesian inverse problems, the lack of knowledge (i.e. the epistemic uncertainty) on the input parameters, is taken into account by considering the input parameters as a random vector $\boldsymbol{X} \sim \pi(\boldsymbol{x})$. The forward model is a mathematical representation of the real system. As a matter of principle, all models are always simplifications of the real world. As consequence, a *discrepancy term* shall be introduced to connect model predictions $\tilde{\boldsymbol{y}} = \mathcal{M}(\boldsymbol{x})$ to the experimental

observations $\mathcal{Y}$ [67], i.e.:

$$\boldsymbol{y} = \mathcal{M}(\boldsymbol{x}) + \varepsilon \tag{2.13}$$

In this discrepancy term are gathered the effects of *measurement error* on $\boldsymbol{y} \in \mathcal{Y}$ and *model inaccuracy*. In general case, whatever distribution can be used to model ε, but for the sake of simplicity the assumption of a simple *additive Gaussian discrepancy* with mean null and residual covariance matrix $\boldsymbol{\Sigma}$ has been made in this work, i.e.:

$$\varepsilon \sim \mathcal{N}(\varepsilon|\boldsymbol{0}, \boldsymbol{\Sigma}) \tag{2.14}$$

Assuming $\boldsymbol{\Sigma}$ perfectly known (i.e. assuming a discrepancy term with known parameters) is essentially unrealistic because ε is a quantity not known a priori in many practical situations [67]. By parametrizing the residual covariance matrix this issue can be overcome. Namely $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\boldsymbol{x}_\varepsilon)$, where its parameters $\boldsymbol{x}_\varepsilon$ are additional unknowns to infer jointly with the input parameters $\boldsymbol{x}_\mathcal{M}$ of $\mathcal{M}$.

Again, to keep things simple, a diagonal covariance matrix with unknown residual variances σ^2 is assumed, specifically $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$. This results in reducing the discrepancy parameter vector to a single scalar, i.e. $\boldsymbol{x}_\varepsilon \equiv \sigma^2$. This assumption leads in setting the parameter vector as $\boldsymbol{x} = (\boldsymbol{x}_\mathcal{M}, \sigma^2)$.

Assuming then a prior distribution $\pi(\sigma^2)$ for the unknown variance, and the uncertain model and the discrepancy as priorly independent, the joint prior distribution can be drawn out:

$$\pi(\boldsymbol{x}) = \pi(\boldsymbol{x}_\mathcal{M})\, \pi(\sigma^2) \tag{2.15}$$

With this in mind a particular measurement point $\boldsymbol{y}_i \in \mathcal{Y}$ is a realization of a Gaussian distribution with mean value $\mathcal{M}(\boldsymbol{x})$ and variance σ^2. In this way the likelihood function reads:

$$\mathcal{L}(\boldsymbol{x}_\mathcal{M}, \sigma^2; \mathcal{Y}) = \mathcal{N}(\boldsymbol{y}|\mathcal{M}(\boldsymbol{x}), \sigma^2) \tag{2.16}$$

specifically:

$$\mathcal{L}(\boldsymbol{x}_\mathcal{M}, \sigma^2; \mathcal{Y}) = \prod_{i=1}^{N} \frac{1}{\sqrt{(2\pi\sigma^2)^N}}\; e^{-\frac{1}{2\sigma^2}\left(\boldsymbol{y}_i - \mathcal{M}(\boldsymbol{x}_\mathcal{M})\right)^\intercal \left(\boldsymbol{y}_i - \mathcal{M}(\boldsymbol{x}_\mathcal{M})\right)} \tag{2.17}$$

In this way the posterior distribution can be computed as:

$$\pi(\boldsymbol{x}_\mathcal{M}, \sigma^2|\mathcal{Y}) = \frac{1}{Z}\, \pi(\boldsymbol{x}_\mathcal{M})\pi(\sigma^2)\, \mathcal{L}(\boldsymbol{x}_\mathcal{M}, \sigma^2; \mathcal{Y}) \tag{2.18}$$

that summarizes the updated information about the unknowns $\boldsymbol{x} = (\boldsymbol{x}_\mathcal{M}, \sigma^2)$ after conditioning on the observation $\mathcal{Y}$. Finally the marginals of individual forward model inputs $\pi(\boldsymbol{x}_{\mathcal{M},i}|\mathcal{Y})$ and of the residual variance $\pi(\sigma^2|\mathcal{Y})$ can be elicited.

The marginal of a specific parameter x_i (with $i \in \{1, ..., M\}$) can be computed by integration over the other components:

$$\pi(x_i|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{x}_{\sim i}}} \pi(\boldsymbol{x}|\mathcal{Y})\ d\boldsymbol{x}_{\sim i} \tag{2.19}$$

where $\boldsymbol{x}_{\sim i}$ refers to the parameter vector $\boldsymbol{x}$ excluding the i-th parameter x_i.

2.3.1 Inverse solution

The posterior distribution (3.26) computed in SECTION 2.3 is characterized through its first statistical moments. The *posterior mean vector* can be considered as a *point estimate* of the unknown parameters:

$$\mathbb{E}\left[\boldsymbol{X}|\mathcal{Y}\right] = \int_{\mathcal{D}_{\boldsymbol{x}}} \boldsymbol{x}\ \pi(\boldsymbol{x}|\mathcal{Y})\ d\boldsymbol{x} \tag{2.20}$$

whereas the uncertainty estimate can be quantified through the *posterior covariance matrix*:

$$\mathrm{Cov}\left[\boldsymbol{x}|\mathcal{Y}\right] = \int_{\mathcal{D}_{\boldsymbol{x}}} \big(\boldsymbol{x} - \mathbb{E}\left[\boldsymbol{X}|\mathcal{Y}\right]\big)\big(\boldsymbol{x} - \mathbb{E}\left[\boldsymbol{X}|\mathcal{Y}\right]\big)^{\intercal}\ \pi(\boldsymbol{x}|\mathcal{Y})\ d\boldsymbol{x} \tag{2.21}$$

More generally, $\pi(\boldsymbol{x}|\mathcal{Y})$ can be seen also as an intermediate quantity used for computing the conditional expectation of a certain QoI (e.g. analytical functions, secondary models), let say $h : \mathcal{D}_{\boldsymbol{X}} \to \mathbb{R}$. The conditional expectation of $h(\boldsymbol{X})$ under $\pi(\boldsymbol{x}|\mathcal{Y})$ is defined by the integral:

$$\mathbb{E}\left[\boldsymbol{X}|\mathcal{Y}\right] = \int_{\mathcal{D}_{\boldsymbol{x}}} h(\boldsymbol{x})\pi(\boldsymbol{x}|\mathcal{Y})\ d\boldsymbol{x} \tag{2.22}$$

2.3.2 Model predictions

Predictive distributions (2.10) and (2.11) (SECTION 2.2.1) can be now computed, in lights of SECTION 2.3, to assess the predictive capabilities of the computational model:

$$\pi'_{pred}(\boldsymbol{y}) = \int_{\mathcal{D}_{\boldsymbol{x}}} \mathcal{L}(\boldsymbol{x};\boldsymbol{y})\,\pi(\boldsymbol{x})\,d\boldsymbol{x} = \int_{\mathcal{D}_{\boldsymbol{x}}} \mathcal{L}(\boldsymbol{x}_{\mathcal{M}}, \sigma^2;\mathcal{Y})\,\pi(\boldsymbol{x}_{\mathcal{M}})\,\pi(\sigma^2)\,d\boldsymbol{x} \tag{2.23}$$

$$\pi''_{pred}(\boldsymbol{y}^{\star}|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{x}}} \mathcal{L}(\boldsymbol{x};\boldsymbol{y})\,\pi(\boldsymbol{x}|\mathcal{Y})\,d\boldsymbol{x} = \int_{\mathcal{D}_{\boldsymbol{x}}} \mathcal{L}(\boldsymbol{x}_{\mathcal{M}}, \sigma^2;\mathcal{Y})\,\pi(\boldsymbol{x}_{\mathcal{M}}, \sigma^2|\mathcal{Y})\,d\boldsymbol{x} \tag{2.24}$$

The integral of equation (2.23) is not in practice computed explicitly, indeed samples from this distribution can be obtained by sampling first $\boldsymbol{x}$ according to its prior (3.24), then sampling $\boldsymbol{Y}$ conditioned on model $\mathcal{M}(\boldsymbol{x})$:

$$\boldsymbol{Y} \sim \mathcal{M}(\boldsymbol{y}|\mathcal{M}(\boldsymbol{x}), \sigma^2) \tag{2.25}$$

in other words, this corresponds to sampling a realization of the discrepancy ε according to (3.23) and adding it to $\mathcal{M}(\boldsymbol{x})$. The same concept can be applied to the posterior predictive: a sample from the posterior predictive distribution is obtained by sampling first $\boldsymbol{x}$ according to the posterior (3.26), then evaluating $\mathcal{M}(\boldsymbol{x})$ and adding an independently sampled discrepancy term ε by drawing it from (3.23).

2.4 Markov Chain Monte Carlo simulations

The posterior distribution (3.26) computed in SECTION 2.3, here rewritten for the sake of simplicity:

$$\pi(\boldsymbol{x}_{\mathcal{M}}, \sigma^2|\mathcal{Y}) = \frac{1}{Z}\,\pi(\boldsymbol{x}_{\mathcal{M}})\pi(\sigma^2)\,\mathcal{L}(\boldsymbol{x}_{\mathcal{M}}, \sigma^2; \mathcal{Y}) \tag{2.26}$$

do not have a closed solution. The reason why lies on evaluating the normalizing factor Z integral:

$$Z = \int_{\mathcal{D}_{\boldsymbol{X}}} \pi(\boldsymbol{x}_{\mathcal{M}})\pi(\sigma^2)\,\mathcal{L}(\boldsymbol{x}_{\mathcal{M}}, \sigma^2; \mathcal{Y})\,d\boldsymbol{x} \tag{2.27}$$

A common approach to get around this obstacle relies on *Monte Carlo* (MC) stochastic integration technique. The former usually has to resort to *Markov chain Monte Carlo* (MCMC) simulations, a powerful tool based on repeated simulations of the model $\mathcal{M}$, which allows the representation of an integral as an expectation. In this way, the posterior is explored by realizing appropriate Markov chains over the prior support. The obtained sample can be used then to empirically estimate output statistics by drawing samples from the posterior [60].

The idea behind MCMC algorithms is to set up over the support $\mathcal{D}_{\boldsymbol{X}}$ of the prior, a Markov chain $(\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, ...)$ with an invariant distribution that equals the posterior distribution of interest [67]. Specifically, a Markov chain can be defined by its *transition kernel* $\mathcal{K}$ (i.e. the *density of the transition probability*) from step t to the successive one $t+1$, namely $\mathcal{K}(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)})$. Hence, if $\mathcal{K}$ fulfills the so-called *"detailed balance condition"*:

$$\pi(\boldsymbol{x}^{(t)}|\mathcal{Y})\mathcal{K}(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)}) = \pi(\boldsymbol{x}^{(t+1)}|\mathcal{Y})\mathcal{K}(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t+1)}) \tag{2.28}$$

the posterior distribution π is the invariant distribution of the Markov chain and the *reversibility* of the chain is guaranteed (i.e. the probability to move from $\boldsymbol{x}^{(t)}$ to $\boldsymbol{x}^{(t+1)}$ is equal to the probability to move from $\boldsymbol{x}^{(t+1)}$ to $\boldsymbol{x}^{(t)}$) [60].

The posterior distribution, finally, is obtained by integrating (2.28) over $d\boldsymbol{x}^{(t)}$:

$$\pi(\boldsymbol{x}^{(t+1)}|\mathcal{Y}) = \int_{\mathcal{D}_{\boldsymbol{x}}} \pi(\boldsymbol{x}^{(t)}|\mathcal{Y})\mathcal{K}(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)})\,d\boldsymbol{x}^{(t)} \tag{2.29}$$

A Markov chain constructed this way can be used to approximate the expectation of a certain QoI $h(\boldsymbol{X})$ as the iteration average of the $T+1$ generated sample points $\boldsymbol{x}^{(t)}$ [67]:

$$\mathbb{E}\left[h(\boldsymbol{X})|\mathcal{Y}\right] = \int_{\mathcal{D}_{\boldsymbol{X}}} h(\boldsymbol{x})\,\pi(\boldsymbol{x}|\mathcal{Y})\,d\boldsymbol{x} \approx \frac{1}{T}\sum_{t=1}^{T} h(\boldsymbol{x}^{(t)}) \tag{2.30}$$

2.4.1 Affine invariant ensemble algorithm (AIES)

The construction of a Markov kernel $\mathcal{K}$ that satisfies (2.28) is based on an easy principle: sampling candidates from a proposal distribution and then accept/reject them through some decision criteria. Almost all MCMC sampling methods (based on the original *Metropolis-Hastings* (MH) algorithm), share a common problem-feature: to reach high performance, these algorithms typically need a considerable amount of tuning of their parameters. Goodman and Weare [29] proposed a MCMC method, known as the *affine invariant ensemble algorithm* (AIES), that alleviates this problem and whose performance is unaffected by affine transformations of the the target distribution [67].

The AIES algorithm runs an ensemble of C *walkers*, i.e. an ensemble of C Markov chains $\{\mathcal{X}_1, ..., \mathcal{X}_C\}$, simultaneously. The locations $\boldsymbol{x}_i$ of the Markov chain are updated walker by walker. In each update a random conjugate walker, let say $\boldsymbol{x}_j^{(t)}$ (with $j \neq i$), is selected form the set of walkers. By generating proposals of a new candidate according to the *stretch move*:

$$\boldsymbol{x}_i^{(*)} = \boldsymbol{x}_i^{(t)} + Z(\boldsymbol{x}_j^{(t)} - \boldsymbol{x}_i^{(t)}) \tag{2.31}$$

the affine invariance property is achieved. Z is computed as:

$$Z \sim p(z) = \begin{cases} \frac{1}{\sqrt{z}\left(2\sqrt{a} - \frac{2}{\sqrt{a}}\right)} & z \in \left[\frac{1}{a}, a\right], \\ 0 & z \notin \left[\frac{1}{a}, a\right]. \end{cases} \tag{2.32}$$

This implies sampling from $p(z)$ defined by the tuning parameter $a > 1$. An evident advantage of this algorithm is the presence of only a single scalar tuning parameter which is often set to $a = 2$ in many practical applications [29].

Then the candidate $x^{(*)}$ is accepted as the new location of the i-th walker with probability:

$$\alpha\left(\boldsymbol{x}_i^{(*)}, \boldsymbol{x}_i^{(t)}, z\right) = \min\left\{1, z^{M-1}\frac{\pi(\boldsymbol{x}_i^{(*)}|\mathcal{Y})}{\pi(\boldsymbol{x}_i^{(t)}|\mathcal{Y})}\right\} \tag{2.33}$$

At the end, repeating this for all C chains, the resulting chains fulfill the *balance condition* (2.28) and the sample generated by the *walkers* can be combined to estimate expectations (2.30) under the posterior distribution, i.e. the algorithm produces chains of sample points that will follow the posterior distribution.

2.4.2 Convergence of MCMC simulations

MCMC simulations are theoretically simple, robust, computationally effortless and well-suited for parallel processing [40]. However, the low convergence rate (proportional to $1/\sqrt{N}$ where N is the number of numerical simulations [40]) and the the lack of a convergence criterion represent a critical issue for these algorithms. Indeed, a large number of forward model runs are typically required. This may be prohibitive, especially in earthquake engineering, where even a single model simulation can be computationally expensive. Owing to this, one is forced to make decisions, in practice, based on a finite number of sample points. Thus conducting a proper MCMC simulation may result not an easy task or even not feasible. For this reason in SECTION 2.5, surrogate modelling for the representation of the computational model response at low computational cost is introduced.

2.5 Surrogate modeling

Monte Carlo simulations are powerful tools for propagating uncertainties. Nevertheless, the low convergence rate, as seen in SECTION 2.4.2, constitutes a crucial issue that needs to be managed. In earthquake engineering each nonlinear analysis of the systems (i.e. each model evaluation) under earthquake excitation, is computationally quite expensive. Notwithstanding the current computing capacity, carrying out a proper MCS may lead, for practical applications, to prohibitive time-consuming issues. With this in mind, *surrogate modelling*, a.k.a. *metamodelling*, allows one to construct approximate models that emulate the behaviour of the whole simulation at low computational expenses. Important classes of surrogate models are based on PCE (*Polynomial Chaos Expansions*) (SECTION 2.5.1) and on *Kriging* (*Gaussian Process models*) (SECTION 2.5.2). Applications in the field of earthquake engineering have been introduced recently, some examples can be found in literature in [31], [41], [62], [40].

Formally, a *metamodel* $\tilde{\mathcal{M}}$ is an approximation of the original computational model:

$$\tilde{\mathcal{M}}(\boldsymbol{X}) \approx \mathcal{M}(\boldsymbol{X}) \tag{2.34}$$

Specifically, the process to build a surrogate model consists of three stages [40]:

1. creation of an *experimental design* (ED), i.e. a set of N random realizations of the model parameters $\boldsymbol{X} = \{\boldsymbol{x}^{(i)}, i = 1, ..., N\}$ with their associated response values obtained by evaluating the computational model $\mathcal{M}$ onto $\boldsymbol{X}$, for fitting the metamodel from numerical simulations data;

2. the *training process*, i.e. the identification of the the surrogate model parameters by means of learning algorithms respectful of specific criteria (e.g. minimizing error estimators);

3. the *validation process*, i.e. the validation of the surrogate model by evaluating its accuracy in terms of predicting the random responses using a new independent validation set of data different from the one used for the training

process [63]. It should be stressed that the surrogate model can be used only once the validation has been satisfied.

Following, SECTION 2.5.1 and SECTION 2.5.2 constitute a brief theoretic summary of the basic principles behind PCE metamodelling and Kriging metamodelling respectively. In the view of this thesis work, the final focus is in applying surrogate modelling to represent the time-dependent stochastic responses of structures under earthquake excitations. Specifically, in SECTION 2.5.3 is called attention on the well-known difficulties that PCE and Kriging exhibit when applied to stochastic dynamical models [68]. To overcome this challenging issue, the approaches proposed by C. Mai et al. [42] and by Worden et al. [70] have been used. Both of them are based on the basic idea of combining NARX models with the surrogates models (SECTIONS 2.5.5, 2.5.6).

2.5.1 Polynomial chaos expansions

The idea is to decompose the forward computational model into polynomial terms that are orthogonal with respect to a weight function [40]. Hence:

$$\mathcal{M}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha}} y_{\alpha} \boldsymbol{\psi}_{\boldsymbol{\alpha}}(\boldsymbol{X}) \tag{2.35}$$

where y_{α} are coefficient to be computed, $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_M)$ are multi-indices that identify the components of the multivariate polynomials orthonormal basis functions $\boldsymbol{\psi}_{\boldsymbol{\alpha}}(\boldsymbol{X}) = \prod_{i=1}^{M} \psi_{\alpha_i}^{i}(X_i)$. However, an infinite series expansion (2.35) cannot be treated practically. For that, a truncated form with a finite number of terms of the PCE has to be considered:

$$\mathcal{M}(\boldsymbol{X}) = \mathcal{M}^{PC}(\boldsymbol{X}) + \varepsilon = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\alpha} \boldsymbol{\psi}_{\boldsymbol{\alpha}}(\boldsymbol{X}) + \varepsilon \tag{2.36}$$

where $\mathcal{A}$ is the truncation set of selected multi-indices of multivariate polynomials and ε is the truncation induced residual error.

Following [40], the relevant polynomials to include in the truncation are those in the subset defined by two schemes:

SCHEME 1 - hyperbolic truncation scheme:

$$\mathcal{A} \equiv \mathcal{A}_q^{M,p} = \left\{ \boldsymbol{\alpha} \in \mathbb{N}^M : \|\boldsymbol{\alpha}\|_q = \left(\sum_{i=1}^{M} \alpha_i^q \right)^{1/q} \leq p \right\} \tag{2.37}$$

SCHEME 2 - low-rank truncation scheme:

$$\mathcal{A} \equiv \mathcal{A}_r^{M,p} = \left\{ \boldsymbol{\alpha} \in \mathbb{N}^M : \|\boldsymbol{\alpha}\|_0 = \sum_{i=1}^{M} \mathbb{1}_{\alpha_i > 0} \leq r, \|\boldsymbol{\alpha}\|_1 = \sum_{i=1}^{M} \alpha_i \leq p \right\} \tag{2.38}$$

where the parameter $q \in (0; 1]$ governs the hyperbol, p is the maximum degree of the multivariate polynomials, whereas r is the rank coefficient suggested by C. Mai

et al. [42] as a small integer value, e.g. $r = 2, 3, 4$.

The coefficients y_α of the PCE are computed from a set of model evaluations, the experimental design (ED), setting up a least-squares minimization problem, i.e.:

$$\hat{y}_\alpha = \arg\min_{y_\alpha} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \sum_\alpha y_\alpha\,\psi_\alpha(\boldsymbol{x}^{(\boldsymbol{i})})\right)^2 \tag{2.39}$$

Leave-one-out cross-validation error (LOO)

The *leave-one-out* (LOO) *cross-validation error* ε_{LOO} is an error estimator of the PCE metamodel based on cross-validation. The latter consists in partitioning the Experimental Design (ED) in two complementary subsets so that one is used to train the model, while the other is used to validate its prediction.

Practically only one sample constitutes the validation set (for this reason the term LOO). Hence, leaving one point $\boldsymbol{x}^{(i)}$ out, a PCE model $\mathcal{M}^{PC\setminus i}(\cdot)$ from the remaining ED $\boldsymbol{X}\setminus\boldsymbol{x}^{(i)} = \{\boldsymbol{x}^{(i)}, i = 1, ..., N\}$ can be built.

The LOO error is defined as:

$$\mathrm{Err}_{LOO} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \mathcal{M}^{PC\setminus i}(\boldsymbol{x}^{(i)})\right)^2 \tag{2.40}$$

Whereas the relative leave-one-out cross-validation LOO error is obtained by:

$$\varepsilon_{LOO} = \frac{\mathrm{Err}_{LOO}}{Var[\mathcal{Y}]} = \frac{\sum_{i=1}^{N}\left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \mathcal{M}^{PC\setminus i}(\boldsymbol{x}^{(i)})\right)^2}{\sum_{i=1}^{N}\left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \hat{\mu}_\mathcal{Y}\right)^2} \tag{2.41}$$

2.5.2 Kriging

Kriging (a.k.a. Gaussian Process (GP) modeling) is a stochastic interpolation method in which the computational model $\mathcal{M}(\boldsymbol{X})$ is assumed being a realization of a Gaussian process:

$$\mathcal{M}(\boldsymbol{X})^K = \beta^\intercal \boldsymbol{f}(\boldsymbol{x}) + \sigma^2 Z(\boldsymbol{x}, \omega) \tag{2.42}$$

where $\beta^\intercal \boldsymbol{f}(\boldsymbol{x})$ is the GP *trend* (i.e. its mean value), σ^2 is the GP constant variance, whereas $Z(\boldsymbol{X}, \omega)$ is the GP zero-mean unit-variance. In the definition (2.42) the probability space is represented by $\omega = \omega\big(R(\boldsymbol{\theta})\big)$ which depends on the *correlation function* R (a function that describes the correlation between two sample points $\boldsymbol{x}, \boldsymbol{x}'$) and its *hyperparameters* θ [67].

The process to build a Kriging metamodel consists of 5 stages:

1. selection of a basis $\boldsymbol{f}(\boldsymbol{x})$ for the Kriging trend (e.g. polynomials, arbitrary specified function);
2. selection of a correlation function (or *kernel*) $R = R(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{\theta})$ that describes, depending on the distance between the input points, how similar observations and new points are (e.g. linear, exponential, Gaussian correlation families);
3. computing $\boldsymbol{\theta}$ hyperparameters setting up an optimization problem on the selected estimation method;
4. computing the Gaussian Process variance σ^2;
5. computing the remaining Kriging unknown parameters (e.g. β coefficients).

Cross-validation estimation

The cross-validation (CV) estimation method (a.k.a. K-fold-cross-validation) is based on the partitioning of the whole set of observations (i.e., the observations related to the experimental design (ED)) $\mathcal{D} = \big\{(\boldsymbol{x}^{(i)}, y^{(i)}), i = 1, ..., N\big\}$, into K mutually exclusive subsets $\mathcal{D}_k$:

$$\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}_k; \quad \text{and} \quad \mathcal{D}_i \cap \mathcal{D}_i = \emptyset, \ \ \forall (i,j) \in \{1, ..., K\} \tag{2.43}$$

Once the model has been estimated using all the subsets of the ED except for the k-th one, the latter is obtained by the prediction of the model. Specifically, when K is chosen equal to N, i.e. when the number of the subsets is equal to the number of observations and the number of elements in $\mathcal{D}_k$ is one, the estimation is called *leave-one-out* (LOO) cross validation.

The cross-validation error of the k-th set reads:

$$\varepsilon_{CV,k} = \sum_{(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathcal{D}_k} \Big(y^{(i)} - \mu_{\hat{Y} \setminus \mathcal{D}_k}(\boldsymbol{x}^{(i)}; \beta, \sigma^2 \boldsymbol{\theta}, \sigma_n^2) \Big)^2 \tag{2.44}$$

while the overall cross-validation error is:

$$\varepsilon_{CV}\big(\beta, \sigma^2, \boldsymbol{\theta}, \sigma_n^2; \mathcal{Y}\big) = \frac{1}{N} \sum_{k=1}^{K} \varepsilon_{CV,k} \tag{2.45}$$

At this point, the idea is to set up an optimization problem that minimizes the cross-validation error to find the set of Kriging parameters:

$$\hat{\boldsymbol{\theta}}, \hat{\sigma}^2 = \arg\min_{\boldsymbol{\theta}, \sigma^2} \ \varepsilon_{CV}\big(\boldsymbol{\theta}, \sigma^2; \mathcal{Y}\big) \tag{2.46}$$

A posteriori estimation and LOO cross-validation error

Once Kriging metamodel is built, its predictive accuracy can be assessed by the relative leave-one-out LOO cross-validation error. As seen in SECTION 2.5.1, the Experimental Design (ED) is partitioned into two complementary subsets: one used to train the model, one used to validate its prediction. Hence:

$$\varepsilon_{LOO} = \frac{\sum_{i=1}^{N} \left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \mathcal{M}^{K\setminus i}(\boldsymbol{x}^{(i)}) \right)^2}{\sum_{i=1}^{N} \left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \hat{\mu}_{\mathcal{Y}} \right)^2} \tag{2.47}$$

2.5.3 Surrogate modelling for stochastic dynamical systems

As earlier mentioned, the focus of this thesis work relies on stochastic dynamical systems. The random output response of those systems is also a time-dependent quantity:

$$\boldsymbol{y}(t, \boldsymbol{\xi}) = \mathcal{M}(t, \boldsymbol{\xi}) \tag{2.48}$$

where t is the time variable and $\boldsymbol{\xi} = \{\xi_1, ..., \xi_M\}$ is the random vector of the M uncertain parameters of the system obtained by realizations of a random variable $\boldsymbol{\Xi} \sim \pi(\boldsymbol{\xi})$.

In this context, it has been observed by C. Mai et al. [42] that the accuracy of the polynomial chaos representation of the response $\boldsymbol{y}(t, \boldsymbol{\xi})$ (time-frozen PCEs) degenerates quickly in time. Thus leads to inaccurate metamodels. To maintain the accuracy, higher-degree polynomials or different, larger and more complicated types of basis functions shall be considered in the updating of the expansion, that means to assign to PCEs the role of catching the dynamics of the system representing the increasing non-linearity of the response with respect to the uncertain parameters [40]. However, PCEs just fail this task due to the simple fact that are a tool deigned for propagating uncertainties and not to catch the dynamics. As consequence, is more effective to separate the problem, i.e. using PCEs to represent the uncertainties-related part of the problem, while using a different tool to capture the dynamic part. For this specific purpose, Spiridonakos and E. Chatzi [65] proposed a numerical approach that combines PCEs and non-linear autoregressive with exogenous input (NARX) models. Specifically, NARX models are used to mimic the dynamics of the system, while PCEs are used to deal with the uncertainties. Later, an evolution of such approach based on LARS algorithm was proposed by C. Mai et al. [42]. The latter is presented in the following sections, and then applied in CHAPTER 3 for numerical benchmarks, and in CHAPTER 5 for a real case study.

2.5.4 Nonlinear Autoregressive with eXogenous input model

Let consider a generic dynamical system subjected to a time-dependent input excitation $x(t)$. Let $y(t)$ be its time-history response, and let $\mathcal{M}$ be a mathematical

model that relates the input-output signals. So that:

$$y(t) = \mathcal{M}\big(x(t)\big) \tag{2.49}$$

The essence of the nonlinear autoregressive with exogenous input (NARX) models is that past outputs are included in the expansion of the model $\mathcal{M}$. More specifically, NARX models allow to build the latter using the observed data of the input and output signals. So that, the output quantity $y(t)$ can be represented, at a certain instant of time t, as an expansion of its past values and values of the input excitation at the current or previous instants [8]:

$$y(t) = \mathcal{M}\big(x(t)\big) = \mathcal{F}\big(z(t)\big) + \varepsilon(t) \tag{2.50}$$

where $\mathcal{F}(\cdot)$ is the NARX model, namely some nonlinear function to be identified, $\varepsilon(t) \sim \mathcal{N}\big(0, \sigma_\varepsilon^2(t)\big)$ is the residual error of the NARX model, and $z(t)$ is the vector of current and past values, i.e.:

$$\boldsymbol{z}(t) = \big\{x(t), ..., x(t-n_x),\ y(t-1), ..., y(t-n_y)\big\}^\intercal \tag{2.51}$$

where n_x, n_y stand for the maximum input and output time lags, respectively. Generally, just a restricted number of time lags ahead of the present time instant t is sufficient to catch the dynamics, since the cause-consequence tends to fade away as time evolves [8].

NARX system identification can be conceptualized into three major steps:

1. *basis function selection*, i.e. choosing a basis function to construct the mapping $\mathcal{F}(\cdot)$ (e.g. polynomial, wavelet, neural networks);
2. *structure detection*, i.e. determining relevant NARX terms to include in the model;
3. *parameter estimation*, i.e. computing the model coefficients by least square minimization.

To keep things simple, and following a popular trend noticed in the literature, polynomial basis function have been used in this work to construct the mapping $\mathcal{F}(\cdot)$. So that:

$$y(t) = \sum_{i=1}^{n_g} \vartheta_i g_i\big(\boldsymbol{z}(t)\big) + \varepsilon(t) \tag{2.52}$$

where n_g is the number of the polynomial NARX model terms $g_i\big(\boldsymbol{z}(t)\big)$, and ϑ_i are coefficients to estimate. It should be stressed that structure detection is crucial, especially for systems involving nonlinearities. Indeed, as noticed in [8], including spurious terms in the NARX may lead to numerical and computational issues. That means that polynomial expansions can be ill-conditioned as a result of the explosion of the terms involved. The aforementioned problem can be avoided exclusively if only significant model terms are included in the model. To this aim, C. Mai et al. [42] suggest the use of orthogonal least squares algorithm and its derivatives for

structure selection. Following their work, a structure detection approach based on *least angle regressions* (LARS) [25] is herein briefly mentioned and then used.

Least angle regression (LAR)

The least angle regression (LAR) is an efficient algorithm aimed at selecting those predictors (in this context the basis polynomials g_i) among a larger set of candidates, that have the greatest impact on the NARX model response.

Following [9], the original LAR algorithm run is itemized below:

- initialize the coefficients $\vartheta_0, ..., \vartheta_{n_g} = 0$;
- initialize the residual equal to the vector of observations $\mathcal{Y}$;
- detect the g_i most correlated with the initial residual;
- move ϑ_j from 0 toward the least-square coefficient of the current residual on g_j, until some other predictor g_k has as much correlation with the current residual as does g_j;
- move jointly $\{\vartheta_j, \vartheta_k\}^\intercal$ in the direction defined by their joint least-square coefficient of the current residual on $\{g_j, g_k\}$, until some other predictor g_l has as much correlation with the current residual;
- keep on the iterations until $m \equiv \min(n_g, N-1)$ predictors have been entered.

2.5.5 PC-NARX

PC-NARX models combines PCEs and NARX models to overcame the aforementioned issues related to stochastic dynamic systems (SECTION 2.5.3). The basic idea is just to split the uncertainties-related part of the problem from the dynamic one. This can be achieved let the PCEs to propagate uncertainty, while the NARX model to capture the time-dependent nonlinearity. In this way, the time-dependent output quantity $\boldsymbol{y}(t, \boldsymbol{\xi}) = \mathcal{M}(t, \boldsymbol{\xi})$ can be expressed by means of a NARX model whose coefficients are only function of the randomness of the input parameters, whereas the basis are only time-dependent quantity, i.e.:

$$\boldsymbol{y}(t, \boldsymbol{\xi}) = \sum_{i=1}^{n_g} \vartheta_i(\boldsymbol{\xi}) g_i\big(\boldsymbol{z}(t)\big) + \varepsilon_g(t, \boldsymbol{\xi}) \tag{2.53}$$

where n_g is the number of the NARX model terms $g_i\big(\boldsymbol{z}(t)\big)$, $z(t)$ is the vector of current and past values, n_x, n_y are the maximum input and output time lags, $\varepsilon_g(t) \sim \mathcal{N}\big(0, \sigma_\varepsilon^2(t)\big)$ is the residual error. $\vartheta_i(\boldsymbol{\xi})$, that are now functions of the uncertain input parameters $\boldsymbol{\xi}$, are represented by means of PCEs:

$$\vartheta_i(\boldsymbol{\xi}) = \sum_{j=1}^{n_\psi} \vartheta_{i,j} \psi_j(\boldsymbol{\xi}) + \varepsilon_i \tag{2.54}$$

where ε_i is the truncation error of the expansion. Finally, substituting equation (2.54) in (2.53) the PC-NARX model is obtained:

$$\boldsymbol{y}(t,\boldsymbol{\xi}) = \sum_{i=1}^{n_g}\sum_{j=1}^{n_\psi} \vartheta_{i,j}\psi_j(\boldsymbol{\xi})g_i\big(\boldsymbol{z}(t)\big) + \varepsilon(t,\boldsymbol{\xi}) \tag{2.55}$$

where $\varepsilon(t,\boldsymbol{\xi})$ is the total error due to NARX and PCE truncations.

Least angle regression-based approach

The process to build a PC-NARX model consists into two-phases approach [40] and is herein described and schematised in **Figure 2.1**:

PHASE 1: selection of the NARX model

- STEP 1.1: define the full NARX model by specifying type and properties of the basis functions, maximum input/output time lags n_x, n_y and/or the nonlinear behaviour involved in the system;
- STEP 1.2: selection of some candidate NARX models, i.e. NARX models containing only a subset of the full NARX terms using LARS algorithm. This is done for each k-experiment of the ED. Specifically, the recorded data $y(t,\boldsymbol{\xi})$ used for training the NARX model can be written as:

$$y(t,\boldsymbol{\xi}) = \hat{y}(t,\boldsymbol{\xi}_k) + \varepsilon(t,\boldsymbol{\xi}) \tag{2.56}$$

where $\hat{y}(t,\boldsymbol{\xi}_k)$ is the one-step-ahed NARX prediction (2.57) and $\varepsilon(t,\boldsymbol{\xi})$ is its residual.

$$\hat{y}(t,\boldsymbol{\xi}_k) = \sum_{i=1}^{n_g} \vartheta_i(\boldsymbol{\xi}_k)g_i(\hat{\boldsymbol{z}}(t,\boldsymbol{\xi}_k)) \tag{2.57}$$

in which:

$$\hat{\boldsymbol{z}}(t,\boldsymbol{\xi}_k) = \big\{x(t,\boldsymbol{\xi}_k),...,x(t-n_x,\boldsymbol{\xi}_k),\ y(t-1,\boldsymbol{\xi}_k),...,y(t-n_y,\boldsymbol{\xi}_k)\big\}^\intercal \tag{2.58}$$

Let now denote $\boldsymbol{\phi}(t) = \{g_i(\hat{\boldsymbol{z}}(t,\boldsymbol{\xi}_k), i = 1,...,n_g)\}^\intercal$ and $\boldsymbol{\vartheta}(\boldsymbol{\xi}_k) = \{\vartheta_i(\boldsymbol{\xi}_k), i = 1,...,n_g\}^\intercal$, so that equation (2.56) becomes:

$$y(t,\boldsymbol{\xi}) = \boldsymbol{\phi}^\intercal(t)\,\boldsymbol{\vartheta}(\boldsymbol{\xi}_k) + \varepsilon(t,\boldsymbol{\xi}) \tag{2.59}$$

Assembling then all time instants in the k-th experiment, one obtains:

$$\begin{Bmatrix} y(1,\boldsymbol{\xi}_k) \\ \vdots \\ y(T,\boldsymbol{\xi}_k) \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{\phi}^\intercal(1) \\ \vdots \\ \boldsymbol{\phi}^\intercal(T) \end{Bmatrix} \boldsymbol{\vartheta}(\boldsymbol{\xi}_k) + \begin{Bmatrix} \varepsilon(1,\boldsymbol{\xi}_k) \\ \vdots \\ \varepsilon(T,\boldsymbol{\xi}_k) \end{Bmatrix} \tag{2.60}$$

or in matrix notations:

$$\boldsymbol{y}_k = \boldsymbol{\Phi}_k + \boldsymbol{\vartheta}(\boldsymbol{\xi_k}) + \boldsymbol{\varepsilon}_k \tag{2.61}$$

where $\boldsymbol{y}_k$ $[T \times 1]$ is the output time-series vector, $\boldsymbol{\Phi}_k$ $[T \times n_g]$ is the *information matrix* whose row contain the evaluations of NARX terms $\boldsymbol{\phi}(t)$, whereas $\boldsymbol{\varepsilon}_k$ is the residual vector. Equation (2.61) represents a linear regression problem, for which the relevant NARX terms among the NARX candidate ones $\boldsymbol{\phi}(t)$ can be selected using LARS algorithm;

- STEP 1.3: computing NARX coefficients by ordinary least-squares (OLS) (2.62) for each experiment of the ED and for each candidate NARX model:

$$\boldsymbol{\vartheta}(\boldsymbol{\xi}_k) = \arg\min_{\boldsymbol{\vartheta}}(\boldsymbol{\varepsilon}_k^\intercal \boldsymbol{\varepsilon}_k) = \left[\boldsymbol{\Phi}_k^\intercal \boldsymbol{\Phi}_k\right]^{-1} \boldsymbol{\Phi}_k^\intercal \boldsymbol{y}_k \tag{2.62}$$

Then, for each experiment, the *free-run reconstruction of the response* (2.63) can be obtained using only the excitation time series $x(t)$ and the response initial condition y_0.

$$\check{y}(t, \boldsymbol{\xi}_k) = \sum_{i=1}^{n_g} \vartheta_i(\boldsymbol{\xi}_k) g_i(\check{\boldsymbol{z}}(t, \boldsymbol{\xi}_k)) \tag{2.63}$$

in which:

$$\check{\boldsymbol{z}}(t, \boldsymbol{\xi}_k) = \left\{x(t, \boldsymbol{\xi}_k), ..., x(t - n_x, \boldsymbol{\xi}_k),\ \check{y}(t-1, \boldsymbol{\xi}_k), ..., \check{y}(t - n_y, \boldsymbol{\xi}_k)\right\}^\intercal \tag{2.64}$$

It should be stressed that the response (2.63) differs from the (2.57) one due to the fact that in the computation of $\check{\boldsymbol{z}}(t, \boldsymbol{\xi}_k)$ it is not more used the recorded data but its free-reconstruction.

For each experiment k, the relative error reads:

$$\varepsilon_k = \frac{\sum_{t=1}^{T} \left(y(t, \boldsymbol{\xi}_k) - \check{y}(t, \boldsymbol{\xi}_k)\right)^2}{\sum_{t=1}^{T} \left(y(t, \boldsymbol{\xi}_k) - \bar{y}(t, \boldsymbol{\xi}_k)\right)^2} \tag{2.65}$$

where $\bar{y}(t, \boldsymbol{\xi}_k)$ is the mean value of the response time series $y(t, \boldsymbol{\xi}_k)$.

- STEP 1.4: selection of the NARX model among the candidate NARX models with the smallest mean error over the entire ED:

$$\bar{\varepsilon} = \frac{1}{K} \sum_{k=1}^{K} \varepsilon_k \tag{2.66}$$

PHASE 2: using the obtained NARX coefficients $\boldsymbol{\vartheta}$ with the corresponding sample set $\boldsymbol{\xi}$ of the random input parameters to train the PC expansions.

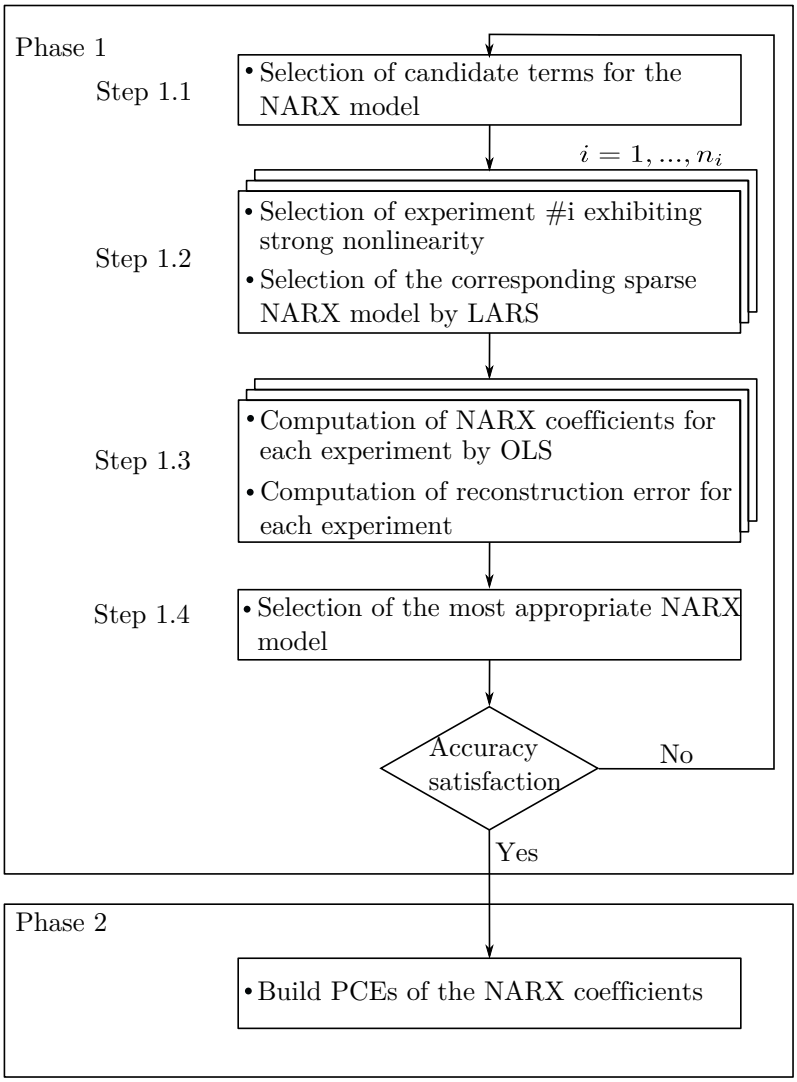


Figure 2.1: Computation of LARS-based PC-NARX model, [40]

PC-NARX prediction

The so built PC-NARX metamodel can be used for a generic realization $\boldsymbol{\xi}'$ of the input parameters to predict the response $\boldsymbol{y}(t, \boldsymbol{\xi}')$. The time history of the system can be recursively given the input excitation $x(t)$ and the initial conditions y_0:

$$\check{y}(t, \boldsymbol{\xi}') = \sum_{i=1}^{n_g} \sum_{j=1}^{n_\psi} \vartheta_{i,j} \psi_j(\boldsymbol{\xi}') g_i\big(\boldsymbol{z}(t, \boldsymbol{\xi}')\big) \tag{2.67}$$

in which:

$$\check{\boldsymbol{z}}(t, \boldsymbol{\xi}_k') = \big\{x(t, \boldsymbol{\xi}_k'), ..., x(t - n_x, \boldsymbol{\xi}_k'),\ \check{y}(t-1, \boldsymbol{\xi}_k'), ..., \check{y}(t - n_y, \boldsymbol{\xi}_k')\big\}^\intercal \tag{2.68}$$

PC-NARX validation

Once built the metamodel, a validation process has to be conducted to validate the PC-NARX surrogate. To this aim, a realization of lager size independent validation set of model input parameters realizations is used (e.g. $n_{val} = 1e^4$) to computing the actual model responses and its predictions by PC-NARX surrogate. The accuracy of the computed PC-NARX model is validated by comparing them in terms of the relative errors:

For i-th prediction, the relative error is:

$$\varepsilon_{val,i} = \frac{\sum_{t=1}^{T} \big(y(t,\boldsymbol{\xi}_i) - \check{y}(t,\boldsymbol{\xi}_i)\big)^2}{\sum_{t=1}^{T} \big(y(t,\boldsymbol{\xi}_i) - \bar{y}(t,\boldsymbol{\xi}_i)\big)^2} \tag{2.69}$$

While the overall mean value of the relative errors is:

$$\bar{\varepsilon}_{val} = \frac{1}{n_{val}} \sum_{i=1}^{n_{val}} \varepsilon_{val,i} \tag{2.70}$$

2.5.6 Kriging-NARX

Whole seen in SECTION 2.5.5 for the PC-NARX models, can be rearranged without loss of generality also for the Kriging metamodel, leading to Kriging-NARX models. Although this time the $\vartheta_i(\boldsymbol{\xi})$ parameters of equation (2.53) are assumed being a realization of a Gaussian process, i.e.:

$$\vartheta_i(\boldsymbol{\xi}) = \beta_i^{\intercal} \boldsymbol{f}_i(\boldsymbol{\xi}) + \sigma_i^2 Z(\boldsymbol{\xi}, \omega_i) \tag{2.71}$$

So that, substituting equation (2.71) in (2.53) the Kriging-NARX model can be obtained:

$$\boldsymbol{y}(t,\boldsymbol{\xi}) = \sum_{i=1}^{n_g} \Big(\beta_i^{\intercal} \boldsymbol{f}_i(\boldsymbol{\xi}) + \sigma_i^2 Z(\boldsymbol{\xi}, \omega_i)\Big)\ g_i\big(\boldsymbol{z}(t)\big) + \varepsilon(t,\boldsymbol{\xi}) \tag{2.72}$$

Chapter 3

Model validation and numerical benchmarks

In this chapter, the structural dynamic problem of hysteretic systems' response is formulated for a simple inelastic structure with a Single-Degree-of-Freedom (SDF). The Bouc-Wen-Baber-Noori (BWBN) model of hysteresis is presented briefly in Section 3.1 and then validated considering earthquake-induced ground motion as external excitation in Section 3.2. Afterwards the same model has been used to generate simulated experimental data (Section 3.3) from which calibration of the system parameters has been carried out taking into account first none model for the discrepancy (Section 3.4), secondly an additive Gaussian model discrepancy (Section 3.5).

3.1 Bouc-Wen-Baber-Noori model of hysteresis

The Bouc–Wen model of hysteresis (**Figure 3.1**) is widely used in structural engineering. The model was proposed by Bouc [11], and thereafter modified by Wen [69].

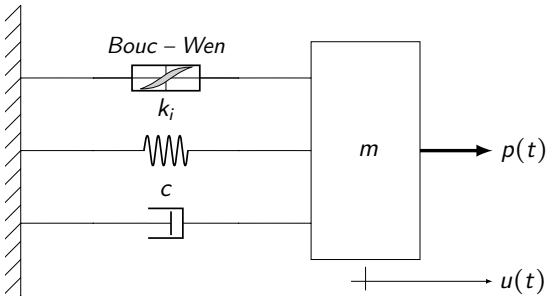


Figure 3.1: SDF Bouc-Wen model of hysteresis

The differential equation governing the motion of the structure is:

$$m\,\ddot{u}(t) + c\,\dot{u}(t) + f_r\big(u(t), z(t)\big) = p(t) \tag{3.1}$$

where m is the mass of the system, c is the viscous linear damping coefficient, $u(t)$ is the displacement, $f_r(u,z)$ is the restoring force, and $p(t)$ is the external excitation $\big($in the case of ground motion $p(t)$ is the effective earthquake force: $p_{eff}(t) = -m\,\ddot{u}_g(t)\big)$. The overdots represents the derivative with respect to time, hence $\ddot{u}(t)$ and $\dot{u}(t)$ represent the velocity and the acceleration respectively.

According to the Bouc–Wen model, the restoring force is given by the following expression:

$$f_r(u,z) = f_r^{el}(u,z) + f_r^{h}(u,z) = \alpha\, k_i\, u(t) + (1-\alpha)\, k_i\, z(t) \tag{3.2}$$

where $f_r^{el}(u,z)$ represents the elastic component whereas $f_r^{h}(u,z)$ is the hysteretic component (which depends on the past history of stresses and strains), k_i is the initial stiffness of the system, α is the ratio between the final stiffness and initial one ($\alpha = k_u/k_i$), $z(t)$ is the hysteretic displacement defined by the next differential equation:

$$\dot{z}(t) = A\,\dot{u}(t) - \Big[\beta\,|\dot{u}(t)||z(t)|^{N-1} z(t) + \gamma\,\dot{u}(t)\,|z(t)|^{N}\Big] \tag{3.3}$$

where the parameters A, β, γ and N control the hysteresis shape.

Later, in 1985, Baber and Noori [6] modified the model including additional parameters which enhance the capacity of the model to represent hysteretic shapes, the so called Bouc–Wen–Baber–Noori hysteresis model. Such changes reads:

$$\dot{z}(t) = h(z)\,\frac{A(\varepsilon)\,\dot{u}(t) - \nu(\varepsilon)\,\Big[\beta\,|\dot{u}(t)||z(t)|^{N-1} z(t) + \gamma\,\dot{u}(t)\,|z(t)|^{N}\Big]}{\eta(\varepsilon)} \tag{3.4}$$

where the parameters β, γ and N control the shape of the cycles, and the additional parameters $A(\varepsilon)$, $\eta(\varepsilon)$ and $\nu(\varepsilon)$ are degradation functions taking in account the stiffness and strength degradation. Those degradation functions are expressed in terms of the dissipated hysteretic energy $\varepsilon^h(t)$:

$$A(\varepsilon) = A_0(\varepsilon) - \delta_A\,\varepsilon^h(t) \tag{3.5a}$$

$$\nu(\varepsilon) = \nu_0(\varepsilon) + \delta_\nu\,\varepsilon^h(t) \tag{3.5b}$$

$$\eta(\varepsilon) = \eta_0(\varepsilon) + \delta_\eta\,\varepsilon^h(t) \tag{3.5c}$$

where the constant values of A_0, ν_0, η_0 are usually set to unity. Whereas the values δ_A, δ_ν, δ_η are constant terms which specify the amount of stiffness and strength degradation. Therefore a value of $\delta_A = \delta_\nu = \delta_\eta = 0$ represents no degradation, in contrast a value of $\delta_A = \delta_\nu = \delta_\eta \neq 0$ involves the degradation phenomenon.

The dissipated hysteretic energy $\varepsilon^h(t)$ is given by:

$$\varepsilon^h(t) = \int_{u(0)}^{u(t)} f_r^h(u,z)\,du \;=\; (1-\alpha)\,k_i \int_0^t z(\tau)\,\dot{u}(\tau)\,d\tau \tag{3.6}$$

whereas the total dissipated energy of the system $\varepsilon_{tot}(t)$ is the sum of the previous one and the elastic energy of the system $\varepsilon^{el}(t)$, i.e.:

$$\varepsilon^{el}(t) = \int_{u(0)}^{u(t)} f_r^{el}(u,z)\,du \;=\; \alpha\,k_i \int_0^t u(\tau)\,\dot{u}(\tau)\,d\tau \tag{3.7}$$

$$\varepsilon_{tot}(t) = \varepsilon^{el}(t) + \varepsilon^h(t) \tag{3.8}$$

The last function, $h(z)$, in equation (3.4), is the pinching function. The pinching state refers to a mild increase in the system's stiffness in points close to the origin in the hysteretic graph, followed by an abrupt increase in the stiffness. It is modeled by the following equation:

$$h(z) = 1 - \zeta_1(\varepsilon)\exp\left(-\frac{\left|z(t)\,sign\big(\dot{u}(t)\big) - q\,z_u\right|^2}{\zeta_2^2(\varepsilon)}\right) \tag{3.9}$$

where z_u is the last value of $z(t)$ which can be computed by:

$$z_u = \left[\frac{1}{\nu(\varepsilon)\,(\beta+\gamma)}\right]^{1/N} \tag{3.10}$$

and the values of $\zeta_1(\varepsilon)$ and $\zeta_2(\varepsilon)$ are given by:

$$\zeta_1(\varepsilon) = \zeta_0\big(1-\exp(-p\,\varepsilon(t))\big)\,; \qquad \zeta_2(\varepsilon) = \big(\psi + \delta_\psi\,\varepsilon(t)\big)\big(\lambda + \zeta_1(\varepsilon)\big) \tag{3.11}$$

where p is a constant which controls the initial change in the slope, ζ_0 is a measure of the total slip, ψ contributes to the pinching behaviour, δ_ψ is a specified constant which measures the dispersion rate of the pinching phenomenon, and λ controls the variation of the parameters $\zeta_1(\varepsilon)$ and $\zeta_2(\varepsilon)$. If the pinching function is chosen equal to unity, its effect is neglected.

3.2 Model validation

The Bouc-Wen-Baber-Noori model of hysteresis has been implemented in Matlab® [50], [59] with the purpose, in a first moment, to simulate experimental response data record of the system itself. The motion equation (3.1) presented in SECTION 3.1 has been normalized respect to the system mass:

$$\ddot{u}(t) + 2\,\xi\,\omega_n\,\dot{u}(t) + \alpha\,\omega_n^2\,u(t) + (1-\alpha)\,\omega_n^2\,z(t) = -\ddot{u}_g(t) \tag{3.12}$$

where ξ is the damping ratio, ω_n is the natural vibration frequency of the system, and $\ddot{u}_g(t)$ is the ground excitation.

Likewise equation (3.2) modifies in:

$$f_r^*(u,z) = f_r^{*el}(u,z) + f_r^{*h}(u,z) = \alpha\,\omega_n^2\,u(t) + (1-\alpha)\,\omega_n^2\,z(t) \tag{3.13}$$

where the star symbol $^{(*)}$, from now on, denotes that the specific quantity is normalized respect to the system mass. And finally:

$$\varepsilon^{*h}(t) = (1-\alpha)\,\omega_n^2 \int_0^t z(\tau)\,\dot{u}(\tau)\,d\tau \tag{3.14}$$

$$\varepsilon^{*el}(t) = \alpha\,\omega_n^2 \int_0^t u(\tau)\,\dot{u}(\tau)\,d\tau \tag{3.15}$$

$$\varepsilon_{tot}^*(t) = \varepsilon^{*el}(t) + \varepsilon^{*h}(t) \tag{3.16}$$

Equation (3.12) is solved using a state-space formulation:

$$\{\dot{x}(t)\} = \mathcal{F}\big(\{x(t)\}\big) \tag{3.17}$$

where the state-vector and its first derivative are defined as follow:

$$\{x\} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{Bmatrix} u \\ \dot{u} \\ z \\ \varepsilon^{*h} \\ \varepsilon^{*el} \end{Bmatrix}; \quad \{\dot{x}\} = \begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{Bmatrix} = \begin{Bmatrix} \dot{u} \\ \ddot{u} \\ \dot{z} \\ \dot{\varepsilon}^{*h} \\ \dot{\varepsilon}^{*el} \end{Bmatrix} \tag{3.18}$$

Substituting equation (3.18) in (3.17), the following system of five first order nonlinear ODEs is found and solved numerically with the explicit Runge-Kutta iterative method of integration using the Matlab solver `ode45` with relative error tolerance set to 1×10^{-3}:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\ddot{u}_g - 2\,\xi\,\omega_n\,x_2 - \alpha\,\omega_n^2\,x_1 - (1-\alpha)\,\omega_n^2\,x_3 \\ \dot{x}_3 = h(z)\;\eta^{-1}\left\{A\,x_2 - \nu\left[\beta\,|x_2||x_3|^{N-1}x_3 + \gamma\,x_2\,|x_3|^N\right]\right\} \\ \dot{x}_4 = (1-\alpha)\,\omega_n^2\,x_3\,x_2 \\ \dot{x}_5 = \alpha\,\omega_n^2\,x_1\,x_2 \end{cases} \tag{3.19}$$

In order to validate the code, the response of the BWBN model in linear field (obtained setting all parameters null save for $N = 100$) has been compared with the response of an equivalent linear SDF system whose response has been calculated using the Duhamel's integral:

$$u(t) = \int_0^t p(\tau)\,h(t-\tau)\,d\tau \tag{3.20}$$

where $h(t-\tau)$ is the impulse response function (IRF) defined as:

$$h(t-\tau) = \frac{1}{m\,\omega_d}\,\exp\big(-\xi\,\omega_n\,(t-\tau)\big)\,\sin\big(\omega_d\,(t-\tau)\big) \tag{3.21}$$

A record of the Montenegro earthquake (1979) was selected from the PEER Ground Motion Database (PEER, 2019), as seismic excitation (**Figure 3.2**). The Peak Ground Acceleration (PGA) measures 3.59 m/s^2.

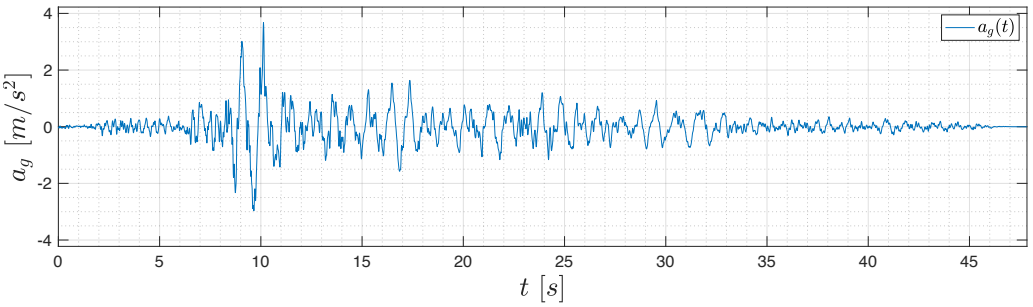


Figure 3.2: Montenegro (1979) ground motion record

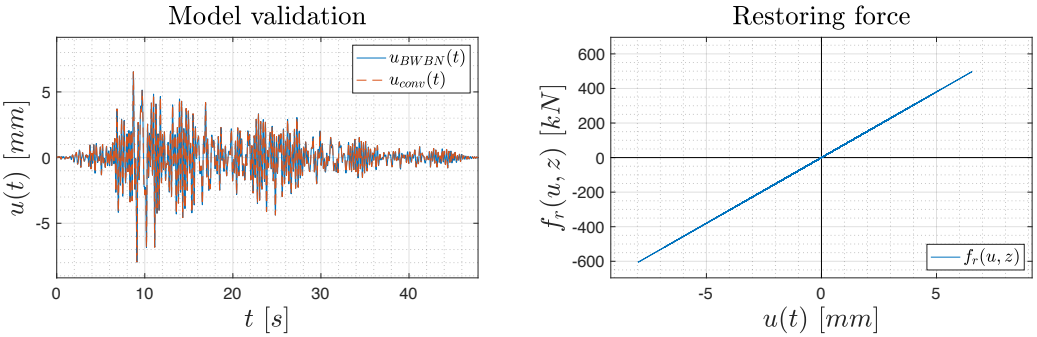


Figure 3.3: Model validation

In **Figure 3.3** the response of the two system and the restoring force of the linearized BWBN one are plotted. It is clear as the response history matches perfectly and as the restoring force has a linear-elastic pattern, so that the model validation can be hold to be satisfied.

3.3 Simulation of experimental record data

The same record of the Montenegro earthquake shown in SECTION 3.2 (**Figure 3.2**) has been used as input excitation to simulate the response of a nonlinear SDF BWBN system with a system mass of $m = 12000\,kg$, in order to get a simulated experimental record measurement. The initial stiffness k_i of the system has been calibrated with the objective of obtaining a plausible system frequency for a masonry structure of approximately $4\,Hz$. A damping ratio of $\xi = 3\%$ was adopted. Finally, the remaining BWBN parameters used to generate the simulated record are presented in **Table 3.1**.

Table 3.1: Bouc-Wen-Baber-Noori parameters

k_i $[kN/mm]$	β $[mm^{-1}]$	γ	N	δ_A	δ_ν	δ_η	$h(z)$
7.6	63×10^{-3}	63×10^{-3}	1	0	2.43	6.5	1

In **Figure 3.4** are plotted the results of the simulated data.

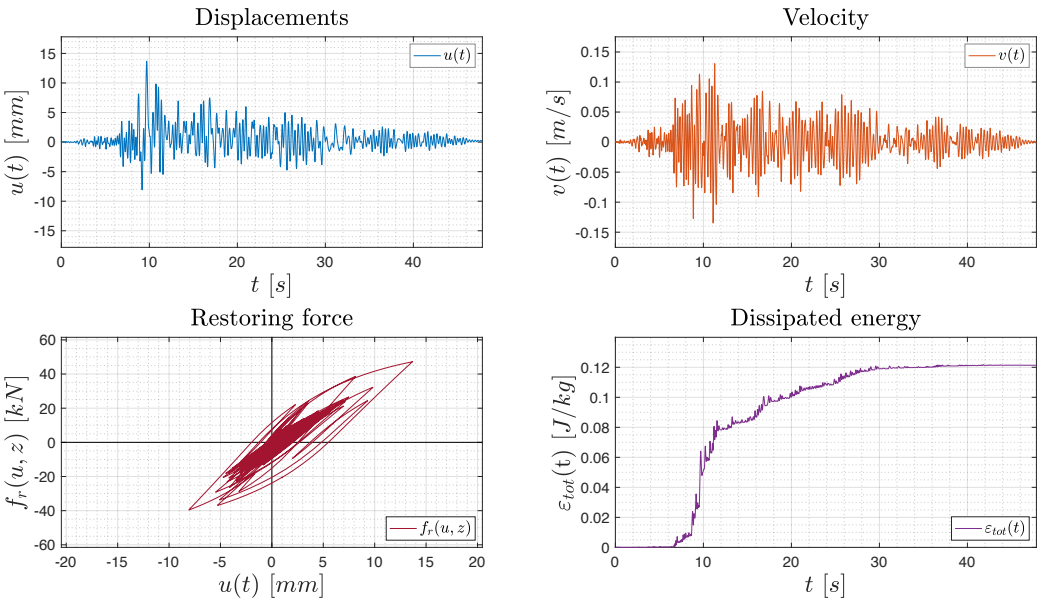


Figure 3.4: Simulated experimental data

These results constitutes the experimental observation used in SECTION 3.4, 3.5 for numerical benchmarks.

3.4 Calibration not accounting discrepancy

In this section optimization algorithms are used to deal with the inverse problem of model parameters calibration. These algorithms are often the tools used to handle problems characterized by a large number of parameters of very complex systems. Regardless the specific algorithm chosen, all these methods present two disadvantages at least: one related to the precision of the results, indeed different trials in the estimation of parameters may lead to different results in the values of the updated parameters [53], and one, more substantial, that is the lack of a model for the discrepancy. These factors lead to a not reliable value of the parameters identified.

Following, the calibration of the BWBN system is treated making use of simulated experimental data obtained in SECTION 3.3. None degrading or pinching effects are included in the model of hysteresis used for the calibration of the parameters (i.e $\delta_A = \delta_\nu = \delta_\eta = 0$ and $h(z) = 1$). This specific assumption is done to replicate what recursively happens in real experimental practice. Due to this choice

a discrepancy between measured data and model arises. For that, none model has been introduced.

The system of equations (3.19) has been solved searching the result in a sub-domain of the parameters defined by the following constraints:

Table 3.2: BWBN parameters ranges

Parameters	Support
k_i (kN/mm)	[4.27, 11.85]
β $(1/mm)$	[55, 65] $\times 10^{-3}$
γ	$[-\beta, \beta]$
α	[0, 1]

The support on model parameters k_i was selected to limit the range of variation of the BWBN system frequency about $f = 3 \div 5\,Hz$, whereas the inequality constraint on γ has been set to ensure the BIBO (Bounded Input–Bounded Output) stability [32]. Two optimization algorithms have been used for the identification of parameters: (a) Pattern Search, and (b) Interior Point. Both algorithms share the following objective function to be minimized:

$$J = \|v_{exp}(t) - v_k(t)\|^2 \tag{3.22}$$

where $v_{exp}(t)$ is the experimental record as generated in SECTION 3.3, whereas $v_k(t)$ is the k-response-evaluation computed by the algorithm.

Once identified the parameters (**Table 3.3**), a nonlinear analysis was run. In the next page, in **Figure 3.5** and **Figure 3.6** are compared the results of the identified response with the experimental data. It can be clearly seen as both algorithms replicate quite satisfactorily the system velocity response, however they miss in capturing the actual hysteretic behavior. This can be explained by the fact that stating the problem in these terms means to be *"blind"* to uncertainties and model discrepancies; in other words, the randomness nature of the event jointly to the incapability of the model to represent the real uncertain world is neglected. So that, conducting such a calibration, none information can be provided on the inaccuracy of the model prediction.

Table 3.3: BWBN identified parameters

Parameters	Identified value		True value
	Pattern Search	*Interior Point*	
k_i	4	5.00	7.6
β	65×10^{-3}	57×10^{-3}	65×10^{-3}
γ	-0.023	0	-0.051
α	0	0.5	0.459

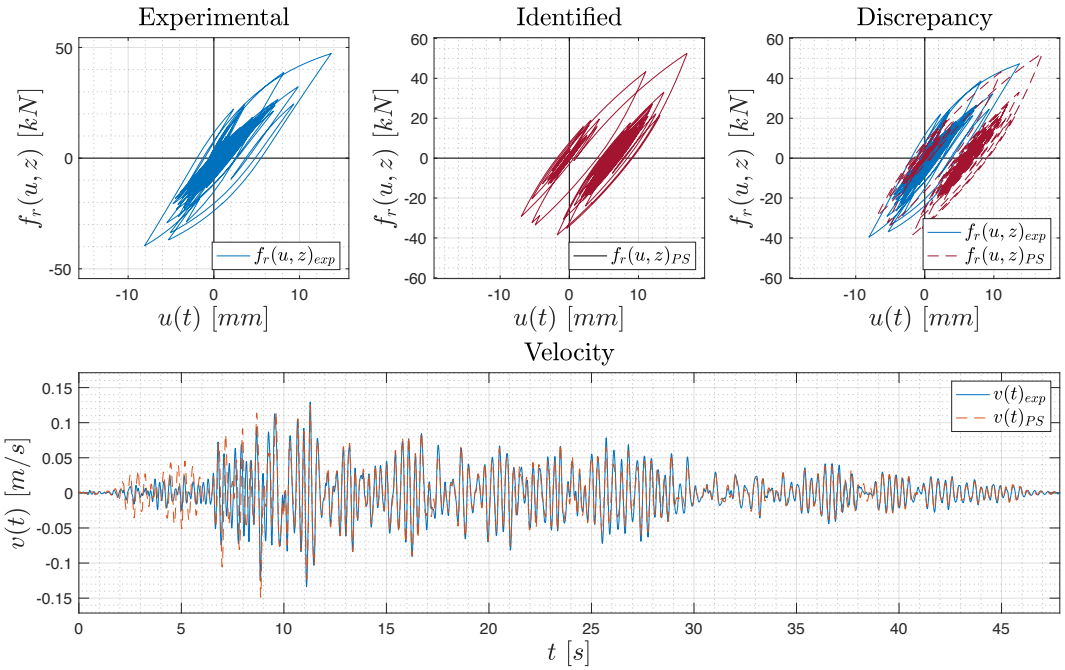


Figure 3.5: Pattern Search algorithm

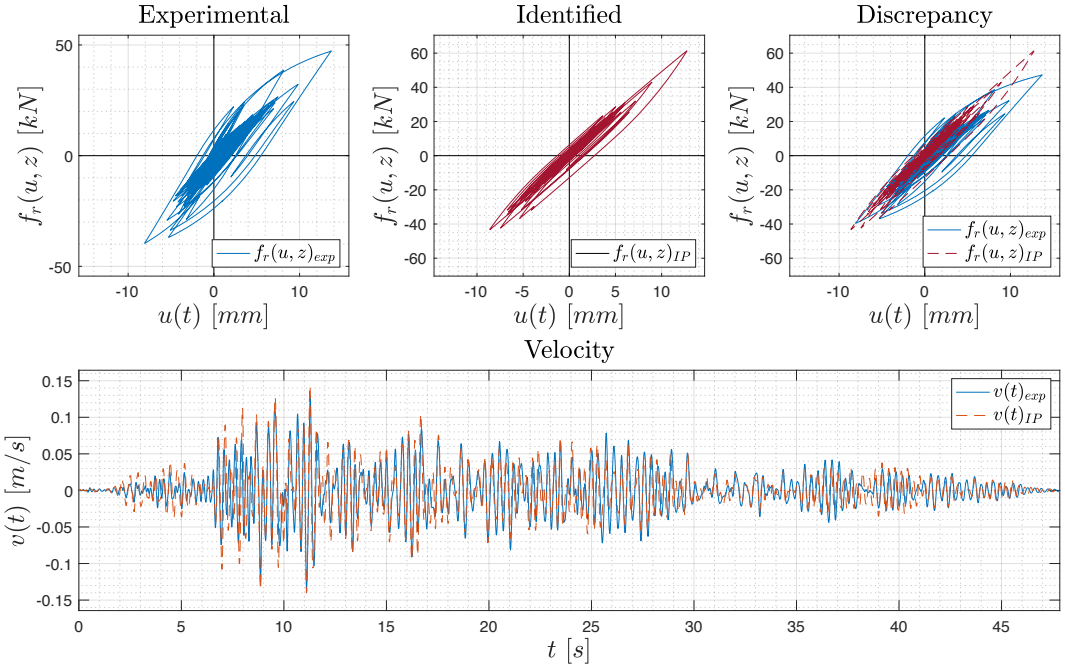


Figure 3.6: Interior Point algorithm

3.5 Calibration accounting random variable discrepancy

The numerical case of study of the model calibration of the Bouc-Wen-Baber-Noori system is now dealt with accounting model uncertainty by using a Bayesian approach. In the view of Bayesian inverse problems, the lack of knowledge on the input parameters is taken into account by considering them as realizations of random vectors. Specifically, deterministic values are used for the following parameters of the BWBN model: $m = 12000\,kg$, $\xi = 3\%$, $A = 1$, $N = 1$. The remaining one are considered independent random variables with associated distributions given in **Table 3.4**, which constitute the vector of uncertain model parameters $\boldsymbol{x}_{\mathcal{M}} = \{k_i, \beta, \gamma^{\star}, \alpha\}$.

Table 3.4: Prior distributions of the BWBN model parameters

Parameter	Distribution	Support	Mean	Standard deviation
k_i (kN/mm)	Gaussian	[4.27, 11.85]	7.42	0.1
β $(1/mm)$	Uniform	$[55, 65] \times 10^{-3}$	60×10^{-3}	2.8×10^{-3}
$\gamma^{\star}$	Uniform	$[-1, 1]$	0	5.7×10^{-1}
α	Lognormal	[0, 1]	0.063	0.01

The first moments of the prior PDFs of model parameters k_i and α, namely $\mu_{k_i} = 7.42$ and $\mu_{\alpha} = 0.063$, have been selected by two linear regressions on the experimental restoring force curve (**Figure 3.7**).

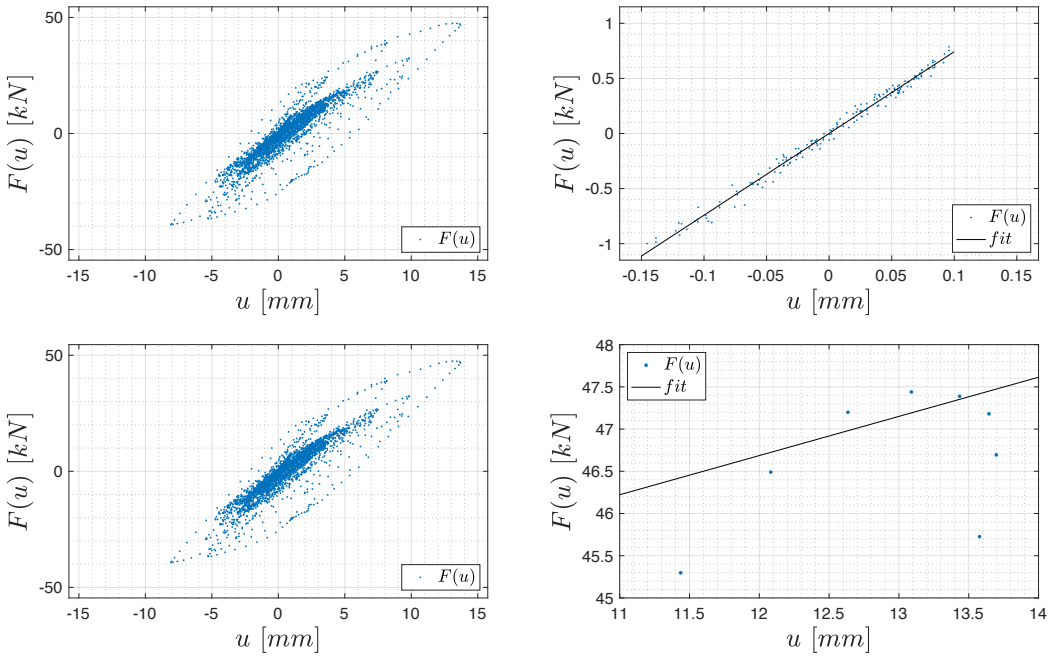


Figure 3.7: Linear regressions for prior moments estimates

It is worth to underly that in **Table 3.4** a new random variable $\gamma^{\star}$ was introduced in order to guarantee the BIBO stability. Indeed, assuming as prior knowledge on

the parameter γ the required BIBO stability condition, i.e. γ being uniformly distributed between the bounds: $-\beta \leq \gamma \leq \beta$, the conditional distribution $\pi(\gamma|\beta)$ has to be uniform. This can be achieved by transforming the input variables, namely removing the parameter γ and introducing an auxiliary variable $\gamma^\star \sim \mathcal{U}(-1,1)$. So that, for a joint realization of the parameters β and $\gamma^\star$, the actual parameter reads: $\gamma = \gamma^\star\beta$.

As seen in CHAPTER 2, all computational models are always simplifications of the real world. As consequence, a discrepancy term has been introduced to connect the Bouc-Wen-Baber-Noori model response to the experimental observations. For the sake of simplicity the assumption of a simple additive Gaussian discrepancy with mean null and residual diagonal covariance matrix $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}$ with unknown residual variances σ^2 has been made, i.e.:

$$\varepsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma} = \sigma^2\boldsymbol{I}) \tag{3.23}$$

To infer σ^2, a uniform prior distribution was instead assumed:

$$\pi(\sigma^2) \sim \mathcal{U}\big(0, \max|\boldsymbol{v}|^2\big) \tag{3.24}$$

whose standard deviation was set equal to the maximum of the absolute value over the time of the experimental observation $\boldsymbol{v}$.

Under this assumption the likelihood function is set as a Gaussian distribution with unknown variance and mean value equal to the norm 2 squared of the difference between the model prediction $\mathcal{M}(\boldsymbol{x}_\mathcal{M})$ and the observation $\boldsymbol{v}$, in accordance with the objective function assumed in SECTION 3.4 (eq. (3.22)):

$$\mathcal{L}(\boldsymbol{x}_\mathcal{M}, \sigma^2; \mathcal{V}) = \prod_{i=1}^{N} \frac{1}{\sqrt{(2\pi\sigma^2)^N}}\, e^{-\frac{1}{2\sigma^2}\left(\boldsymbol{v}_i - \mathcal{M}(\boldsymbol{x}_\mathcal{M})\right)^\intercal\left(\boldsymbol{v}_i - \mathcal{M}(\boldsymbol{x}_\mathcal{M})\right)} \tag{3.25}$$

In this way the posterior distribution can be computed as:

$$\pi(\boldsymbol{x}_\mathcal{M}, \sigma^2|\mathcal{V}) = \frac{1}{Z}\,\pi(\boldsymbol{x}_\mathcal{M})\pi(\sigma^2)\,\mathcal{L}(\boldsymbol{x}_\mathcal{M}, \sigma^2; \mathcal{V}) \tag{3.26}$$

that summarizes the updated information about the unknowns $\boldsymbol{x} = \{\boldsymbol{x}_\mathcal{M}, \sigma^2\}$ after conditioning on the observation $\mathcal{V}$.

In the following sections Bayesian inversion was carried out first using the reference computational model $\mathcal{M}^{BW}(\boldsymbol{x})$ for representing the velocity time-histories $v(t)$ of the BWBN model, then using surrogate models.

3.5.1 Reference model

Markov Chain Monte Carlo simulations have been conducted using Matlab-based Uncertainty Quantification framework UQLab-V1.3-113® [44] with 100 chains,

700 steps and AIES solver algorithm. The number of the BWBN forward model $\mathcal{M}^{BW}(\boldsymbol{x})$ calls in MCMC was 70000, that means solving, for each model evaluation, the system of ODEs (3.19) with the Matlab solver `ode45` (explicit Runge-Kutta method with relative error tolerance 1×10^{-3}), for a total duration of about 33 hours. Latin hypercube sampling (LHS) method was used to get model parameters prior distributions (**Table 3.4**), which are shown in **Figure 3.8**.

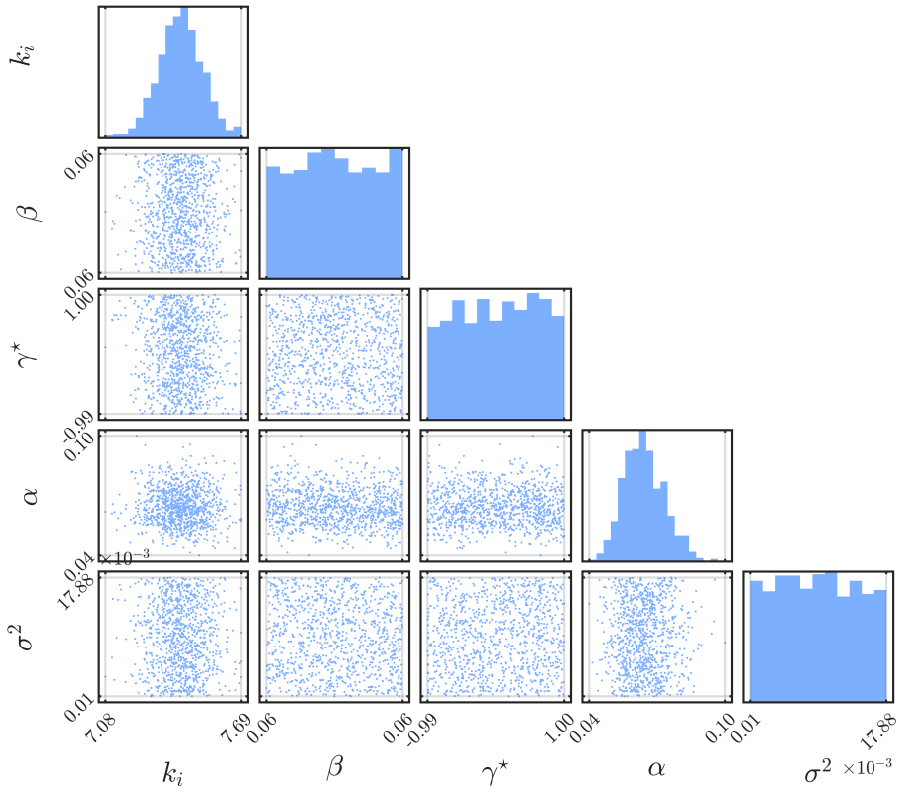


Figure 3.8: Prior samples

The evolutions of the Markov chains are plotted in **Figure 3.9** for each model parameter $\boldsymbol{x} = \{k_i, \beta, \gamma^\star, \alpha, \sigma^2\}$; these give valuable insights about convergence of the chains. Indeed, it can be clearly seen as 700 steps are sufficient to reach the steady state. So that, samples generated by the chains follow the posterior distributions. However, the sample points generated by the AIES MCMC algorithm before convergence, can pollute the estimate of posterior properties, therefore they have been post-processed carrying out the *burn-in* of the first half of sample points (in other words their contribution was discarded). Furthermore, in the same figure, the acceptance rate r_a of the chains is reported. This parameter can be seen as an indicator of a badly tuned algorithm [67]. In practical applications r_a close to one indicates that the proposal distribution does not sufficiently explore the target distribution, while r_a close to zero indicates rather that the proposed candidate points are in low probability regions; the optimal acceptance rate is shown to approach $r_a = 0.23$ [61]. With this evidence, it can be concluded that in our case the quality of the generated MCMC chains can be judged satisfactory.

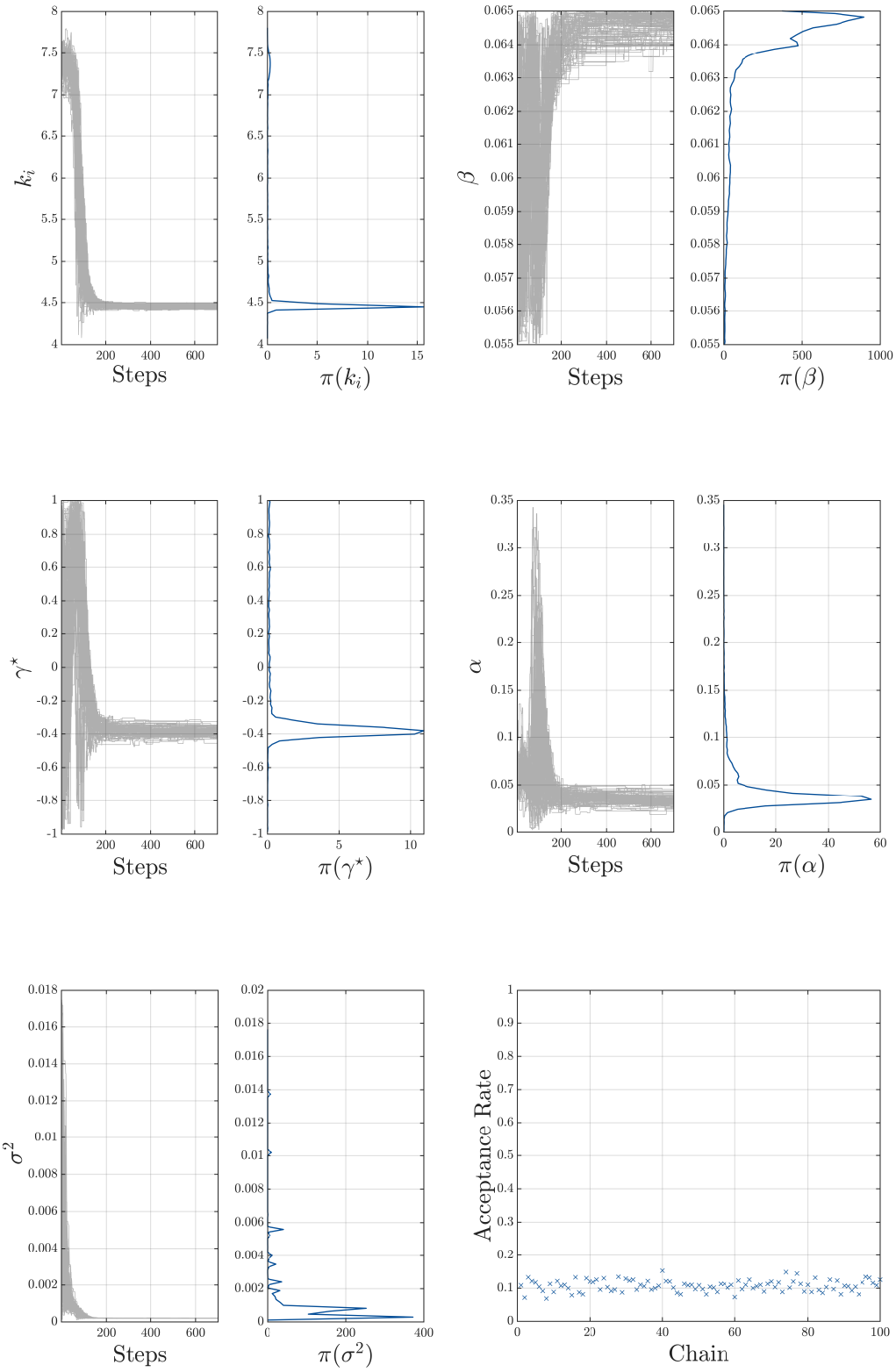


Figure 3.9: Trace plots and acceptance rate of the chains

Table 3.5 reports the result of the Bayesian inversion analysis: mean and standard deviation of the posterior, $5\% - 95\%$ quantiles of the distribution, MAP point estimate. Whereas the posterior distributions of the calibrated parameter are plotted in **Figure 3.10**.

Table 3.5: Posterior marginals

Parameter	Mean	Standard deviation	(0.05-0.95) Quant.	MAP
k_i	4.5	0.18×10^{-1}	(4.4 - 4.5)	4.4
β	65×10^{-3}	0.36×10^{-3}	(64 - 65) $\times 10^{-3}$	64×10^{-3}
$\gamma^\star$	-0.38	0.25	$(-0.42$ - $-0.34)$	-0.40
α	0.035	0.53×10^{-2}	(0.027 - 0.044)	0.031
σ^2	2×10^{-4}	4.7×10^{-6}	(1.9 -2.1) $\times 10^{-4}$	2×10^{-4}

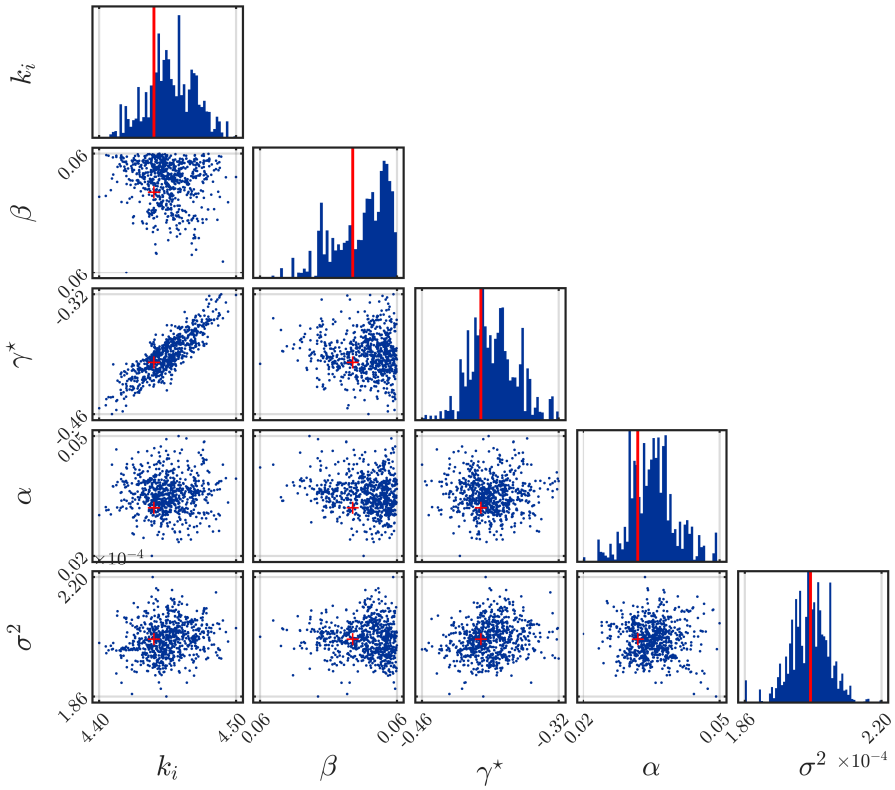


Figure 3.10: Posterior samples

The maximum a posteriori value (MAP) can be considered the most probable parameter value following calibration. This value has been used as a best-fit parameter. The model response prediction using MAP as point estimate of the posterior distributions is plotted in **Figure 3.11**. It can be clearly seen as the inferred response reproduces quite accurately the experimental record. Actually, we should not dwell upon this specific prediction, rather upon its confidence intervals. The latters tell us how uncertainties on model input propagate through the model. From the inspection of **Figure 3.11** can be concluded that uncertainties on the

input parameters produce higher uncertainty of the response prediction in the lower amplitude regions rather than in the peak values ones. However, this is a satisfying result in earthquake engineering, where we are more interested in predicting maximum response quantities.

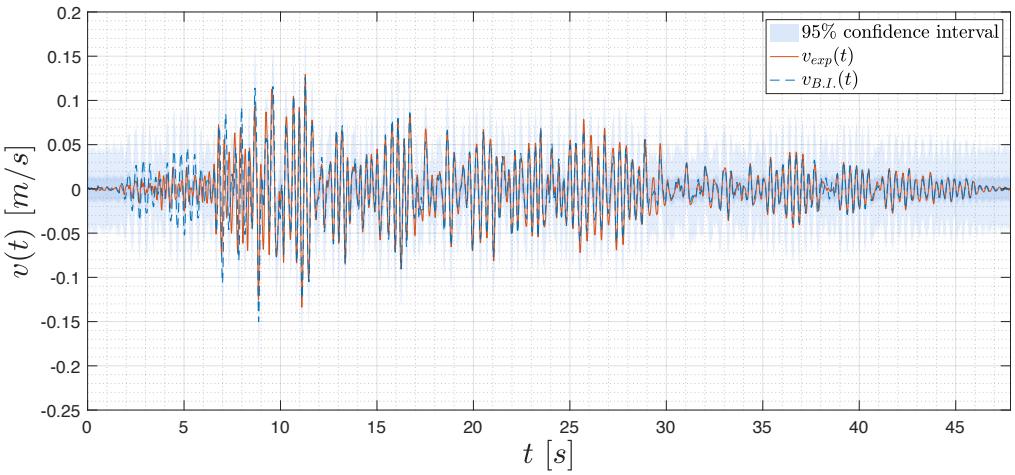


Figure 3.11: Model response

As mentioned at the beginning of the section, the number of forward model calls in MCMC simulations was 70000. This lead to a total analysis duration of about 33 hours. It is worth underlining that the system considered previously, has only a single degree of freedom. Dealing with actual structures means taking into account thousand/million DOFs. As consequence, it appears clear from that the need of surrogate modelling which, at lower computational cost, approximate accurately the original expensive computational model.

3.5.2 NARX model

To build the NARX model for representing the velocity time-histories $v(t)$ of the BWBN system, as first 1000 samples of the input parameters were generated by LHS and 1000 corresponding model simulations were conducted. These data constitute the Experimental Design.

Then a NARX model structure with absolute terms was chosen based on their effectiveness in capturing the hysteretic behaviour of nonlinear systems [65], and whose basis terms are defined by:

$$g_i(t) = x(t-k)^l |v(t-1)|^m \tag{3.27}$$

$$g_i(t) = v(t-j)^l |v(t-1)|^m \tag{3.28}$$

with $l = 0, 1$, $m = 0, 1$, $k = 0, ..., n_x$, $j = 1, ..., n_y$, and $n_x = 4$, $n_y = 4$. So that the initial full NARX model contains totally 19 terms.

Next, the candidate NARX models were computed. For this purpose, LARS was applied to the initial full NARX model for each experiment of the ED, to select the most relevant terms. This lead to have 40 NARX candidates in total. OLS (eq. (2.62)) was used then to determine the NARX coefficients ϑ_i for all the simulations. Subsequent to the evaluation of the accuracy of the NARX candidates (eq. (2.62)), the most appropriate NARX model with a mean relative error of $\bar{\varepsilon} = 7.74 \times 10^{-3}$ over 1000 experiment, was selected among the candidates. This contains 10 model terms, namely: $x(t)$, $x(t-1)$, $x(t-3)$, $x(t-4)$, $x(t)|v(t-1)|$, $x(t-3)|v(t-1)|$, $v(t-1)$, $v(t-4)$, $v(t-1)|v(t-1)|$, $v(t-4)|v(t-1)|$.

Figure 3.12 depicts the cross-validation plot of the velocities (resp. maximal velocity) predicted by the NARX model against the reference values obtained with the numerical solver. Note that the the accuracy is high for the prediction peak values.

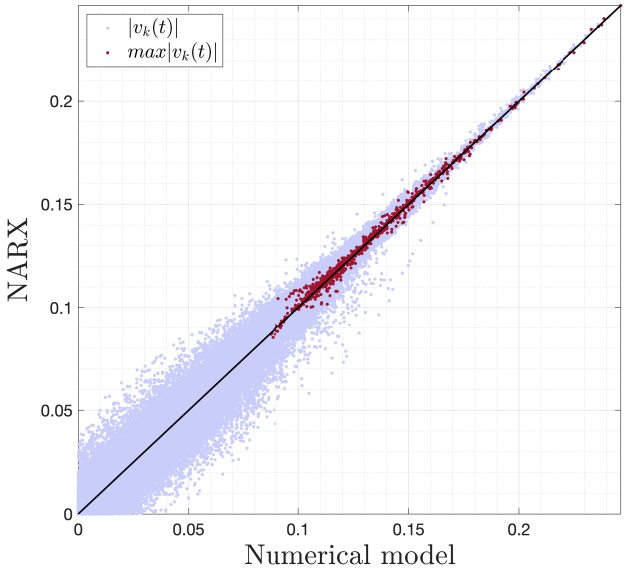


Figure 3.12: NARX free-run-reconstruction of the entire ED

In **Figure 3.13**, as an example, is plotted the NARX prediction for the $k = 2$ experiment of the ED.

3.5.3 PC-NARX and Kriging-NARX surrogate models

Once detected the NARX model, its coefficients computed on the ED have been used to train both PCEs and Kriging metamodels. Specifically, the $\vartheta_i(\boldsymbol{x})$ coefficients have been represented by sparse adaptive PCEs with degree up to $p = 21$, maximum interaction order $r = 2$ and truncation parameter $q = 1$. Jointly, the NARX coefficients have been represented by GP with hyperbolic truncation scheme, constant trend, ellipsoidal Matern-5/2 correlation, using Hybrid Genetic optimization Algorithm (HGA) to compute the hyperparameters. The PCEs of the NARX coefficient have LOO errors smaller than $\varepsilon_{LOO} = 9.766 \times 10^{-3}$, whereas a value of $\varepsilon_{LOO} = 4.391 \times 10^{-3}$ is reached with the Kriging.

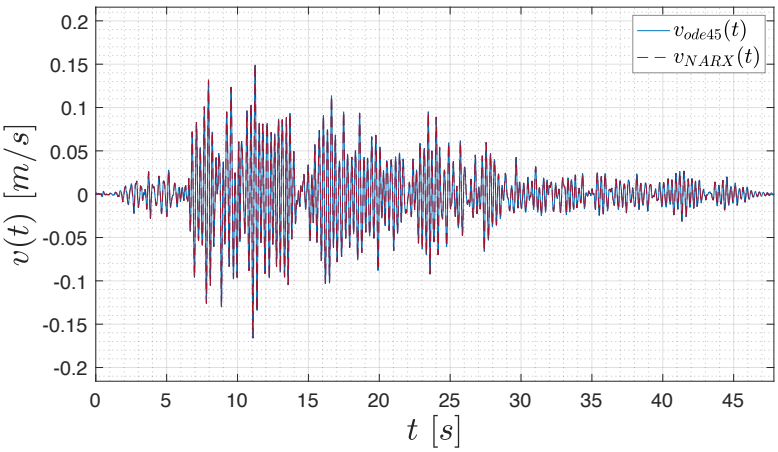


Figure 3.13: NARX free-run-reconstruction for the experiment $k = 2$

Figure 3.14 depicts the estimate of the NARX coefficient by the PC expansions versus the ϑ_i value of the ED, in order to evaluate the accuracy of the surrogate. As a check, the Kriging's estimate is also superposed. It should be not surprising that the cross-validation of the latter is exact, due to the fact that GP assumes as estimate on ED the same ED value. In contrast, it can be clearly seen as the PCEs spread some degree of uncertainty. Hence, Kriging works fine for numerical surrogates that are not affected by accidental errors, whereas PCEs are more adapt to mimic actual models which are affected by experimental errors.

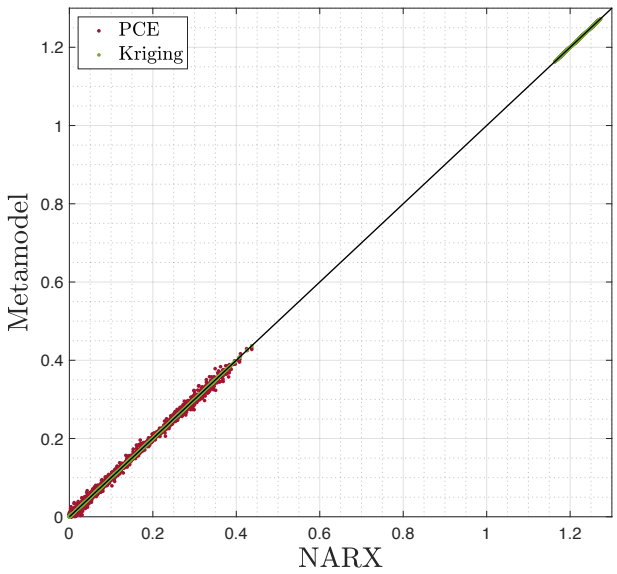


Figure 3.14: NARX coefficients cross-validation PCEs vs Kriging

Successively, the two surrogates PC-NARX and Kriging-NARX have been used for predicting the velocity response over a validation set of size 10^4, with a mean relative error respectively of: $\bar{\varepsilon}_{val} = 9.655 \times 10^{-3}$ and $\bar{\varepsilon}_{val} = 9.002 \times 10^{-3}$. The

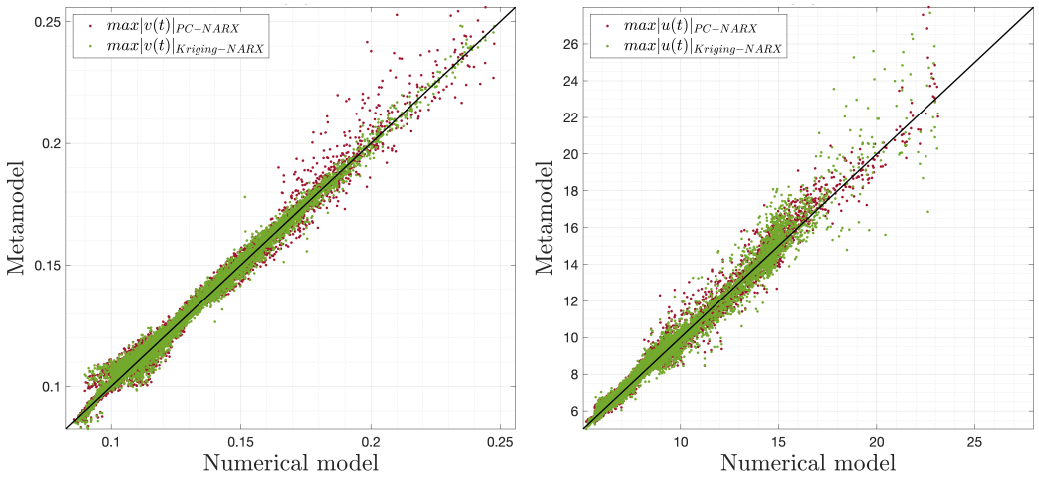


Figure 3.15: PC-NARX - Kriging-NARX validation

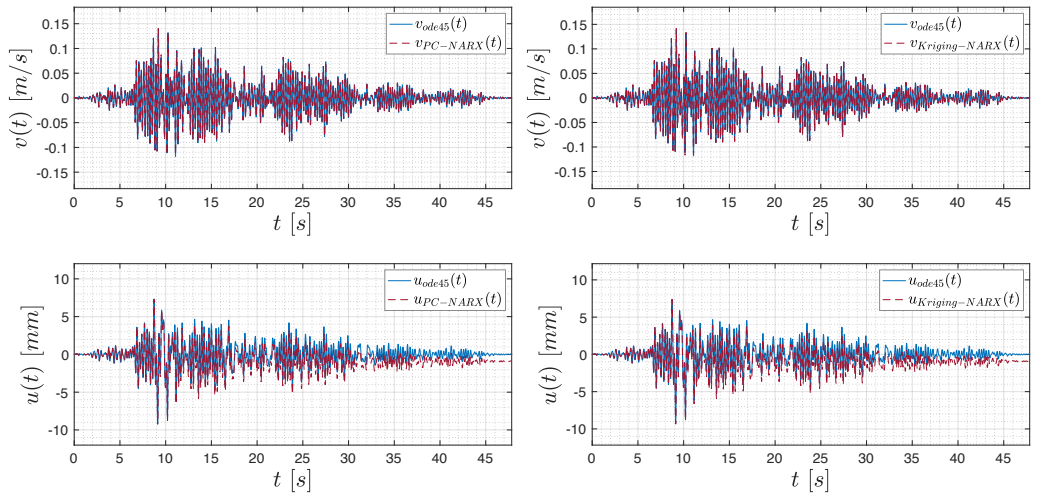


Figure 3.16: PC-NARX and Kriging-NARX prediction on validation set

displacement time history was then obtained by integration. Cross-validation plots are reported in **Figure 3.15**. It is evident from its observation as Kriging-NARX is a bit more accurate than the PC-NARX in the prediction.

As an example, **Figure 3.16** depicts two specific velocity and displacement trajectories for a realization of the BWBN parameters onto the validation sets. Note that the velocity trajectories are perfectly predicted by both surrogates. Despite the high accuracy of the PC-NARX and Kriging-NARX models for the velocity, the displacements obtained by integration might exhibit some slight discrepancies with respect to the actual responses. The inaccuracies occur when the peak values of the velocity are not well predicted, as result the error is accumulated in time through integration. This is due to the fact that the different response quantities might experience their largest variabilities at different instants of time. A possible solution to cut off the error accumulation consists in applying an high pass filter to the velocity response and then integrating to obtain the displacements time

history. However, it should be stressed that with such approach, due to the filter processing, potential permanent displacements cannot be captured.

3.5.4 Bayesian inversion using PC-NARX metamodel

The same MCMC simulations, as seen in SECTION 3.5.1, with 100 chains and 700 steps, have been conducted using the surrogate PC-NARX model, for a total metamodel calls of 70000 and a total duration of about 55 minutes.

The evolutions of the Markov chains are plotted in **Figure 3.17** for each model parameter $\boldsymbol{x} = \{k_i, \beta, \gamma^\star, \alpha, \sigma^2\}$. As for the reference model, 700 steps are sufficient to reach the steady state even though not unique-value convergence was reached. So that, samples generated by the chains follow multimodal posterior distributions. A plausible cause of this may consist in the systematic propagation of the error of the PC-NARX metamodel prediction through the Markov chains. Indeed, comparing **Figure 3.9** with **Figure 3.17**, it can be clearly seen how each model parameter distribution, except for $\pi(\beta)$, matches quite well the one obtained using the reference model excluding for the additional modes. Much more uncertainty arises instead in the calibration of β, this may be ascribed at its sensibility, i.e. on how and how much this specific parameter influence the system response.

To the estimate of the posterior samples, as before, a *burn-in* of the first half of sample points was carried out (**Figure 3.18**). **Table 3.6** reports the result of the Bayesian inversion analysis: mean and standard deviation of the posterior, $5\% - 95\%$ quantiles of the distribution, MAP point estimate.

Table 3.6: Posterior marginals (PC-NARX)

Parameter	Mean	Standard deviation	(0.05-0.95) Quant.	MAP
k_i	4.6	0.11	(4.4 - 4.7)	4.5
β	61×10^{-3}	0.17×10^{-2}	(58 - 63) $\times 10^{-3}$	63×10^{-3}
$\gamma^\star$	-0.14	0.19	$(-0.35$ - $-0.089)$	-0.33
α	0.043	0.98×10^{-2}	(0.03 - 0.058)	0.052
σ^2	2×10^{-4}	5.1×10^{-6}	(2 -2.2) $\times 10^{-4}$	2×10^{-4}

Finally, the model response prediction using MAP as point estimate of the posterior distributions is used to fit the observations (**Figure 3.19**). The accuracy of the model response prediction using the PC-NARX surrogate is comparable to the one of the reference model. Although more uncertainties are introduced by the surrogates, its time-computation benefit is remarkable (about 30 times faster).

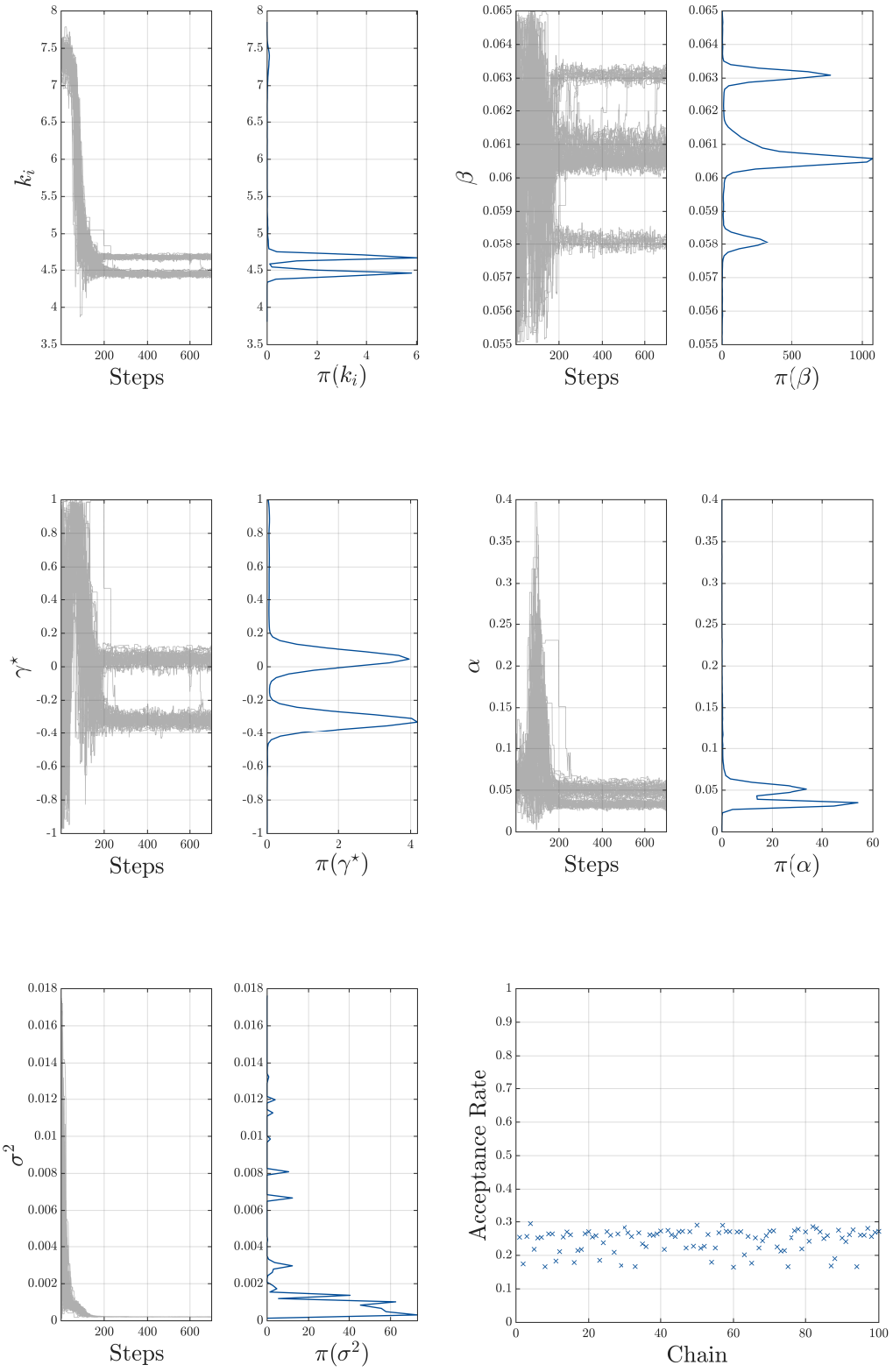


Figure 3.17: Trace plots and acceptance rate of the chains (PC-NARX)

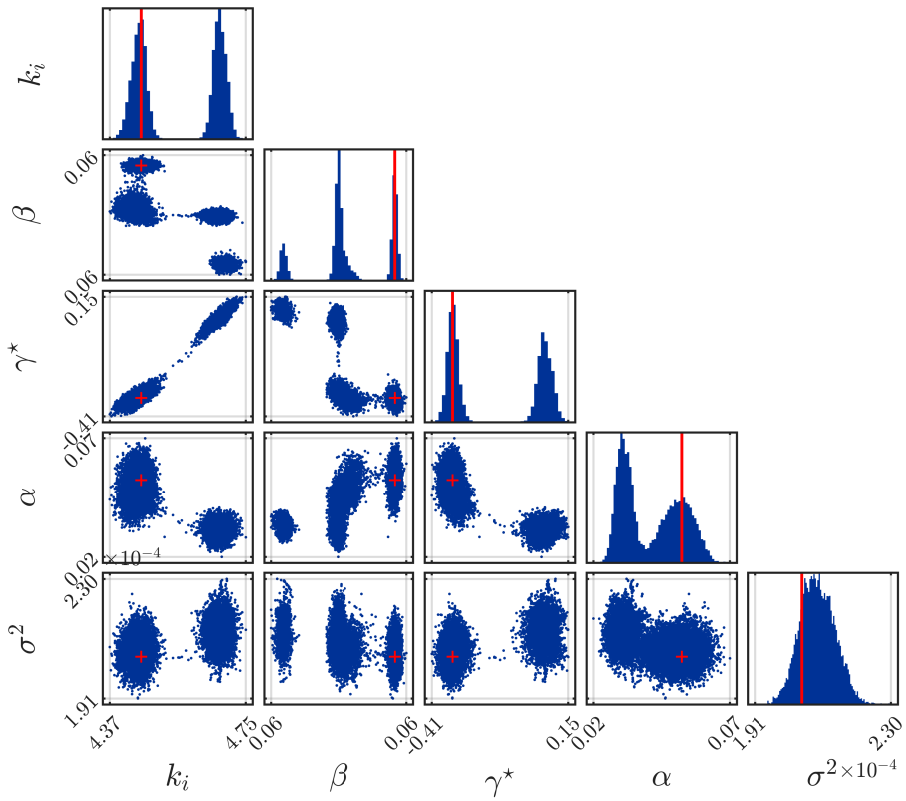


Figure 3.18: Posterior samples (PC-NARX)

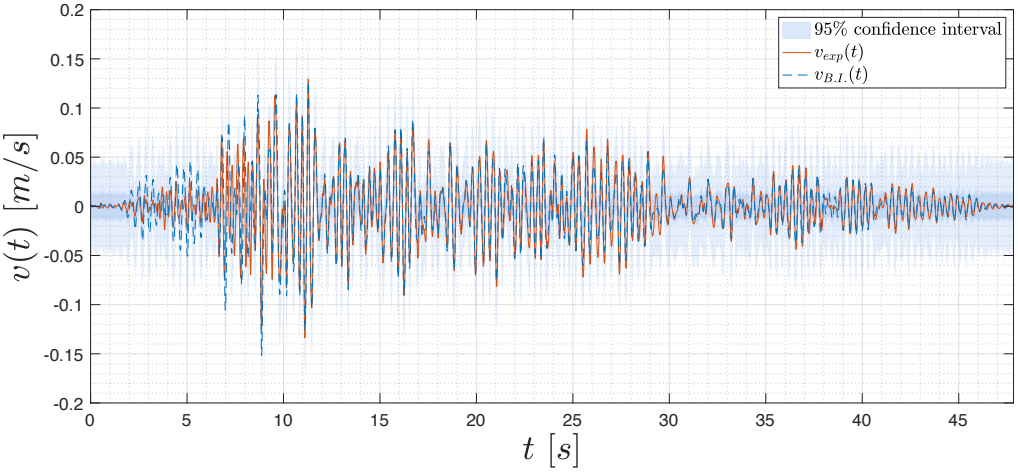


Figure 3.19: Model response (PC-NARX)

3.5.5 Bayesian inversion with Kriging-NARX

The same MCMC simulations with 100 chains and 700 steps, have been conducted using also the Kriging-NARX surrogate model, for a total metamodel calls of 70000 and a total duration of about 18 hours.

The evolutions of the Markov chains are plotted in **Figure 3.20** for each model parameter $\boldsymbol{x} = \{k_i, \beta, \gamma^\star, \alpha, \sigma^2\}$. As for the reference model and for the PC-NARX surrogate, 700 steps are sufficient to reach the steady state. However, in contrast with the inference by means of the PC-NARX metamodel, samples generated by the chains follow now unimodal posterior distributions. It is evident from that how the Kriging-NARX surrogate not affects the inference analysis propagating prediction errors through the Markov chains.

To the estimate of the posterior samples a *burn-in* of the first half of sample points also in this case was carried out (**Figure 3.21**). **Table 3.7** reports the result of the Bayesian inversion analysis: mean and standard deviation of the posterior, $5\% - 95\%$ quantiles of the distribution, MAP point estimate.

Table 3.7: Posterior marginals (Kriging-NARX)

Parameter	Mean	Standard deviation	(0.05-0.95) Quant.	MAP
k_i	4.5	0.01	(4.48 - 4.51)	4.5
β	64×10^{-3}	0.25×10^{-3}	(63.6 - 64.2) $\times 10^{-3}$	64×10^{-3}
$\gamma^\star$	-0.22	0.014	$(-0.25$ - $-0.2)$	-0.23
α	0.023	0.25×10^{-2}	(0.019 - 0.027)	0.023
σ^2	1.9×10^{-4}	3.9×10^{-6}	(1.8 - 1.95) $\times 10^{-4}$	2×10^{-4}

Finally, the model response prediction using MAP as point estimate of the posterior distributions is used to fit the observations (**Figure 3.22**).

Summary

In this chapter the model calibration of the nonlinear hysteretic SDF Bouc-Wen-Baber-Noori system subjected to ground motion excitation was treated. It was shown first as deterministic approaches cannot furnish a complete overview on the model and its parameters and on propagation of uncertainties, then how stochastic approaches succeed reach to this aim. Moreover it was shown how conducting a proper Bayesian inference may result time and computationally expansive (remember that the total analysis duration of the reference model was about 33 hours). To get time improvements polynomial chaos expansions and Kriging metamodels were introduced coupled with non-linear autoregressive with exogenous input models (PC-NARX and Kriging-NARX). As result, it was pointed out how the accuracy of this metamodels in the model response prediction is comparable to the one of the mathematical physical model with a significant time-computation improvement (of about 30 and 2 times fast for PCE-NARX and Kriging-NARX respectively).

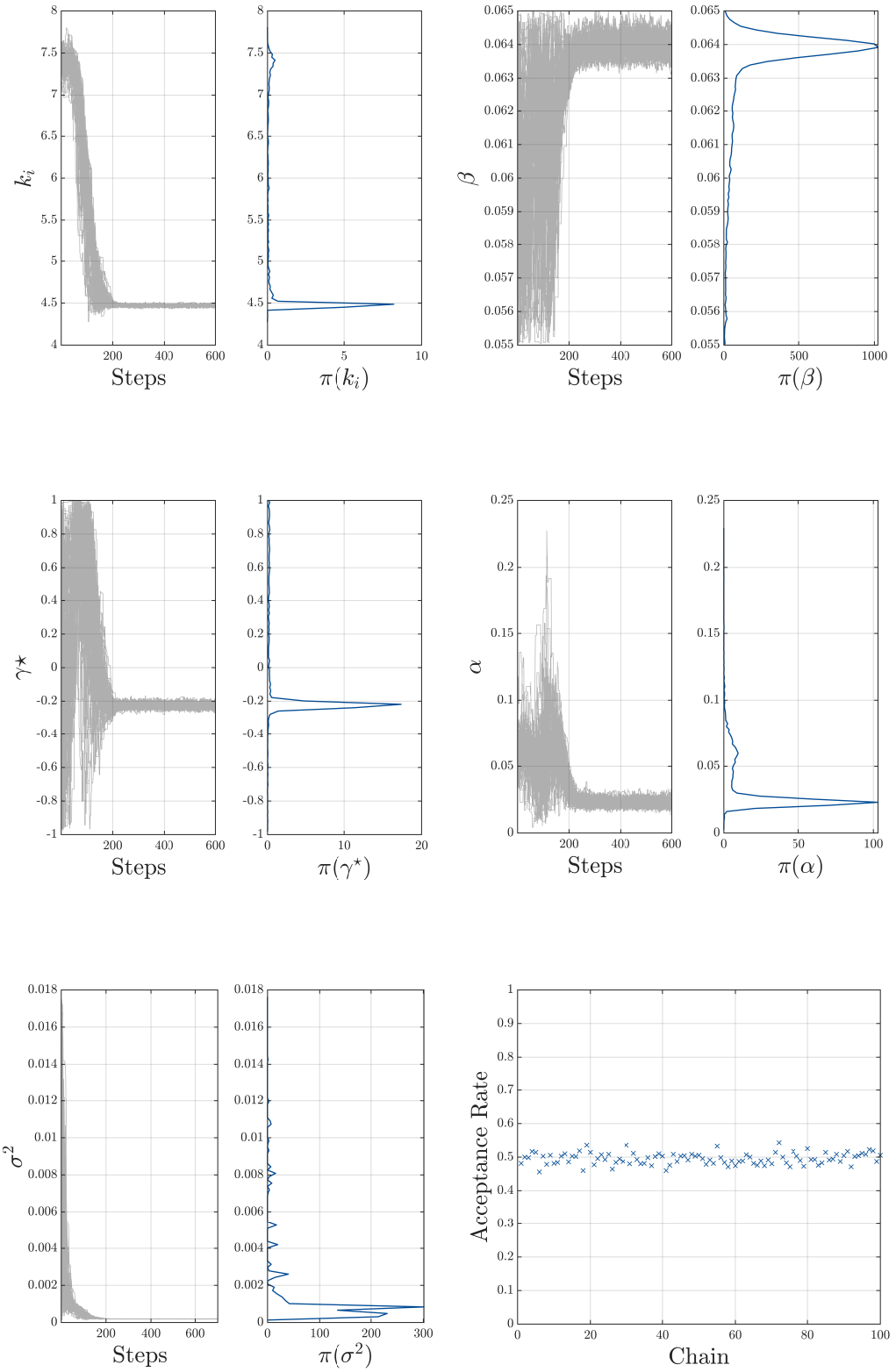


Figure 3.20: Trace plots and acceptance rate of the chains

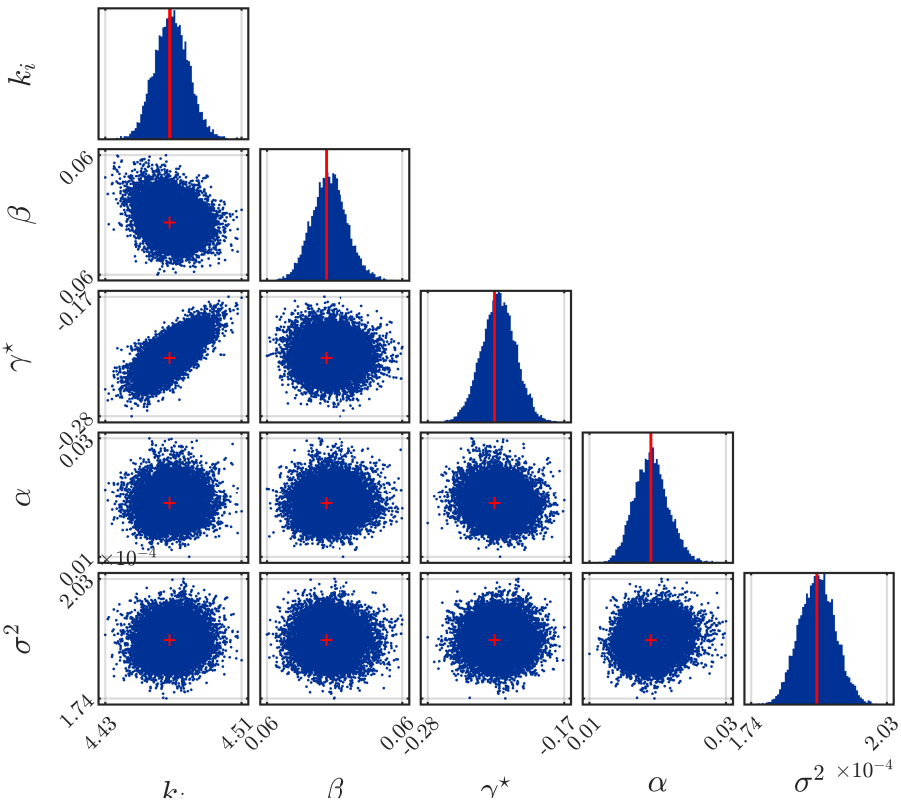


Figure 3.21: fig: Posterior samples (Kriging-NARX)

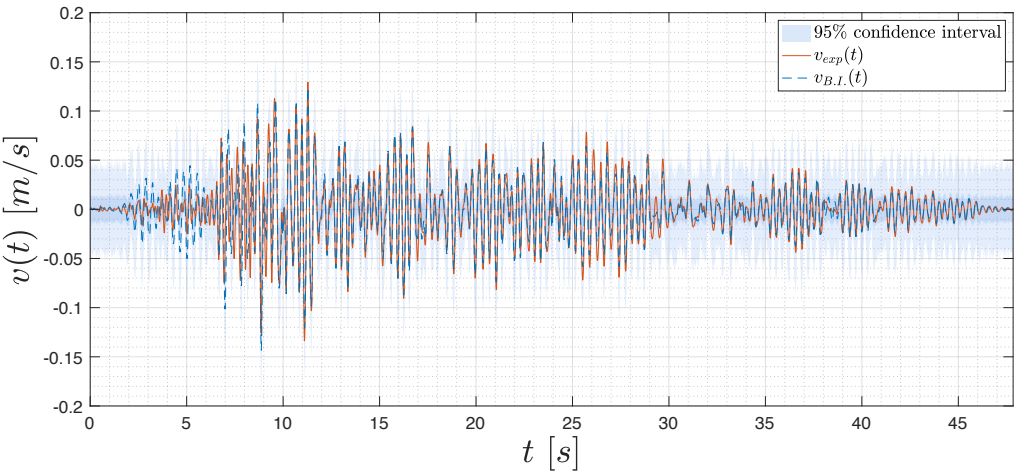


Figure 3.22: Model response (Kriging-NARX)

Chapter 4

Hybrid simulation tests of the masonry facade case study

This chapter deals with the case study description. In detail, SECTION 4.1 introduces the Hybrid Simulation methodology and the architecture of the hybrid simulators. SECTION 4.2 concerns the description of the framework built at ETH Zürich to perform the pseudo-dynamic hybrid-simulations on the masonry facade system. Furthermore, the experimental testing campaign results are shown. The latters constitutes the data that will be used in CHAPTER 5 for the model calibration.

4.1 Hybrid simulations

Hybrid simulations (HS) are testing methods used to emulate structures' time-history responses under dynamic loads by combining numerical techniques and experimental procedures. The benefit of conducting such a simulation relies into the fact that it allows to split a system into a Physical Subdomain (PS) and a Numerical Subdomain (NS) which are parallelized and coupled after each time step (in this sense they are *"hybrid"*) [53]. Specifically, the seismic excitation is transferred experimentally via hydraulic actuators directly to the tested specimen, while the numerical analysis is performed by means of numerical models solved using time integration algorithms. So that, HS generally allows the sub-structuring of the system of interest into a well-known part modelled numerically, and a strongly nonlinear one tested by a physical experimental setup. Thus carrying experimental tests prevents modelling inaccuracies, whereas the numerical analysis leads to a less extensive testing setup for simulations [28].

As first, the structure is discretized with a certain amount of degrees of freedom (DOFs). In the experimental test setup, each DOF is controlled by a servo-actuator. Stiffness parameters of the structure are detected from the physical specimen, while the inertia forces are modelled numerically [28].

When the response of the physical setup is assumed being independent of the rate of loading, the implementation of a *pseudo-dynamic* Hybrid Simulation at an extended time scale is practicable. This testing procedure contributes to less

control tracking errors and less destabilizing impacts of experimental substructure with the actuators [2].

The system's equations of motion expressed in the state-space form for each subdomain (experimental and analytical) are then solved by a partitioned time integration algorithm [1] adopting a dual assembly scheme of the modified PH method [13]. Precisely, the displacements computed for each DOF, are imposed on the sub-structures. The corresponding actuators regulate the displacements of the PS in a quasi-static manner. Once displacements have been imposed on the structure, the static restoring forces of the PS are measured via load cells included in the servo-actuators, while the restoring forces of the NS are calculated numerically. Both obtained restoring forces are introduced in the equations of motion to be solved in the integration scheme during the next time step. The newly computed displacements are again imposed on the structure, repeating the Hybrid Simulation loop [28] (**Figure 4.1**).

It is worth to underlying that the simulation time step used in the time integration algorithm differs from the one used for the controller (that corresponds to the specimen-clock-time in the experimental setup). Their ratio is defined by the parameter λ, whose value is usually set about $50 \div 200$ [54].

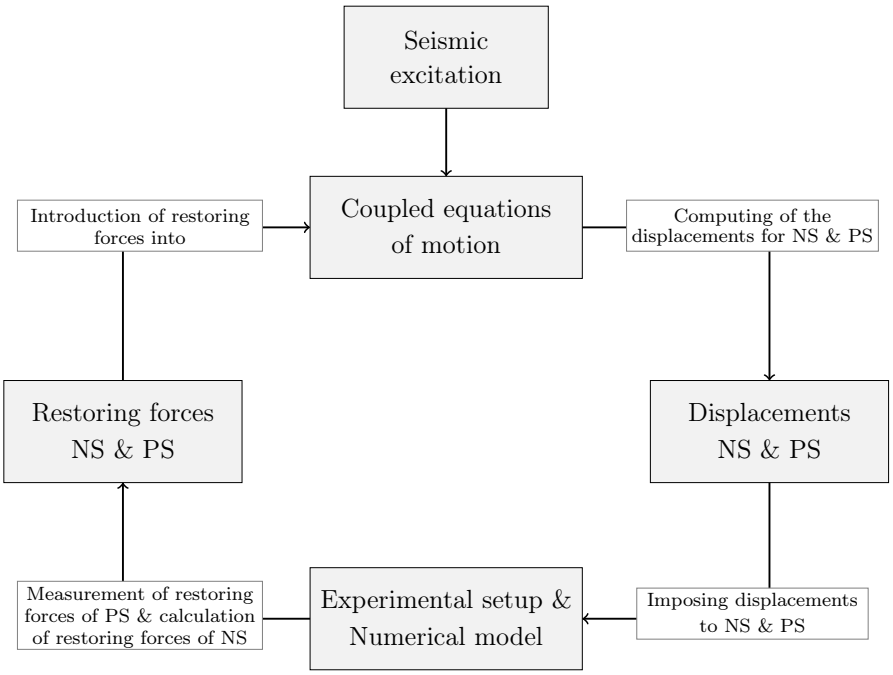


Figure 4.1: Hybrid Simulation loop [28]

4.1.1 Architecture of the hybrid simulator

The main components of a Hybrid Simulation are the experimental equipment, the computational apparatus (controller, host and real-time computers), the hydraulic units for actuator control and the measurement devices (e.g. digital image correlation (DIC) and linear variable displacement transducers (LVDT)). **Figure 4.1** depicts the architecture of the hybrid simulator.

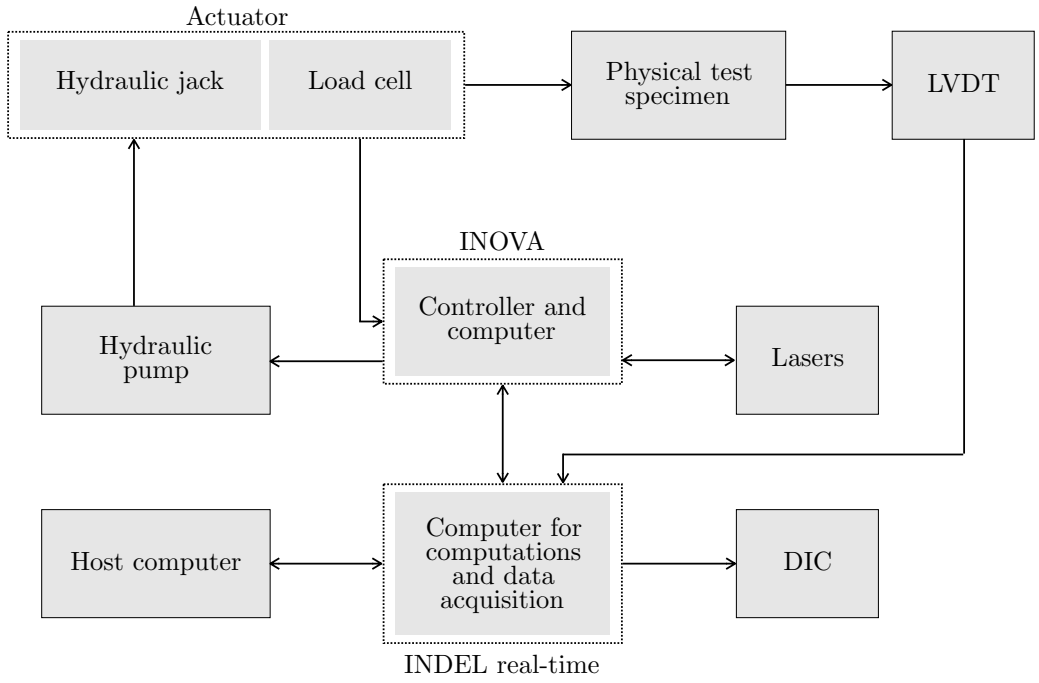


Figure 4.2: Architecture of the hybrid simulator [28]

Host computer

The host computer is used to build a Hybrid Simulation software (based on Matlab Simulink) and to handle the real-time computer. The Simulink model consists of three blocks: the input block acquires the data from the sensors, i.e. the restoring forces measured on the test specimen by the load cells in the actuators; the acquired data is then sent to the block containing the hybrid simulator algorithm which computes the new displacements to impose on the structure; from those the third block automatically generates C code to broadcast to the real-time computer.

INDEL real-time computer

The embedded real-time operating system of the INDEL real-time computer executes the host broadcasted tasks within a deterministic time scheduling [28] and transmits the commands to the controller. Simultaneously data acquisition from all the sensors and triggers and from the digital camera for the digital image correlation (DIC).

INOVA controller and computer

The real-time computer transmits displacement command at each time step of the simulation to the INOVA real-time controller system which manages the servo-controlled actuators. Simultaneously the controller sends force feedback to the real-time computer using the EtherCAT communication [28].

Actuator force measurement

As well as applying a certain force, the servo-hydraulic actuators are also able to send a force feedback to the controller through the load cells included.

Actuator displacement measurement

Two devices are used to measure the displacement of the actuator: the actuator internal sensor and an external laser. For vertical actuators, the displacements are measured via the internal sensor while the laser devices are just used to check the measurements.

Digital image correlation

Digital image correlation (DIC) is an optical measurement technique for capturing, tracking and measuring deformations and strains on the specimen surface comparing processed images of the specimen surface at different instants of time.

Linear variable displacement transducers

Linear variable displacements transducers (LVDTs) are mounted to the specimen surface in order to punctual monitor its deformations.

4.2 Description of the case study

In this section the case study of a planar masonry wall subjected to hybrid test is introduced. The facade system (**Figure 4.3**) is a two storey wall structure of $2.7 \times 5.2 \times 0.15\ m$ size, with two openings located one at the ground floor and one at the upper floor, respectively a door and a window.

The upper sub-structure of the facade was simulated numerically (NS), whereas the lower portion was tested in the laboratory (PS). To this aim an HS framework was built at ETH Zürich to perform PSD-HS of the system. As shown in **Figure 4.3**, in order to to guarantee a rigid interface between NS and PS, a steel beam assumed infinitely stiff, connects the tested specimen to the servo-controlled actuators which impose distributed boundary conditions. Indeed, the PS is subjected to vertical loading due to self-weight and the loads arising from upper storey masses. The static vertical stress applied on the substructure amounts to 10% of the masonry's final compressive strength.

4.2.1 Materials

The physical subdomain was made of calibrated Swiss K-Modul 5/19 clay bricks (type B according to SIA 266:2015) and standard cement mortar. The brick's nominal dimensions are $0.29 \times 0.15 \times 0.19\,m$ with a minimum norm characteristic compressive strength of $f_{bk,\min} = 28$ MPa. **Table 4.1** and **Table 4.2** [28] list further specifications and the results of the compression test performed according to SIA 266:2015 and SN EN 771-1:2011, respectively.

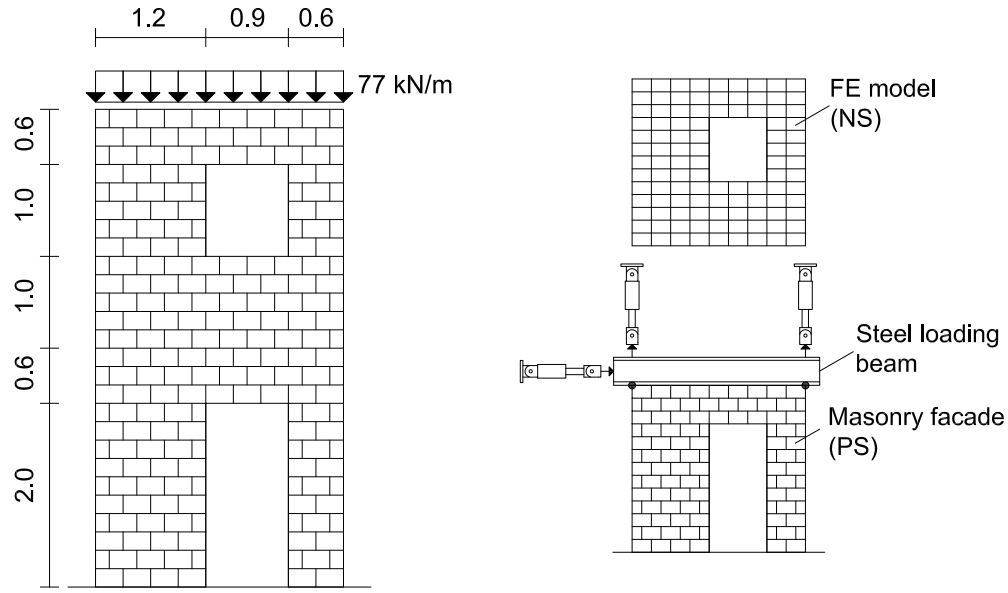


Figure 4.3: Masonry wall

Table 4.1: Material specifications of clay brick Swiss K-Modul 15/19 [28]

Clay brick Swiss K-Modul 15/19		
Length	290	mm
Width	150	mm
Height	290	mm
Void ratio	$25-55$	%
Rate of water absorption	< 3.5	$\frac{kg}{m^2 \min}$
Gross bulk density	NPD	
Compressive strength f_b	28	MPa
Bond strength	0.16	MPa
Water vapor permeability μ	4	

The cement mortar characteristic bending and compressive strengths are determined according to SN EN 196-1:2016 through bending and compressive testing performed on six cuboid mortar specimens of $40 \times 40 \times 160\,mm$ size and a storage time age of 62 days. Specifically, three samples (K1, K2 & K3) were stored in a climate room (constant air humidity at 95%), the others (S1, S2 & S3) were kept at the making site (laboratory) where the masonry walls for the Hybrid Simulation were built and stored. The setups used for the tests are shown in **Figure 4.4** [28], while the test results of the samples stored in the climate room (K) and at the making site (S) are shown in **Table 4.3** [28].

According to SN EN 1052-1:1998, compression test on three masonry specimens built and stored at the making site of $0.59 \times 0.15\,m^2$ nominal cross-section and 1 m height and with a bed joint thickness of approximately 10 mm, were performed in order to determine the compressive strength and the modulus of elasticity of the

Table 4.2: Compression test results of clay bricks [28]

Sample	F_{max} [kN]	f_{bk} [MPa]
1	985.0	22.7
2	1195.9	27.1
3	1194.0	27.1
4	760.9	17.4
5	871.1	19.9
6	1099.9	25.1
7	1123.9	25.7
8	1001.7	23.2
9	1185.8	27.1
10	1108.4	25.5
Average	1052.7	24.1
Std. deviation		3.3

Table 4.3: Test results of mortar samples [28]

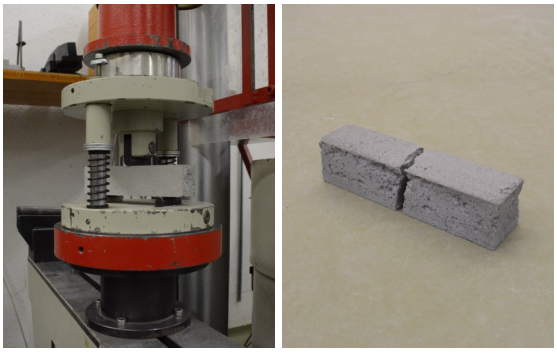
Sample	f_{mq} [MPa]	f_m [MPa]
K1	4.71	15.36
		14.17
K2	4.77	15.48
		17.12
K3	4.42	15.25
		14.27
Average	4.63	14.90
Std. dev.	0.19	0.63

Sample	f_{mq} [MPa]	f_m [MPa]
S1	1.09	2.73
		2.65
S2	0.96	2.86
		2.95
S3	1.12	2.65
		2.43
Average	1.06	2.77
Std. dev.	0.09	0.13

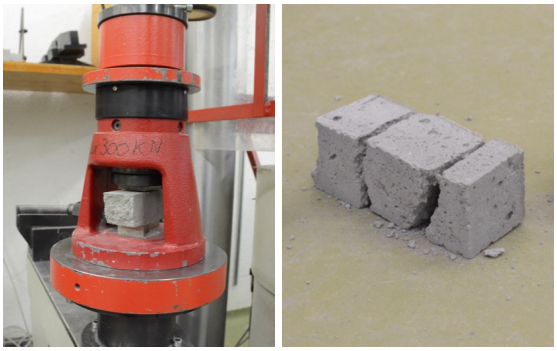
masonry used for the Hybrid Simulation tests. The test was conducted in force control mode until failure, with a ramp rate of 27 kN/min [28]. The displacements on the specimen surface were measured to determine the strains arising during the compression test by one horizontal and two vertical linear variable displacement transducers (LVDTs) mounted on each side of the specimen. The test setup is represented in **Figure 4.5**, whereas the results of the compression test are summarized in **Table 4.4**. Finally, **Table 4.5** presents a summary of all material properties.

4.2.2 Hybrid simulations

The PSD-HS architecture setup used in the case study is shown in **Figure 4.6**. In detail, a steel beam interfaces three servo-hydraulic actuators of 1 MN capacity each to the wall specimen, while 11 LVDTs are used to measure three types of displacements: sliding, uplift and vertical deformations. Furthermore, during the tests, a DIC system (NIKON D810 digital camera - 50 mm lens) acquires the in-plane displacements of the wall surface prior painted with a random speckle

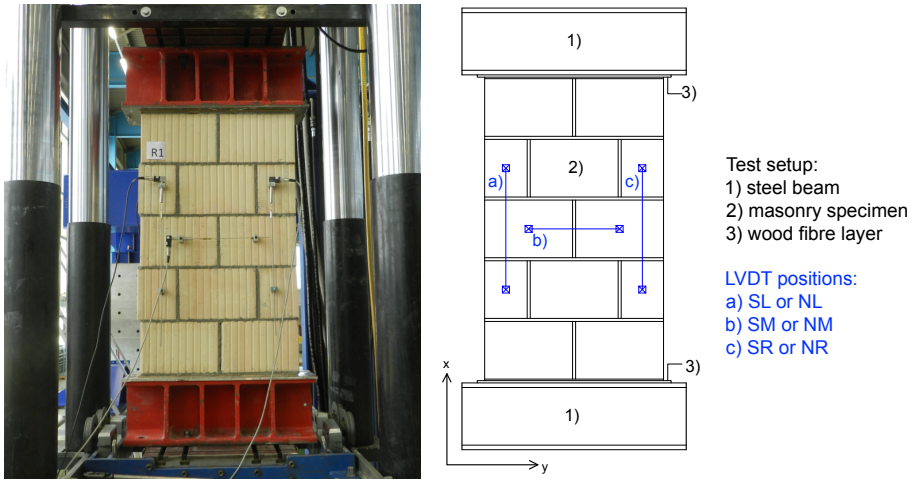


(a) Bending test setup (b) Bending failure



(c) Compression test setup (d) Compression failure

Figure 4.4: Bending and compressive strength testing [28]



(a) Specimen inserted into the test device (b) Schematic representation of the test setup with LVDT positions

Figure 4.5: Test setup for compression test of masonry specimens [28]

Table 4.4: Compression test results of masonry specimens [28]

Specimens	F_{max} [kN]	f_{xk} [MPa]	E_{xk} [GPa]
R1	460.47	5.20	4.51
R2	421.89	4.77	4.01
R3	482.69	5.45	4.10
Average	455.02	5.14	4.21

Table 4.5: Material properties [28]

Material	f_{mq} [MPa]	f_{bk} [MPa]	E_{xk} [MPa]
Brick	–	24.1	–
Mortar (K)	4.63	14.9	–
Mortar (S)	1.06	2.77	–
Masonry	–	5.14	4.21

pattern, shooting planar black & white pictures every 2 sec (**Figure 4.7**).

In order to perform PSD-HS of the system, the actuator setup was interfaced to an INDEL GIN-SAM4 real-time computer via Ether CAT. Before starting the PSD-HS experiment, a vertical load ramp was imposed to the specimen up to 208 kN corresponding to the nominal vertical load. Then, the INDEL GIN-SAM4 real-time computer executes the time integration algorithm, sends actuator displacement commands to the INOVA controller and reads corresponding feedback forces measured with loads cells at each time step of the simulation [53]. Detailed description of the time integration algorithm used in this testing campaign can be found in [13] and [57].

In **Table 4.6** the experimental test campaign of the six Hybrid Simulations is summarized. A record of the Montenegro earthquake (1979) was selected from the PEER Ground Motion Database (PEER, 2018), as seismic excitation and scaled to different values of PGA (**Figure 4.8**). The so determined accelerograms were fed into the equations of motion as seismic excitation, thus defining the displacement path of the horizontal actuator (horizontal loading under displacement control conditions).

The first two experiments, Test #1a and #1b, were conducted considering a small value of PGA to guarantee a linear response of the PS. The difference between these two tests was in the testing time scale. They were carried out in order to chose which was the best suited prototype structure to prevent dynamic instability due to experimental errors. So that, all following experiments were conducted on SM1 structures (4-DoFs NS and a $\lambda = 200$) [53]. During Test #2 a linear response was observed, while slightly nonlinear responses characterized Tests #3 and #4 although damage accumulation was very small. Detailed description of the tests can be found in [53]. The focus of this thesis work relies on Test #5 specifically, which was stopped earlier (after approximately 2.5 s of simulation time, i.e. 500 s

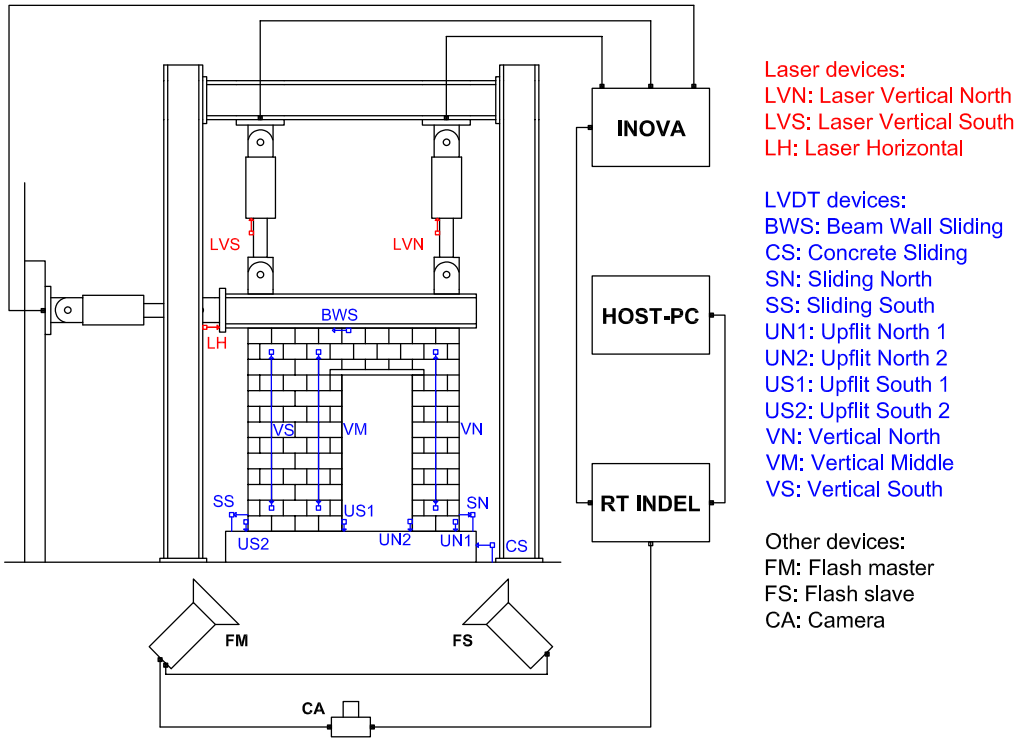
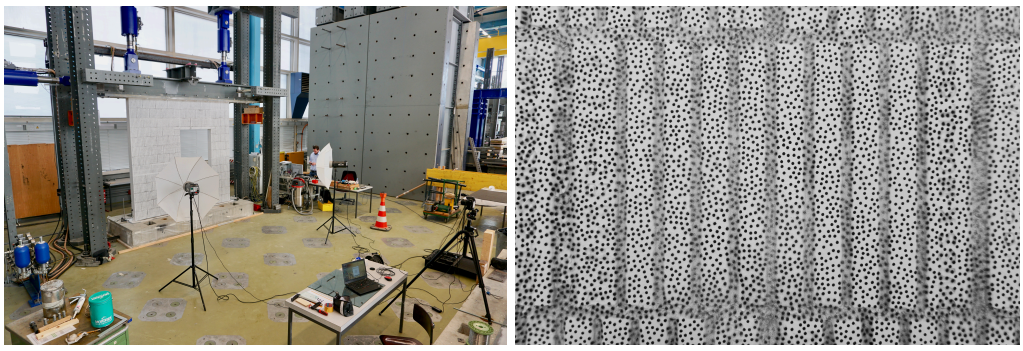


Figure 4.6: Architecture of the PSD-HS setup [28]



(a) DIC installations (b) Pattern applied onto test specimen

Figure 4.7: Test setup for DIC analysis [28]

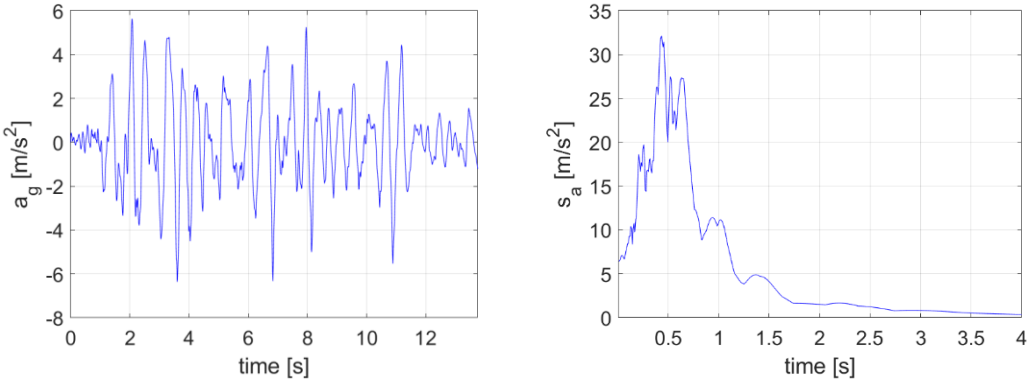


Figure 4.8: 1979 Montenegro earthquake

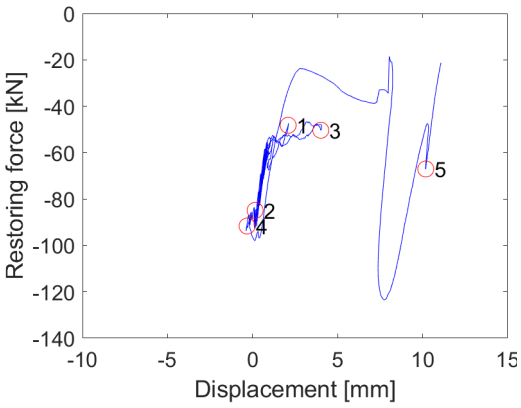
Table 4.6: Test program [53]

ID	PGA $[m/s^2]$	λ	Prototype structure	freq. b.w. [Hz]
#1a	0.45	500	SM2 (7-DoFs NS)	$0 \div 0.55$
#1b	0.45	200	SM2 (7-DoFs NS)	$0 \div 1.38$
#2	1.82	200	SM1 (4-DoFs NS)	$0 \div 0.56$
#3	3.18	200	SM1 (4-DoFs NS)	$0 \div 0.56$
#4	3.18	200	SM1 (4-DoFs NS)	$0 \div 0.56$
#5	6.36	200	SM1 (4-DoFs NS)	$0 \div 0.56$

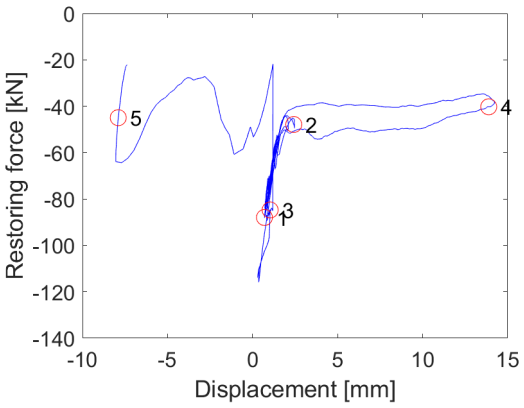
of wall-clock time) than the duration of the ground motion owing to the collapse of the wall specimen.

Figure 4.9 depicts the hysteresis loops of the horizontal and vertical restoring forces measured by the actuators. Five milestones (T1, T2, T3, T4, T5) are pointed out on the figure referring to specific evolution of the specimen collapse. While in **Figure 4.10** are shown the displacements recorded during the collapse test. It is worth to underlying as well that all plots refer to simulation time, which corresponds to wall-clock time divided by testing time scale.

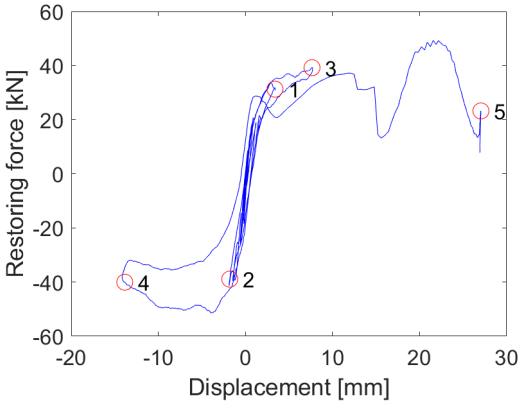
For each milestone, **Figure 4.11** depicts the Von Mises strain field measured via DIC. At T1 (about 1.2 s), von Mises strain concentrates at the lower mortar joint of the left wall bay and along a diagonal path following the mortar joints of the upper left part of the wall starting from the upper left corner of the opening. Such von Mises strain concentrations indicate joint opening, which allows relative rocking between wall subparts. Between T1 and T2 (about 1.45 s), the wall experience horizontal loading reversal, the lower left mortar joint closes and von Mises strain concentrations arise at both the lower and the upper levels of the thinner right wall bay. At T3 (about 1.75 s), remarkable von Mises strain concentrations are visible on both left and right lower mortar joints as well as along a diagonal path that connects the upper left corner of the opening to the upper mortar joint. At this point, joint opening allows relative rocking of three facade blocks namely, left and right bays and the spandrel. Suddenly, at T4 (about 2.2 s), the thinner wall bay spits at the level of the upper mortar joint and detaches from the spandrel,



(a) Vertical south actuators



(b) Vertical north actuators

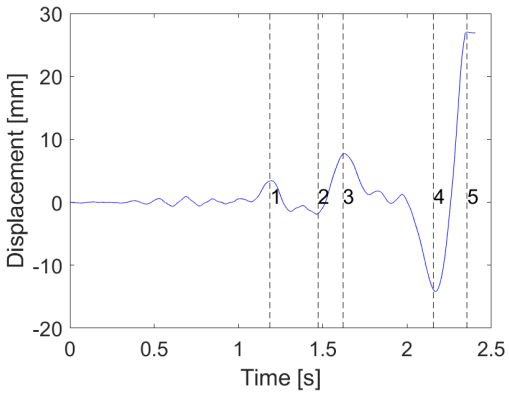


(c) Horizontal south actuators

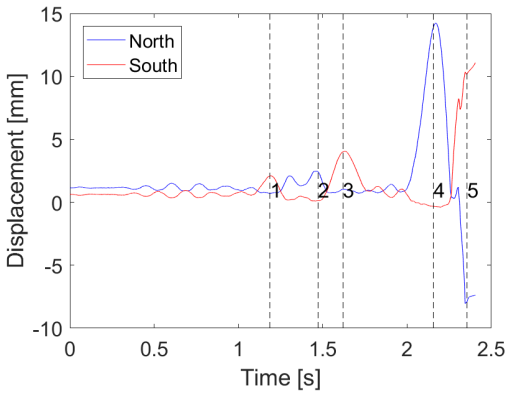
Figure 4.9: Restoring forces hysteresis loops measured during Test #5

which starts uplifting. The right edge of the spandrel rotates in clock-wise between T4 and T5 (about 2.4s) and impacts the thinner wall bay, which crashes under compressive load as testified by the large diagonal crack visible at the end of the experiment. The test stops immediately afterwards [53].

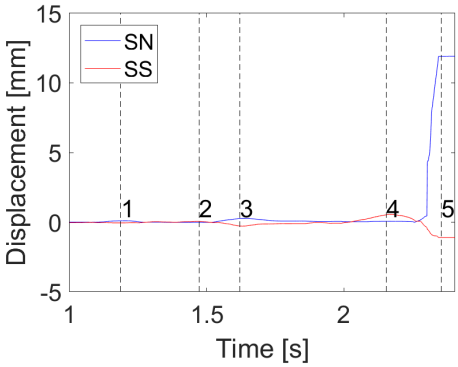
Finally, **Figure 4.12** shows the specimen after Test #5 and a close-up view on critical regions where damage concentrated.



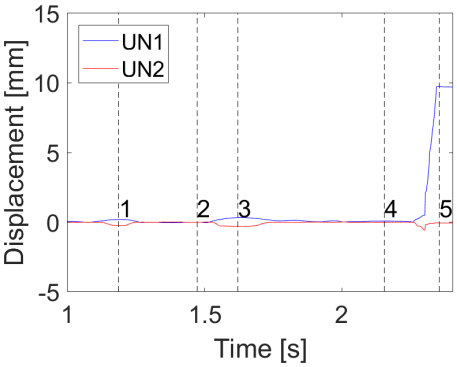
(a) Horizontal displacement



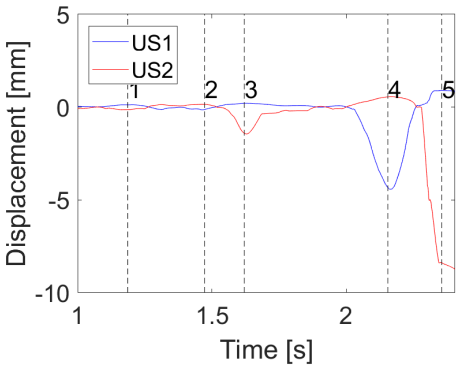
(b) Vertical displacement



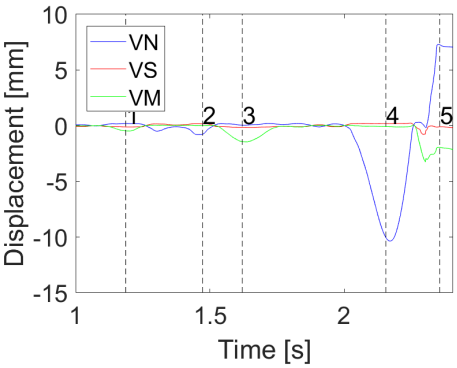
(c) LVDT displacements (SS-SN)



(d) LVDT displacements (UN1-UN2)

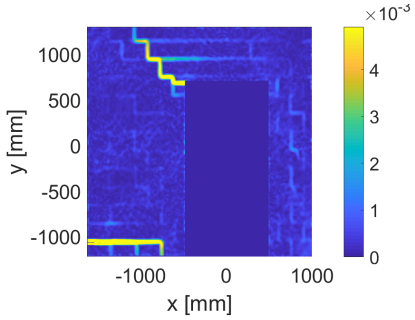


(e) LVDT displacements (US1-US2)

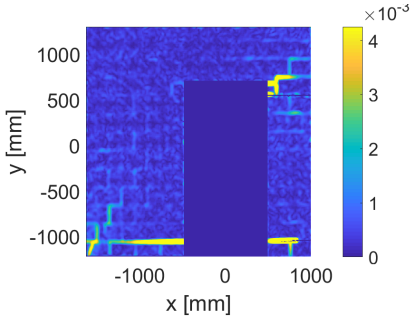


(f) LVDT displacements (VN-VS-VM)

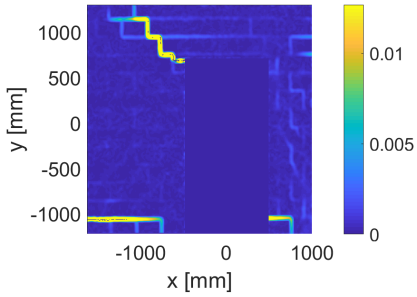
Figure 4.10: Displacement responses during Test #5



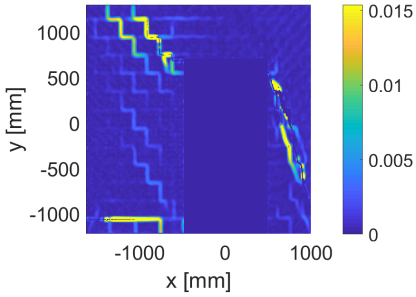
(a) Milestone T1



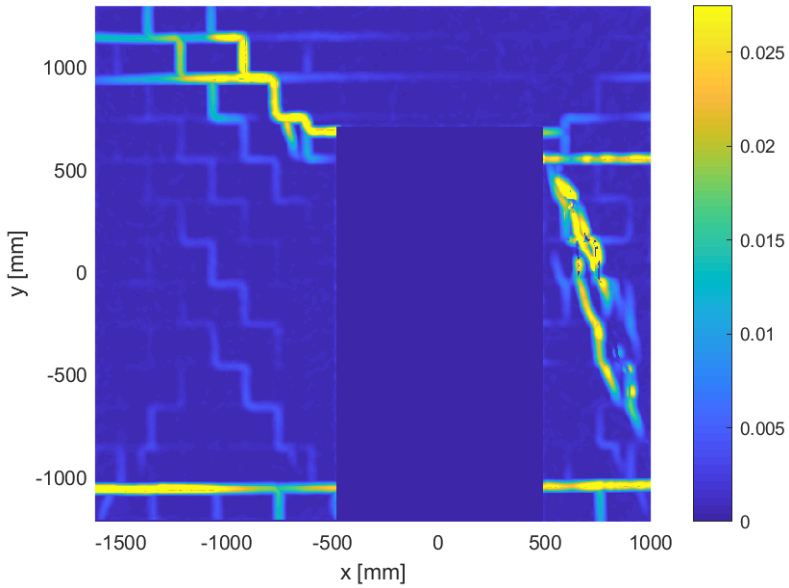
(b) Milestone T2



(c) Milestone T3



(d) Milestone T4



(e) Milestone T5

Figure 4.11: Von Mises strain field measured via DIC during Test #5

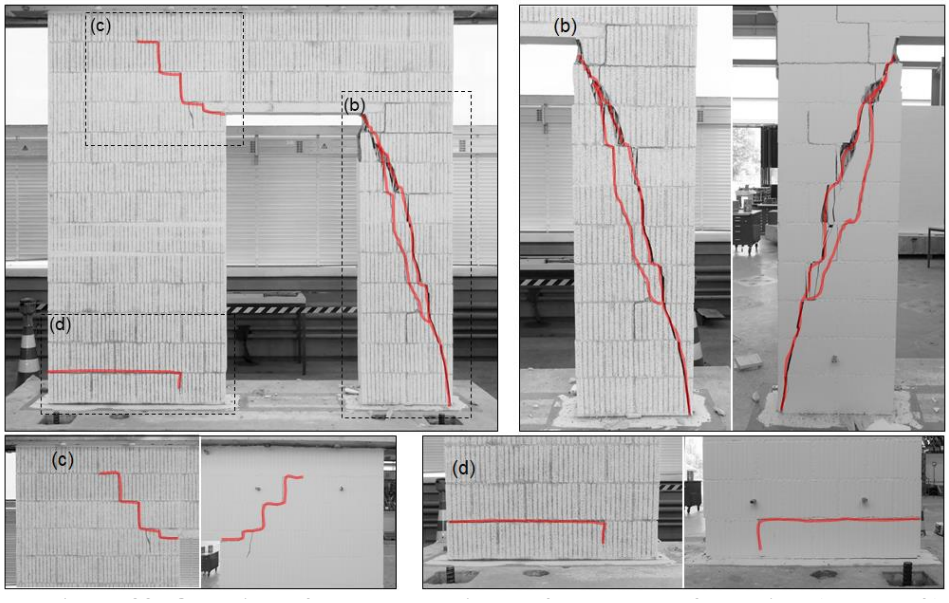


Figure 4.12: Overview of the wall specimen after Test #5

Chapter 5

NARX model for the masonry facade case study

Despite the high generality of surrogate modelling and Bayesian inversion, when such techniques are applied to data coming from experimental tests, additional effort in the posedness of the problem is needed. For this reason this chapter focuses on the definition of best regressors suitable for the case study introduced in Chapter 4. In Section 5.1 an holonomic model for mortar joints is briefly introduced theoretically. Then the FE model of the masonry structure and its parameterization is presented pointing out how model parameters variability influence the model response (Section 5.2). Finally, in Section 5.3 a set of NARX basis terms are proposed as best regressor suitable for the underlying problem.

5.1 Cohesive Zone Models for mortar joints in masonry

Cohesive Zone (CZ) models are useful tools to describe mixed-mode separation and mixed-mode over-closure phenomena in structural systems. A typical separation/ over-closure that need to be described when working with civil structures is the cracks opening and closure. Despite that every type of building can experience fracture mechanisms in its components, masonry structures are very incline to this type of nonlinear phenomenon. In recent years several researchers [52], proposed the use of holonomic relationships (i.e. laws that express the stress field as function of jump of displacements) for mortar joints reduced to interfaces. In their paper, Milani and Bertolesi [52], make use of an improved version of the potential-based model (i.e. that admits potential function) proposed by Xu-Needleman [73], to describe the behaviour of mortar joints under mixed-mode (i.e. related to mode 1 and mode 2 of fracture) loading conditions. In this work the authors used a linear elastic law in over-closure. Despite the powerful of potential-based models, such as the Xu-Needleman model, it is demonstrated that derivation of traction-separation relationships from a potential function can result in non-physical repulsive normal tractions and instantaneous negative incremental energy dissipation under displacement controlled monotonic mixed-mode separation when the work of tangential separation exceeds the work of normal separation [51]. For this reason, three non-potential CZ models, named NP1, NP2 and SMC, have been derived

and implemented in a Finite Elements (FE) framework in the two papers, [51] and [43]. The NP1 model has been proposed to fully eliminate the problem of repulsive normal tractions. However, using this model a reduction in the total work is obtained for small values of separations when an interface first undergoes pure normal separation and then a complete tangential separation. This unphysical behaviour is due to a rapid decrease in tangential work as the separation increases from 0. Another minor drawback of the NP1 model is that it uses different forms to describe mode I and mode II separation (mode I has a linear term in the exponent, while mode II has a quadratic term in the exponent), thus no model parameters can be chosen so that identical mode I and mode II traction-separation relationships are obtained, making the model inappropriate to describe the wide range of behaviours that may be encountered when working with real materials. To overcome this limitation the NP2 was proposed. NP2 introduces two additional coupling parameters, α and β, that weight the mixed-mode coupling terms in the traction-separation/over-closure laws. Despite this modification, the NP2 model provide unphysical behaviours for high values of separations. In fact, in this case the total work needed to separate two surfaces is not monotonic and after a monotonic increase it start to decrease to reach a horizontal asymptote as the separation tend to infinity. Despite the problems in separation, the NP2 model provides a physical realistic mixed-mode over-closure representation (such as NP1), maintaining greater generality than NP1. The third CZ model, named SMC overcomes the problems of both NP1 and NP2 in separation, providing a physical representations of the traction-separation phenomenon. However, the model cannot be used in over-closure as it provides identical traction-separation behaviour in over-closure and in separation, so that the closure phenomenon is not penalized, and unphysical interpenetrations can occur. In the paper, [51], it is suggested that in cases where both separation and over-closure may be encountered the SMC model could be used in separation with the NP2 formulation being used in over-closure. For the authors this scheme is computationally attractive as the NP2 and SMC CZ models have an identical form of the mode II traction-separation relationship. In addition NP2, if used just in over-closure, can cover a wide range of behaviours than NP1 maintaining at the same time a physical meaning. For the present study, the NP2 and SMC laws have been implemented in a home-made FE software developed in Matlab. The laws (written in terms of force and displacement instead of stress and displacement), are used to define a bar element with two Degree of Freedoms (DoFs) for each node, i.e. the axial and tangential displacement of the bar. The NP2 model is used in over-closure, while the SMC model is used in separation. It is worth mentioning that the two models provide the same initial (i.e. when no separation, over-closure or sliding occur) tangent stiffness. The tangent stiffness has been derived analytically and implemented with the traction-separation/over-closure laws in the FE software. The implemented laws are reported hereinafter:

- NP2 if $\Delta_n < 0$:

$$T_n(\Delta_n, \Delta_t) = \sigma_{max} \exp(1) \left(\frac{\Delta_n}{\delta_n} \right) \exp\left(- \frac{\Delta_n}{\delta_n} \right) \exp\left(\alpha \sqrt{\frac{\Delta_t^2}{\delta_t^2}} \right) \tag{5.1}$$

$$T_n(\Delta_n, \Delta_t) = \tau_{max} \exp(1) \left(\frac{\Delta_t}{\delta_t} \right) \exp \left(- \sqrt{\frac{\Delta_t^2}{\delta_t^2}} \right) \exp \left(- \beta \frac{\Delta_n}{\delta_n} \right) \tag{5.2}$$

- SMC if $\Delta_n \geqslant 0$:

$$T_n(\Delta_n, \Delta_t) = \sigma_{max} \exp(1) \left(\frac{\Delta_n}{\delta_n} \right) \exp \left(- \sqrt{\frac{\Delta_n^2}{\delta_n^2} + \frac{\Delta_t^2}{\delta_t^2}} \right) \tag{5.3}$$

$$T_n(\Delta_n, \Delta_t) = \tau_{max} \exp(1) \left(\frac{\Delta_t}{\delta_t} \right) \exp \left(- \sqrt{\frac{\Delta_n^2}{\delta_n^2} + \frac{\Delta_t^2}{\delta_t^2}} \right) \tag{5.4}$$

where:

- σ_{max} is the tensile stress strength without sliding, unit of [force/area];
- τ_{max} is the shear stress strength without separation, unit of [force/area];
- $\delta_n = \phi_n / (\sigma_{max}\, exp(1))$ is the characteristic length for separation, unit of [length];
- $\delta_t = \phi_t / (\tau_{max}\, (0.5\, exp(1))^{0.5})$ is the characteristic length for separation, unit of [length];
- ϕ_n is the work of separation, unit of [force/length];
- ϕ_t is the work of sliding, unit of [force/length];
- $\alpha = \beta = \sqrt{2} - 1$ are dimensionless coupling parameters imposed at this specific value to ensure identical normal and tangential components of the traction vector for 45 mixed-mode separation.

The laws have been multiplied for the area, A, of the bar element to obtain a force-displacement domain: $N = T_n\, A$, $T = T_t\, A$. The FE implementation has been validated by comparing the results of a benchmark FE analysis with the reference values obtained from the paper [51]. For the validation, a unit area of the bar has been used. The reference values have been calculated with the WebPlotDigitizer application (https://automeris.io/WebPlotDigitizer/). The results of the validation are reported **Figure 5.1**.

5.2 FE model parameterization

The PS of the case study described in CHAPTER 4 has been modeled in a homemade FE software developed in Matlab. Its discretization (**Figure 5.2**) has been accurately selected to best-reproduce the crack patterns experimented in the laboratory tests. Specifically, it is constitutes by 15 linear plane-brick elements, 34 nonlinear beam-mortar elements following the laws seen in SECTION 5.1, and 7 beam-steel elements to model the steel beam used in the laboratory tests, for a total of DOF of 120.

In order to calibrate the model, a preliminary parameterization was carried out. Namely, the horizontal and vertical displacement response of two nodes of

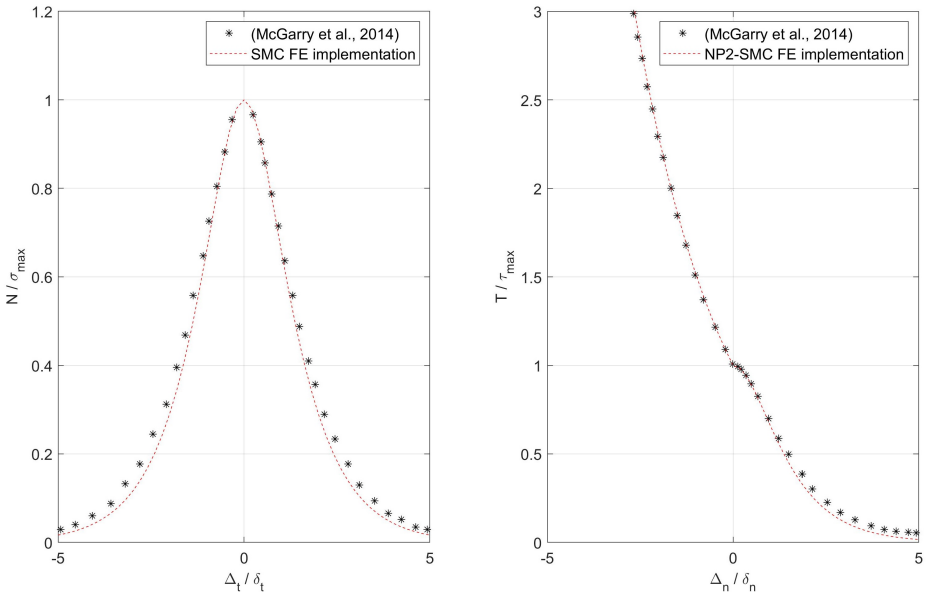


Figure 5.1: Validation implementation

the structure (N44 and N55) have been evaluated. These two nodes corresponds to the monitored points in the experimental setup, whose responses were recorded in the laboratory. The model parameterization turned out to be necessary for two reasons mainly: first, in order to get, as much as possible, prior informations on how the response of the system changes in function of the parameters; secondly, to get insight on numerical instability arising in the FE time-history response output. The latter is experimented on the divergence of the nodes displacements at different instant of time when changing the input parameters (the experimental divergence occurs at $t = 2.41$ sec, see **Figure 4.10**). To handle these, a data-set of size 1×10^4 has been constructed using LHS. The McGarry model parameters, which constitute the vector of uncertain parameters $\boldsymbol{x}_{\mathcal{M}} = \{\phi_n, \sigma_{max}, \phi_t, \tau_{max}\}$, are considered independent random variables with associated uniform distributions given in **Table 5.1**. In detail, the support on σ_{max} has been chosen on the evidence of experimental tests results (SECTION 4.2.1), while the support on τ_{max} was chosen based on reasonable values of engineering interest found in literature. Finally, bigger supports have been set to the model parameters ϕ_n and ϕ_t respectively, due to the lack of a consistent knowledge on their actual values.

Table 5.1: McGarry model parameter distributions

Parameter	Distribution	Support
ϕ_n (N/m)	Uniform	$[1, 10] \times 10^3$
σ_{max} (N/m^2)	Uniform	$[0.87, 1.23] \times 10^6$
ϕ_t (N/m)	Uniform	$[0.5, 8] \times 10^3$
τ_{max} (N/m^2)	Uniform	$[0.3, 3.7] \times 10^6$

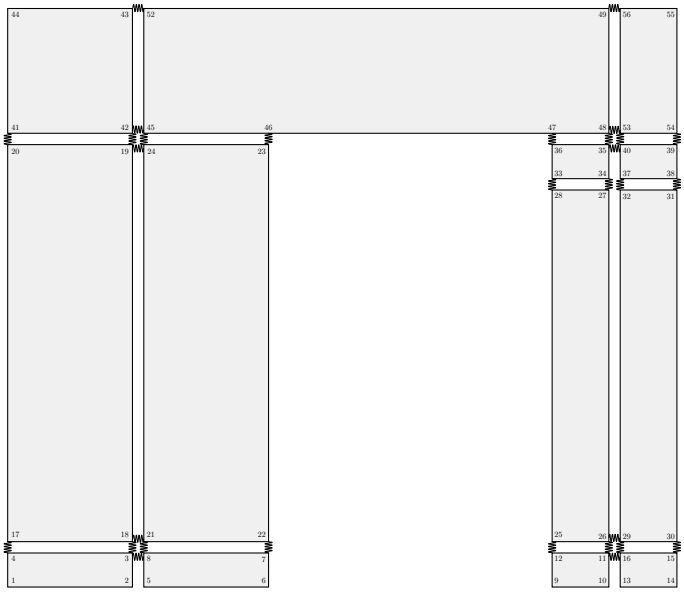


Figure 5.2: FE discretization

Each realization of the data-set corresponds to a FE forward model call that means running the nonlinear dynamical analysis through Newmark method with a tolerance of $\varepsilon = 1 \times 10^{-10}$ and iterations limit set to 20. The results are plotted below in function of the parameters and the instant of time when instability occurs. **Figure 5.3 (e)** clearly shows how the numerical instability occurs systematically for low values of the resistance parameters, namely for $\sigma_{max} < 1 \times 10^6$ N/m^2 and $\tau_{max} < 1.5 \times 10^6$ N/m^2. Noted that, a filter on these parameters was applied on the data-set in order to get insight on how they affect the instability issue. The results of this filtering are shown in **Figure 5.4**. Particularly, **Figure 5.4 (e)** shown how the instability disappears when $\sigma_{max} > 1 \times 10^6$ N/m^2 and $\tau_{max} > 1.5 \times 10^6$ N/m^2. In addition, some conclusion on parameters ϕ_n and ϕ_t values can be achieved by the examination of **Figure 5.4 (a)**. Note that exceeding values of ϕ_n, specifically $\phi_n < 3.2 \times 10^3$ and $\phi_n > 6.8 \times 10^3$, affect the stability of the model markedly. On the contrary, the parameter ϕ_t appears affecting the stability noway.

The parameterization required $10\times^4$ FE forward model call. Each single FE call is time-computationally quite expensive and requires about 40 sec for run. The computation of a such huge data-set, was possible parallelizing the forward model calls on multi-clusters service furnished by HPC@Polito [59]. The total time-cost necessary to build the aforementioned data-set was approximately of some days. In the view of a future calibration of the model parameters within the exposed Bayesian paradigm, it appears clear as conducting a proper Bayesian inference results not feasible using the FEM as the forward model. Indeed the total forward model calls of a MCMC simulation are significantly huger. With this in mind the aim of next final section is to find a proper basis for the NARX model for the case study.

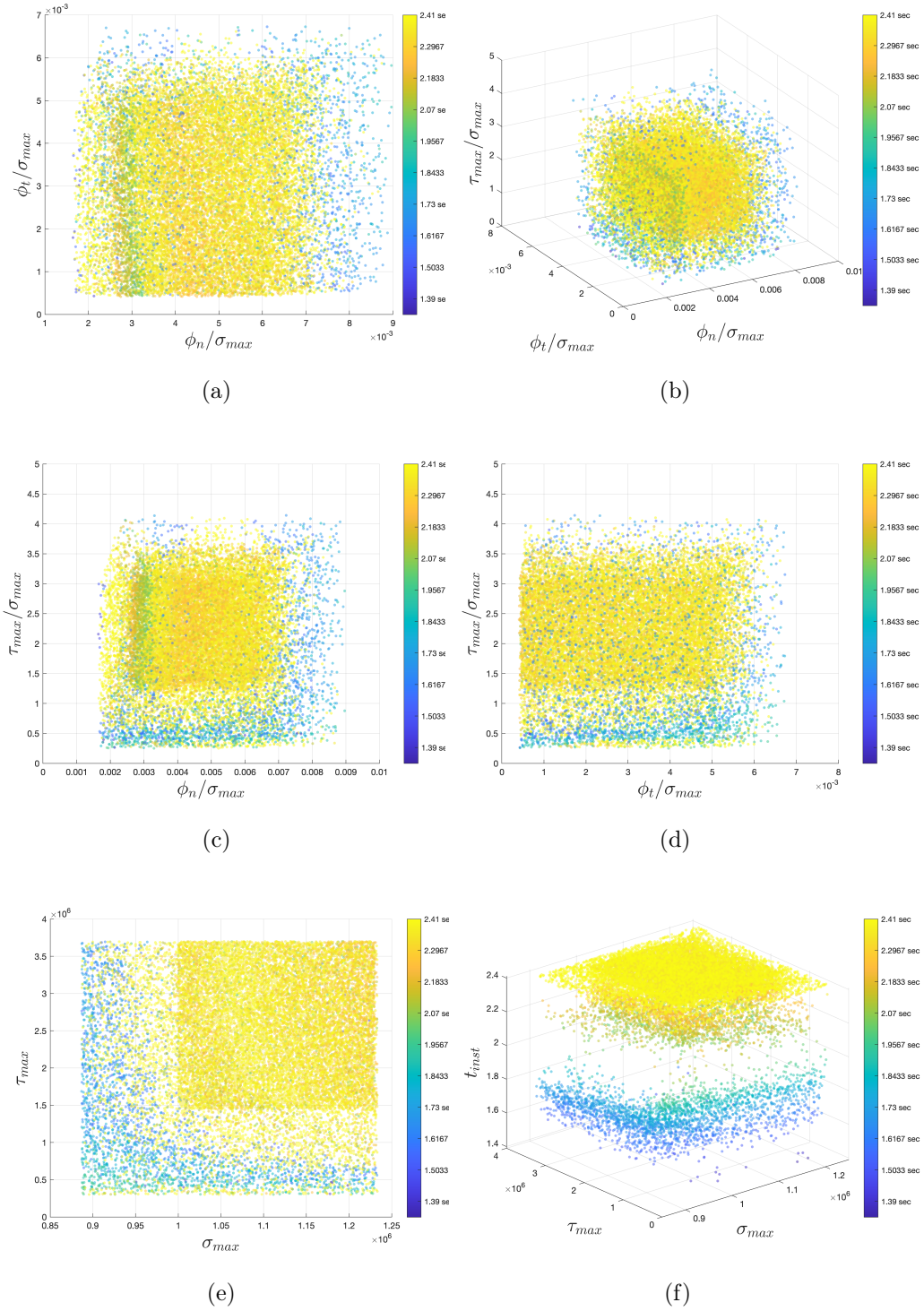


Figure 5.3: Model parameterization

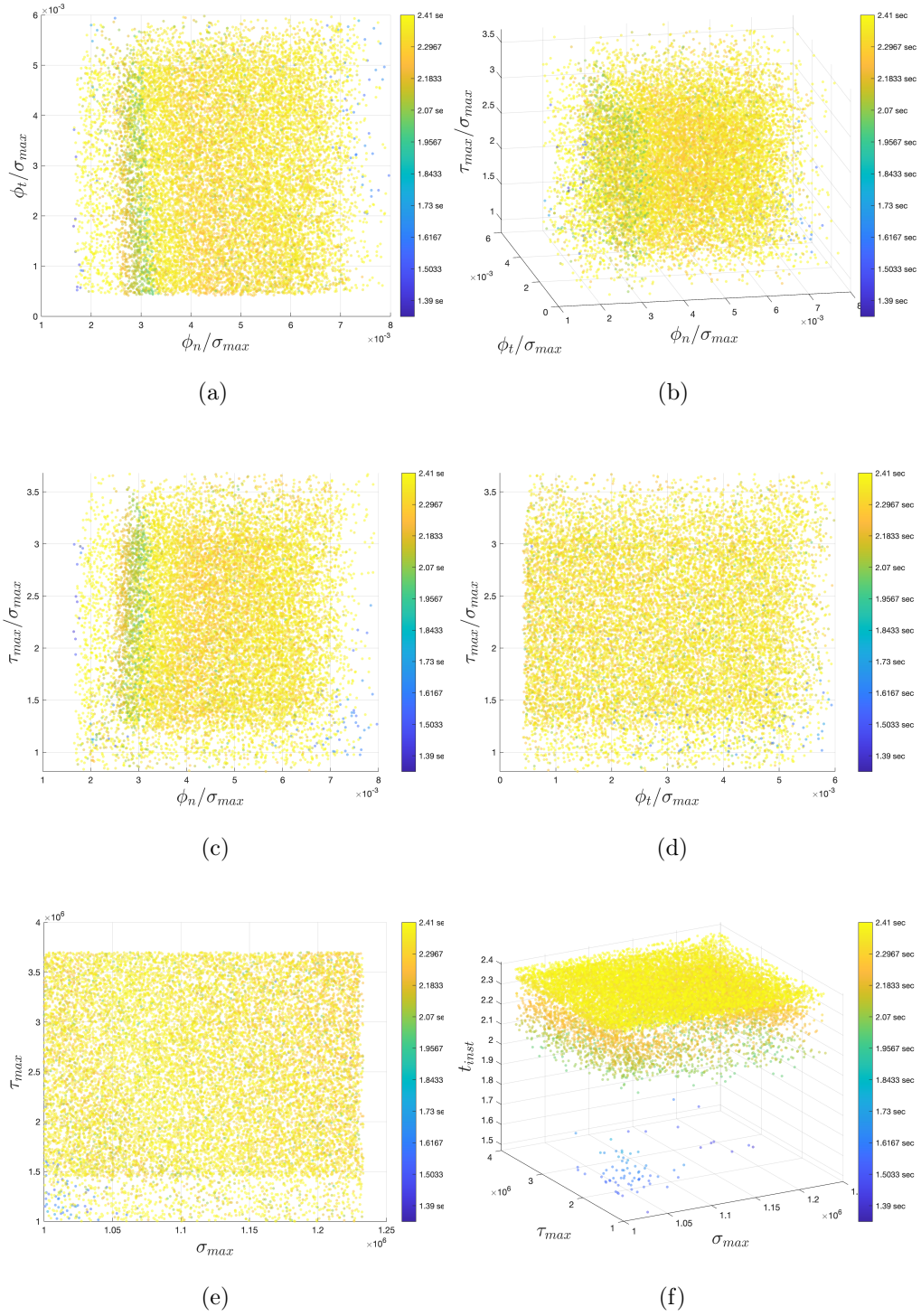


Figure 5.4: Model parameterization on filtered data-set

5.3 NARX model

As seen in CHAPTER 2, NARX models allow to build the the expansion of the model $\mathcal{M}$ using the observed data of the input and output signals (2.50). The goal is now to find a NARX model for each monitored node (N44 and N55) of the case study structure. This time, the nonlinear function to be identified $\mathcal{F}(\cdot)$, is chosen as function of:

$$\boldsymbol{z}(t) = \left\{x(t), ..., x(t-n_x),\ y^{(j)}(t-1), ..., y^{(j)}(t-n_y)\right\}^{\intercal} \tag{5.5}$$

where the index $j = \{k, l, m\}$ stands for j-th node response. So that, the existing correlation in the response between nodes can be taken into account. In detail, k refers to the response of the node N44 in x direction, m refers to the response of the node N44 in y direction, while m refers to the response of the node N55 in y direction.

To keep things simple, also in this case polynomial basis function have been used to construct the mapping $\mathcal{F}(\cdot)$:

$$y^{(j)}(t) = \sum_{i=1}^{n_g^{(j)}} \vartheta_i^{(j)} g_i^{(j)}\big(\boldsymbol{z}(t)\big) + \varepsilon^{(j)}(t) \tag{5.6}$$

To build each NARX model for representing the displacement time-histories $u(t)$ and $v(t)$ of the masonry system, first 1500 samples of the input parameters were generated by LHS and 1500 corresponding FE model simulations were conducted. These data-set was then reduced rejecting all realization that generated a $t_{inst} \leqslant$ 2.36 sec. The so built data-set of size 200 constitutes the Experimental Design used to select the NARX models.

For this purpose NARX model structures based on Prony's series were adopted. These series were chosen for their capacity to represent damped complex exponentials or sinusoids. In detail, the basis terms are defined by:

$$g_i(t) = x(t-k)^l \tag{5.7}$$

$$g_i(t) = e^{x(t-k)} \tag{5.8}$$

$$g_i(t) = y^{(j)}(t-j); \tag{5.9}$$

$$g_i(t) = x(t-k)\, e^{y^{(j)}(t-1)}\, |y^{(j)}(t-j)|; \tag{5.10}$$

$$g_i(t) = y^{(j)}(t-j)\, e^{t\, x(t-k)}\, sin(2\pi\, y^{(j)}(t-j)\, t + \tau); \tag{5.11}$$

$$g_i(t) = x(t-k)\, e^{2\,\sigma_{max}\, t}\, cos(2\pi\, t + \tau); \tag{5.12}$$

with $l = 0, 1, 2$, $k = 0, ..., n_x$, $j = 1, ..., n_y$, and $n_x = 10$, $n_y = 5$. So that the initial full NARX model contains totally 59 terms.

Next, the candidate NARX models were computed. For this purpose, LARS was applied to the initial full NARX model for each experiment of the ED, to select the most relevant terms for each node. This lead to have 313 NARX candidates in

total. OLS (eq. (2.62)) was used then to determine the NARX coefficients $\vartheta_i^{(j)}$ for all the simulations. Subsequent to the evaluation of the accuracy of each NARX candidates (eq. (2.62)), the most appropriates NARX models with mean relative error of $\bar{\varepsilon}^{(k)} = 1.28 \times 10^{-1}$, $\bar{\varepsilon}^{(l)} = 1.79 \times 10^{-1}$, $\bar{\varepsilon}^{(m)} = 7.87 \times 10^{-1}$ over 200 experiment, were selected among the candidates.

Figure 5.5, 5.6, 5.7 depicts the cross-validation plots of the displacements predicted by the NARX models against the values obtained with the FEM.

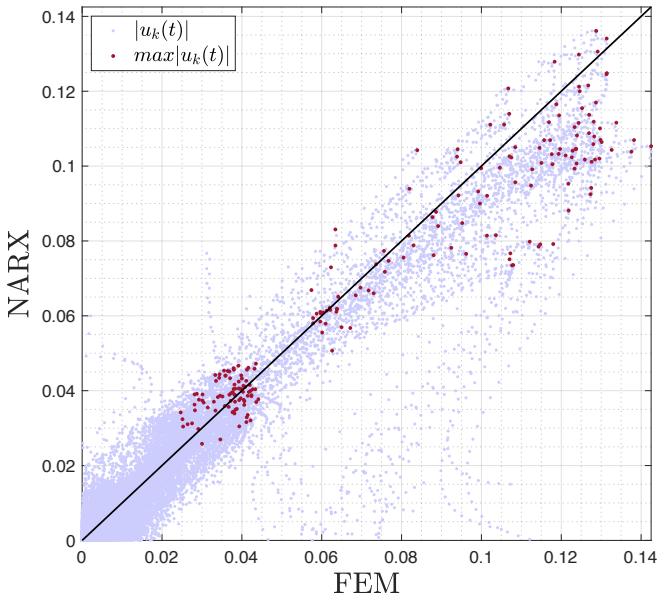


Figure 5.5: NARX$^{(k)}$ free-run-reconstruction of the entire ED

In **Figure 5.8, 5.9, 5.10**, as an example, are plotted the NARX predictions for three experiments of the ED ($k = 129, k = 132, k = 211$).

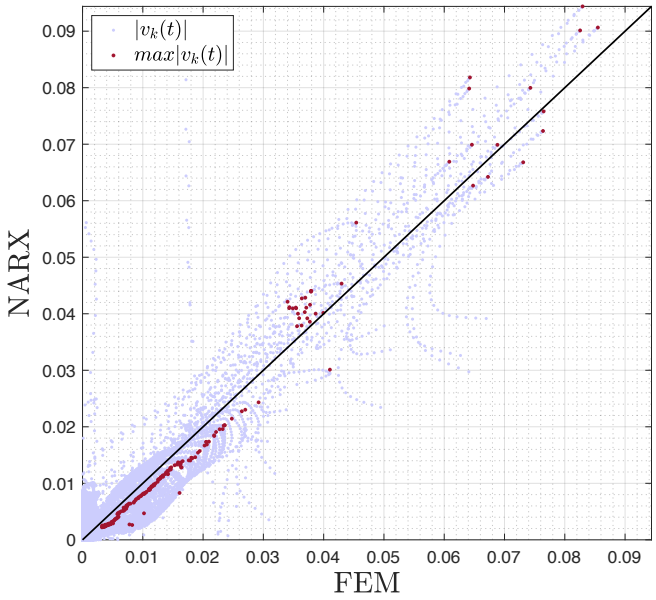


Figure 5.6: NARX$^{(l)}$ free-run-reconstruction of the entire ED

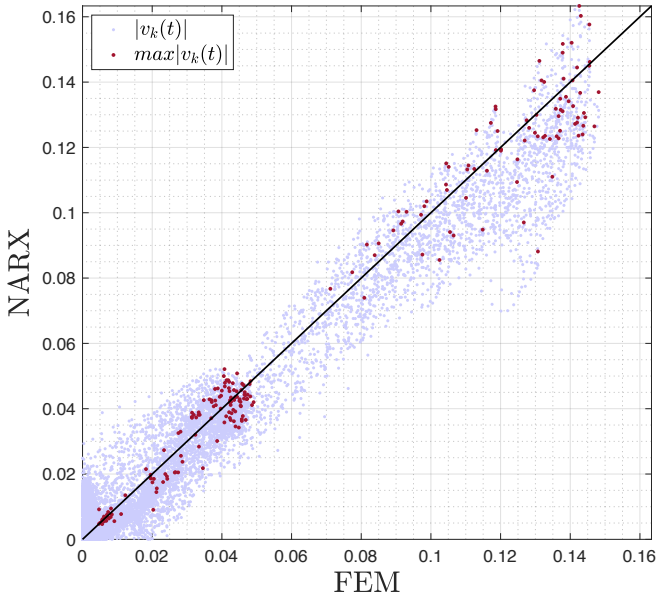


Figure 5.7: NARX$^{(m)}$ free-run-reconstruction of the entire ED

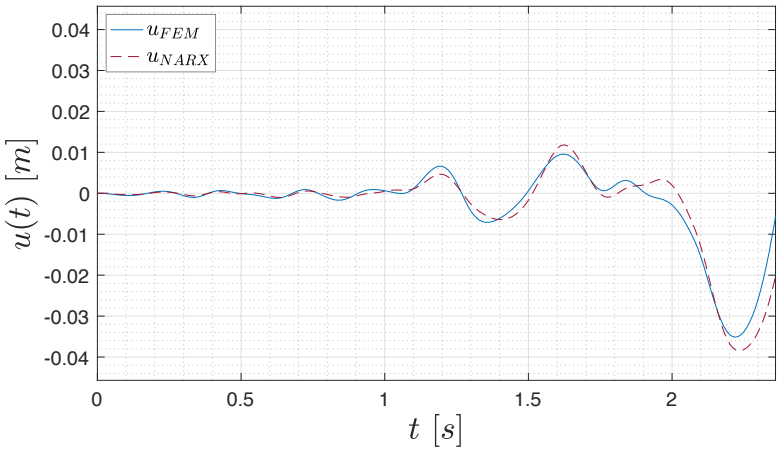


Figure 5.8: NARX(k) free-run-reconstruction of N44 in x-direction

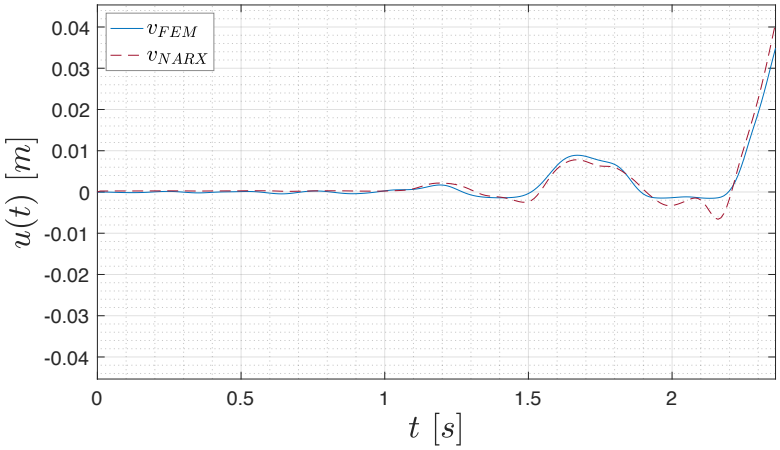


Figure 5.9: NARX(l) free-run-reconstruction of N44 in y-direction

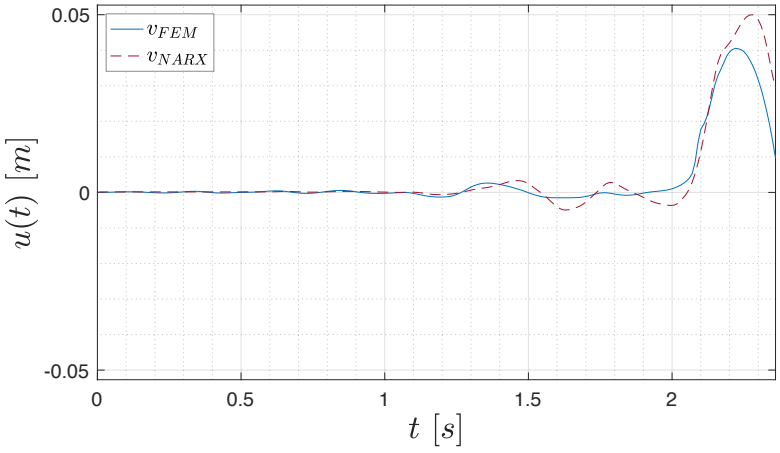


Figure 5.10: NARX(m) free-run-reconstruction of N55 in y-direction

Conclusions and outlooks

The present thesis aimed at developing a Bayesian based approach to solve structural systems with uncertain governing parameters subject to stochastic seismic excitations. The proposed method allows to rigorously deal with model uncertainties and experimental errors intrinsic in any system identification method considering the problem from a statistical perspective.

The capability of the proposed Bayesian model inversion in uncertainty framework lies in providing probabilistic information of the estimated parameters and on their error, which can be useful at the moment of making decisions with respect to the selection of parameters and/or the assessment of mathematical models that simulate the nonlinear behaviour experienced by the system.

However, it was shown as the effectiveness of a Bayesian approach applied to earthquake engineering runs into time and computationally expansive issues. To overcame these difficulties surrogate modeling technique have been introduced to deal with structural systems with uncertain governing parameters subject to stochastic seismic excitations. This is a challenging task because one has to deal at once with two tough problems: nonlinear dynamics and uncertainty quantification. In the approach followed, the two aspects are treated separately. The increasingly strong nonlinearity in dynamics is taken into account by NARX models, namely by techniques able representing future responses as functions of its past values. Whereas the consideration of uncertainties is taken into account by surrogate models, specifically through high-degree polynomial chaos expansions (PCEs) and Gaussian process. The effectiveness of the proposed methods were demonstrated with numerical benchmark on nonlinear system.

Despite the high generality of surrogate modelling and Bayesian inversion, when such techniques are applied to data coming from experimental tests, additional effort in the posedness of the problem is needed. Indeed, the uncertainties associated with civil structures make the prediction of the actual mechanical characteristics a difficult task. For this reason, the aim of this thesis relies in the definition of the best NARX regressor suitable to the case study of a masonry wall subjected to dynamical.

To conclude, dealing with stochastically dynamical system is not an easy task. To get reliable estimates and insight on structural systems behaviour require additional effort.

Appendix

BoucWenClass.m

```
classdef BoucWenClass
% Bouc–Wen–Baber–Noori SDF system class
%
%  .Earthquake  =  sub–class containing earthquake data
%  .Parameters  =  sub–class containing system's parameters data
%  .u           =  displacement response of the system
%  .v           =  velocity response of the system
%  .z           =  hysteretic displacement response of the system
%  .eps         =  dissipated hysteretic energy of the system
%  .fr          =  hysteretic restoring force of the system
%
% See also  EarthquakeClass, ParametersClass, BWBN_accelerogram,
%           BWBN_linearValidation, BWBN_solve, BWBN_saveExperimentalData,
%           BWBN_print, BWBN_plot.

  properties
    Earthquake
    Parameters
    u
    v
    z
    eps
    fr
  end

  methods
  %%  DEFINE SUB–CLASSESs
      function obj = BoucWenClass
        obj.Earthquake = EarthquakeClass;
        obj.Parameters = ParametersClass;
      end
  %%  PLOT ACCELEROGRAM
      function BWBN_accelerogram(obj,~)
      % BWBN_accelerogram plot the input accelerogram.
      %
      % If .Earthquake.filter == 'YES' plots the filtered accelerogram.
      %
      %   BWBN_accelerogram(obj) plot the accelerogram.
      %   BWBN_accelerogram(obj,'save') plot and save the accelerogram.
      %
      % See also BoucWenClass

      %   Plot accelerogram
          %   Figure properties
              fig = figure;
              fig.Position          = [0, 0, 800, 450];
              fig.PaperUnits        = 'points';
              fig.PaperSize         = [800, 450];
              fig.PaperPositionMode = 'manual';
              fig.PaperPosition     = [0, 0, 800, 450];
          %   Plot data
              p1 = plot(obj.Earthquake.time,obj.Earthquake.ag);
          %   Plot properties
              p1.Color           = '#0072BD';
              p1.LineWidth       = 0.5;
              p1.LineStyle       = '–';
          %   Axes properties
              ax = gca;
              ax.FontSize = 15;
              % X axis Limits
              ax.XLim = [obj.Earthquake.time(1), obj.Earthquake.time(end)];
              % Y axis limits
              ax.YLim =...
```

```
64              [ -(max(abs(obj.Earthquake.ag)) + 0.15 * ...
65                                              max(abs(obj.Earthquake.ag))),...
66                  (max(abs(obj.Earthquake.ag)) + 0.15 * ...
67                                                max(abs(obj.Earthquake.ag)))];
68          %   Grid and Minorgrid
69              ax.XGrid      = 'on';
70              ax.YGrid      = 'on';
71              ax.XMinorGrid = 'on';
72              ax.YMinorGrid = 'on';
73          %   X label
74              ax.XLabel.String      = '$t$ $[s]$';
75              ax.XLabel.Interpreter = 'latex';
76              ax.XLabel.FontSize    = 26;
77          %   Y label
78              ax.YLabel.String      = '$a_g$ $[m/s^2]$';
79              ax.YLabel.Interpreter = 'latex';
80              ax.YLabel.FontSize    = 26;
81          %   Title
82              title(sprintf('%s ground motion record',obj.Earthquake.name))
83              ax.Title.Interpreter = 'latex';
84              ax.Title.FontSize    = 26;
85          %   Legend
86              if strcmp(obj.Earthquake.filter,'YES') == 1
87                  legend('$a_g(t)$')
88              else
89                  legend('$a_g(t)$')
90              end
91              ax.Legend.Interpreter = 'latex';
92              ax.Legend.FontSize  = 18;
93          %   Save graph
94              if nargin > 1
95                  FigureName =  sprintf('%s ground motion record', ...
96                                obj.Earthquake.name);
97                  for i = 1 : length(FigureName)
98                      if FigureName(i) ==' '
99                          FigureName(i) ='_';
100                     end
101                 end
102                 saveas(fig, fullfile('..', 'Matlab', 'Figures','pdf', ...
103                       FigureName),'pdf');
104                 saveas(fig, fullfile('..', 'Matlab', 'Figures','fig', ...
105                       FigureName),'fig');
106                 saveas(fig, fullfile('..', 'Matlab', 'Figures','tif', ...
107                       FigureName),'tiffn');
108                 saveas(fig, fullfile('..', 'Matlab', 'Figures','png', ...
109                      FigureName),'png');
110             end
111     end
112 %%  SYSTEM LINEAR VALIDATION IN ELASTIC FIELD
113     function BWBN_linearValidation (obj,~)
114     % BWBN_linearValidation validates the Bouc-Wen-Baber-Noori system in
115     % linear field.
116     %
117     %   BWBN_linearValidation(obj) prints and plots the result.
118     %   BWBN_linearValidation(obj,'save') prints, plots and saves the result.
119     %
120     % See also BoucWenClass
121
122     %   Display status
123         fprintf('Bouc-Wen-Baber-Noori system validation: ');
124     %   Solve Bouc-Wen-Baber-Noori ODEs system in linear field
125         [~,x] = ode45(@(t,x)solveODE(t,x,obj,'L'),obj.Earthquake.time,...
126                 [0 , 0 , 0 , 0, 0]);
127         uLinear  = x(:,1);
```

```
128         zLinear  = x(:,3);
129         frLinear = (obj.Parameters.alpha * obj.Parameters.w0^2 * uLinear +...
130                (1-obj.Parameters.alpha) * obj.Parameters.w0^2 * zLinear) *...
131                 obj.Parameters.mass * 1e-5;
132     %   Solve equivalent linear SDF system with convolution
133         ht       = (1 / obj.Parameters.wd) * (exp( - obj.Parameters.zita *...
134 
135                     obj.Parameters.w0 * obj.Earthquake.time)) .*...
136                     sin(obj.Parameters.wd * obj.Earthquake.time);
137         uConv    =   conv(ht, - obj.Earthquake.ag * 1000) * obj.Earthquake.ts;
138         tConv    =   0 : obj.Earthquake.ts : (length(uConv) - 1) * ...
139                     obj.Earthquake.ts;
140         toll     =   2; % [mm]
141         valid    =   ones(length(uLinear), 1);
142         for i = 1 : length(uLinear)
143             if abs(uConv(i) - uLinear(i)) <= toll
144               valid(i) = 0;
145             end
146         end
147     %   Display validation
148         if sum(valid) == 0
149           fprintf('YES\n');
150         else
151           fprintf('NO\n');
152         end
153     %   Plot linear validation
154         %   Figure properties
155             fig = figure;
156             fig.Position          = [0, 0, 1400, 375];
157             fig.PaperUnits        = 'points';
158             fig.PaperSize         = [1400, 375];
159             fig.PaperPosition     = [0, 0, 1400, 375];
160             fig.PaperPositionMode = 'manual';
161         %   Plot convolution
162             ax1 = subplot(1,2,1);
163                 p1 = plot(obj.Earthquake.time,uLinear);
164                 hold on
165                 p2 = plot(tConv,uConv);
166             %   Plot properties
167                 p1.Color     = '#0072BD';
168                 p1.LineWidth = 0.75;
169                 p1.LineStyle = '-';
170                 p2.Color     = '#D95319';
171                 p2.LineWidth = 0.75;
172                 p2.LineStyle = '--';
173             %   Axes properties
174                 ax1.FontSize = 18;
175             %   X axis limits
176                 ax1.XLim = [obj.Earthquake.time(1),obj.Earthquake.time(end)];
177             %   Y axis limits
178                 Ymax     = max(max(abs(uConv)), max(abs(uLinear)));
179                 ax1.YLim = [ - Ymax - 0.15 * Ymax, Ymax + 0.15 * Ymax];
180             %   Grid and Minorgrid
181                 ax1.XGrid      = 'on';
182                 ax1.YGrid      = 'on';
183                 ax1.XMinorGrid = 'on';
184                 ax1.YMinorGrid = 'on';
185             %   X label
186                 ax1.XLabel.String      = '$t$ $[s]$';
187                 ax1.XLabel.Interpreter = 'latex';
188                 ax1.XLabel.FontSize    = 30;
189             %   Y label
190                 ax1.YLabel.String      = '$u(t)$ $[mm]$';
191                 ax1.YLabel.Interpreter = 'latex';
```

```
192                 ax1.YLabel.FontSize    = 30;
193             %   Title
194                 title(sprintf('Model validation'));
195                 ax1.Title.Interpreter = 'latex';
196                 ax1.Title.FontSize    = 30;
197             %   Legend
198                 legend('$u_{BWBN}(t)$','$u_{conv}(t)$');
199                 ax1.Legend.Interpreter = 'latex';
200                 ax1.Legend.FontSize    = 20;
201         %   Plot restoring force
202             ax2 = subplot(1,2,2);
203                 XMax    = max(abs(uLinear));
204                 xleft   = - (XMax + 0.15 * XMax);
205                 xright  = xleft * - 1;
206                 ybottom = - (max(abs(frLinear)) + 0.15 * max(abs(frLinear)));
207                 ytop    =  ybottom * - 1;
208                 xaxis_center = [xleft, xright];
209                 yaxis_center = [ybottom, ytop];
210                 null_axis    = [0, 0];
211                 p1 = plot(xaxis_center, null_axis);
212                 hold on
213                 p2 = plot(null_axis, yaxis_center);
214                 hold on
215                 p3 = plot(uLinear, frLinear);
216             %   Plot properties
217                 p1.LineWidth = 0.5;
218                 p1.LineStyle = '-';
219                 p2.LineWidth = 0.5;
220                 p2.LineStyle = '-';
221                 p1.Color     = 'k';
222                 p2.Color     = 'k';
223                 p3.Color     = '#0072BD';
224                 p3.LineWidth = 0.5;
225                 p3.LineStyle = '-';
226             %   Axes properties
227                 ax2.FontSize = 18;
228             %   X axis limits
229                 ax2.XLim = xaxis_center;
230             %   Y axis limits
231                 ax2.YLim = yaxis_center;
232             %   Grid and Minorgrid
233                 ax2.XGrid      = 'on';
234                 ax2.YGrid      = 'on';
235                 ax2.XMinorGrid = 'on';
236                 ax2.YMinorGrid = 'on';
237             %   X label
238                 ax2.XLabel.String      = '$u(t)$ $[mm]$';
239                 ax2.XLabel.Interpreter = 'latex';
240                 ax2.XLabel.FontSize    = 30;
241             %   Y label
242                 ax2.YLabel.String      = '$f_r(u,z)$ $[kN]$';
243                 ax2.YLabel.Interpreter = 'latex';
244                 ax2.YLabel.FontSize    = 30;
245             %   Title
246                 title(sprintf('Restoring force'));
247                 ax2.Title.Interpreter = 'latex';
248                 ax2.Title.FontSize    = 30;
249             %   Legend
250                 legend(p3,'$f_r(u,z)$');
251                 ax2.Legend.Interpreter = 'latex';
252                 ax2.Legend.Location    = 'southeast';
253                 ax2.Legend.FontSize    = 20;
254         %   Save graph
255             if nargin > 1
```

```
256                 FigureName =  sprintf('Model validation');
257                 for i=1:length(FigureName)
258                     if FigureName(i)==' '
259                         FigureName(i)='_';
260                     end
261                 end
262                 saveas(fig, fullfile('..', 'Matlab', 'Figures','pdf', ...
263                       FigureName),'pdf');
264                 saveas(fig, fullfile('..', 'Matlab', 'Figures','fig', ...
265                       FigureName),'fig');
266                 saveas(fig, fullfile('..', 'Matlab', 'Figures','tif', ...
267                       FigureName),'tiffn');
268                 saveas(fig, fullfile('..', 'Matlab', 'Figures','png', ...
269                       FigureName),'png');
270             end
271     end
272 %%  SOLVE ODEs SYSTEM WITH ode45
273     function obj = BWBN_solve(obj,~)
274     % BWBN_solve solve the ODEs system of the Bouc-Wen-Baber-Noori
275     % system with Matlab solver ode45 for initial condition at rest.
276     %
277     %   obj = BWBN_solve(obj) solve the ODEs system.
278     %   obj = BWBN_solve(obj,'verbose') solve the ODEs system & display info.
279     %
280     %   OUTPUT:   .u      = system displacement              [mm]
281     %             .v      = system velocity                 [m/s]
282     %             .z      = system hysteretic displacement  [mm]
283     %             .fr     = restoring force                 [kN]
284     %                .el  = elastic component               [kN]
285     %                .hys = hysteretic component            [kN]
286     %                .tot = total restoring force           [kN]
287     %             .eps    = system energy                   [J/kg]
288     %                .el  = elastic energy                  [J/kg]
289     %                .hys = hysteretic dissipated energy    [J/kg]
290     %                .tot = total energy                    [J/kg]
291     %
292     % See also BoucWenClass
293
294     %   Display info 1 (verbose mode)
295         if nargin > 1
296             % ODEs status
297             fprintf('Solving Bouc-Wen-Baber-Noori SDF system...\n');
298             ticStart = tic;
299         end
300     %   Solving ODEs and get results
301         [~,x]   = ode45(@(t,x)solveODE(t,x,obj),obj.Earthquake.time,...
302                   [0 , 0 , 0 , 0, 0]);
303         %   Getting u from state-vector x
304             obj.u   =   x(:,1)';
305         %   Getting v from state-vector x and change [mm/s] to [m/s]
306             obj.v   =   x(:,2)' * 1e-3;
307         %   Getting z from state-vector x
308             obj.z   =   x(:,3)';
309         %   Getting hysteretic normalized energy from state-vector x
310             obj.eps.hys =   x(:,4)';
311         %   Getting elastic normalized energy from state-vector x
312             obj.eps.el  =   x(:,5)';
313         %   Computing total normalized energy
314             obj.eps.tot =   obj.eps.el + obj.eps.hys;
315         %   Computing hysteretic restoring force
316             obj.fr.hys  =   1e-6 * (1 - obj.Parameters.alpha) * ...
317                             obj.Parameters.w0^2 * obj.z * obj.Parameters.mass;
318         %   Computing elastic restoring force
319             obj.fr.el   =   1e-6 * obj.Parameters.alpha * ...
```

```
320                             obj.Parameters.w0^2 * obj.u * obj.Parameters.mass;
321         %   Computing elastic restoring force
322             obj.fr.tot  =   obj.fr.el + obj.fr.hys;
323
324     %   Display info 2 (verbose mode)
325         if nargin > 1
326           t = toc(ticStart);
327           fprintf('ode45 solving time: %.2f sec\n',t);
328         end
329     end
330 %%  SAVE SIMULATED EXPERIMENTAL DATA
331     function BWBN_saveExperimentalData(obj)
332     % BWBN_saveExperimentalData saves Bouc-Wen-Baber-Noori system responces
333     % to simulate experimental data record.
334     %
335     %   BWBN_saveExperimentalData(obj) save data record in '/Matlab/Output'.
336     %
337     % See also BoucWenClass
338
339     %   Save simulated experimental displacements
340         fileID = fopen(fullfile('..', 'Matlab', 'Output', ...
341                 'u_experimental.txt'), 'w');
342         fprintf(fileID,'%.10f\n', obj.u);
343         fclose(fileID);
344     %   Save simulated experimental velocity
345         fileID = fopen(fullfile('..', 'Matlab', 'Output', ...
346                 'v_experimental.txt'), 'w');
347         fprintf(fileID,'%.10f\n', obj.v);
348         fclose(fileID);
349     %   Save simulated experimental total restoring force
350         fileID = fopen(fullfile('..', 'Matlab', 'Output', ...
351                'fr_tot_experimental.txt'), 'w');
352         fprintf(fileID,'%.10f\n', obj.fr.tot);
353         fclose(fileID);
354         % Display status
355         fprintf('Simulated experimental record data saved.\n');
356     end
357 %%  PRINT BOUC-WEN-BABER-NOORI SYSTEM PROPERTIES
358     function BWBN_print(obj)
359     % BWBN_print print on command window earthquake and Bouc-Wen-Baber-Noori
360     % system properties.
361     %
362     %   BWBN_print(obj) print info.
363     %
364     % See also BoucWenClass
365
366     %   Earthquake info
367         fprintf('\n<strong>EARTHQUAKE INFO:</strong>\n')
368         fprintf([' Name: %s\n PGA: %.2f m/s^2\n ts: %.2f sec\n '...
369                  'fs: %.0f Hz\n Duration: %.2f sec\n\n'], ...
370                   obj.Earthquake.name, obj.Earthquake.PGA, ...
371                   obj.Earthquake.ts, ...
372                   obj.Earthquake.fs, obj.Earthquake.time(end));
373     %   Properties
374         fprintf('<strong>BOUC-WEN-BABER-NOORI PROPERTIES:</strong>\n')
375         fprintf([' m: %.2f kg\n ki: %.2e kN/mm\n f: %.2f Hz\n '...
376                  'T: %.2f sec\n \x3C9_n: %.2f rad/sec\n \x3C9_d: %.2f '...
377                  'rad/sec\n \x3B6: %.0f %%\n c: %.2e kg/sec\n\n'],...
378                   obj.Parameters.mass, obj.Parameters.ki,...
379                   obj.Parameters.frequency, obj.Parameters.T, ...
380                   obj.Parameters.w0, obj.Parameters.wd, ...
381                   obj.Parameters.zita * 100, obj.Parameters.c)
382     %   Parameters
383         fprintf('<strong>BOUC-WEN-BABER-NOORI PARAMETERS:</strong>\n')
```

```
384             if isempty(obj.Parameters.hz) ~= 1
385                 fprintf([' \x3B1: %.3f\n ki: %.2e kN/mm\n \x3B2: %.2e\n '...
386                          '\x3B3: %.2e\n n: %.1f\n \x3B4_\x3B7: %.2f\n '...
387                          '\x3B4_A: %.2f\n \x3B4_\x3BD: %.2f \n hz: %.0f\n\n'],...
388                         obj.Parameters.alpha, obj.Parameters.ki, ...
389                         obj.Parameters.beta, obj.Parameters.gamma,...
390                         obj.Parameters.N, obj.Parameters.delta_eta, ...
391                         obj.Parameters.delta_A, obj.Parameters.delta_nu, ...
392                         obj.Parameters.hz)
393             else
394                 fprintf([' \x3B1: %.2f\n ki: %.2e kN/mm\n \x3B2: %.2f\n '...
395                          '\x3B3: %.2f\n n: %.3f\n \x3B4_\x3B7: %.2f\n '...
396                          '\x3B4_A: %.2f\n \x3B4_\x3BD: %.2f \n p: %.2f \n'...
397                          ' \x3B6_0: %.2f \n \x3C8_0: %.2f \n \x3B4'...
398                          '\x3C8_0: %.2f \n \x3BB: %.2f \n q: %.2f\n\n'],...
399                         obj.Parameters.alpha, obj.Parameters.ki,...
400                         obj.Parameters.beta, obj.Parameters.gamma, ...
401                         obj.Parameters.N, obj.Parameters.delta_eta, ...
402                         obj.Parameters.delta_A, obj.Parameters.delta_nu,...
403                         obj.Parameters.p, obj.Parameters.z0,...
404                         obj.Parameters.psi0, obj.Parameters.delta_psi0,...
405                         obj.Parameters.lambda, obj.Parameters.q)
406             end
407         %   Dissipated Energy
408             if isempty(obj.eps.tot) ~= 1
409                 fprintf('<strong>DISSIPATED ENERGY:</strong>\n')
410                 fprintf(' \x3B5_hys: %.2e J/kg\n\n',obj.eps.tot(end))
411             end
412         end
413     %% PLOT BOUC-WEN-BABER-NOORI SYSTEM RESPONSE
414         function BWBN_plot(obj,~)
415         % BWBN_plot plot Bouc-Wen-Baber-Noori system response.
416         %
417         %   BWBN_plot(obj) plots:
418         %     (a) system displacement
419         %     (b) system velocity
420         %     (c) hysteretic restoring force
421         %     (d) dissipated energy.
422         %
423         %   BWBN_plot(obj,'save') save the plot in '/Matlab/Figures'.
424
425         %   Plot system response
426             %   Figure properties
427                 fig = figure;
428                 fig.Position = [0, 0, 1400, 750];
429                 fig.PaperUnits = 'points';
430                 fig.PaperSize = [1400, 750];
431                 fig.PaperPositionMode = 'manual';
432                 fig.PaperPosition = [0, 0, 1400, 750];
433             %   Plot displacement
434                 ax1 = subplot(2,2,1);
435                     p1 = plot(obj.Earthquake.time,obj.u);
436                 %   Plot properties
437                     p1.Color     = '#0072BD';
438                     p1.LineWidth = 0.75;
439                     p1.LineStyle = '-';
440                 %   Axes properties
441                     ax1.FontSize = 18;
442                 %   X axis limits
443                     ax1.XLim = [obj.Earthquake.time(1) obj.Earthquake.time(end)];
444                 %   Y axis limits
445                     ax1.YLim = [-(max(abs(obj.u))+0.3*max(abs(obj.u))), ...
446                                  (max(abs(obj.u))+0.3*max(abs(obj.u)))];
447                 %   Grid and Minorgrid
```

```
448                 ax1.XGrid =      'on';
449                 ax1.YGrid =      'on';
450                 ax1.XMinorGrid = 'on';
451                 ax1.YMinorGrid = 'on';
452             %   X label
453                 ax1.XLabel.Interpreter = 'latex';
454                 ax1.XLabel.String      = '$t$ $[s]$';
455                 ax1.XLabel.FontSize    = 30;
456             %   Y label
457                 ax1.YLabel.Interpreter = 'latex';
458                 ax1.YLabel.String = '$u(t)$ $[mm]$';
459                 ax1.YLabel.FontSize    = 30;
460             %   Title
461                 title(sprintf('Displacements'));
462                 ax1.Title.Interpreter = 'latex';
463                 ax1.Title.FontSize    = 30;
464             %   Legend
465                 legend('$u(t)$');
466                 ax1.Legend.Interpreter = 'latex';
467                 ax1.Legend.FontSize  = 20;
468                 ax1.Legend.EdgeColor = [0.55 0.55 0.55];
469         %   Plot velocity
470             ax2 = subplot(2,2,2);
471                 p2 = plot(obj.Earthquake.time,obj.v);
472             %   Plot properties
473                 p2.Color     = '#D95319';
474                 p2.LineWidth = 0.75;
475                 p2.LineStyle = '-';
476             %   Axes properties
477                 ax2.FontSize = 18;
478             %   X axis limits
479                 ax2.XLim = [obj.Earthquake.time(1) obj.Earthquake.time(end)];
480             %   Y axis limits
481                 ax2.YLim = [-(max(abs(obj.v))+0.3*max(abs(obj.v))), ...
482                             (max(abs(obj.v))+0.3*max(abs(obj.v)))];
483             %   Grid and Minorgrid
484                 ax2.XGrid =      'on';
485                 ax2.YGrid =      'on';
486                 ax2.XMinorGrid = 'on';
487                 ax2.YMinorGrid = 'on';
488             %   X label
489                 ax2.XLabel.Interpreter = 'latex';
490                 ax2.XLabel.String      = '$t$ $[s]$';
491                 ax2.XLabel.FontSize    = 30;
492             %   Y label
493                 ax2.YLabel.Interpreter = 'latex';
494                 ax2.YLabel.String = '$v(t)$ $[m/s]$';
495                 ax2.YLabel.FontSize    = 30;
496             %   Title
497                 title(sprintf('Velocity'));
498                 ax2.Title.Interpreter = 'latex';
499                 ax2.Title.FontSize    = 30;
500             %   Legend
501                 legend('$v(t)$');
502                 ax2.Legend.Interpreter = 'latex';
503                 ax2.Legend.FontSize  = 20;
504                 ax2.Legend.EdgeColor = [0.55 0.55 0.55];
505         %   Plot total restoring force
506             ax3 = subplot(2,2,3);
507                 XMax   = max(abs(obj.u));
508                 xleft  = -(XMax+0.15*XMax);
509                 xright = - 1 * xleft;
510                 ybottom    = - max(abs(obj.fr.tot));
511                 ytop = ybottom*-1;
```

```
512                 xaxis_center = [xleft+0.3*xleft, xright+0.3*xright];
513                 yaxis_center = [ybottom+0.3*ybottom, ytop+0.3*ytop];
514                 null_axis    = [0, 0];
515                 p1 = plot(xaxis_center,null_axis);
516                 hold on
517                 p2 = plot(null_axis,yaxis_center);
518                 hold on
519                 p3 = plot(obj.u, obj.fr.tot);
520             %   Plot properties
521                 p1.Color = 'k';
522                 p2.Color = 'k';
523                 p1.LineWidth = 0.5;
524                 p1.LineStyle = '-';
525                 p2.LineWidth = 0.5;
526                 p2.LineStyle = '-';
527                 p3.Color     = '#A2142F';
528                 p3.LineWidth = 0.5;
529                 p3.LineStyle = '-';
530             %   Axes properties
531                 ax3.FontSize = 18;
532             %   X axis
533                 ax3.XLim = xaxis_center;
534             %   Y axis
535                 ax3.YLim = yaxis_center;
536             %   Grid and Minorgrid
537                 ax3.XGrid =      'on';
538                 ax3.YGrid =      'on';
539                 ax3.XMinorGrid = 'on';
540                 ax3.YMinorGrid = 'on';
541             %   X label
542                 ax3.XLabel.String = '$u(t)$ $[mm]$';
543                 ax3.XLabel.Interpreter = 'latex';
544                 ax3.XLabel.FontSize    = 30;
545             %   Y label
546                 ax3.YLabel.Interpreter = 'latex';
547                 ax3.YLabel.String = '$f_r(u,z)$ $[kN]$';
548                 ax3.YLabel.FontSize    = 30;
549             %   Title
550                 title(sprintf('Restoring force'));
551                 ax3.Title.Interpreter = 'latex';
552                 ax3.Title.FontSize    = 30;
553             %   Legend
554                 legend(p3,'$f_r(u,z)$');
555                 ax3.Legend.Interpreter = 'latex';
556                 ax3.Legend.Location = 'southeast';
557                 ax3.Legend.FontSize  = 20;
558         %   Plot hysteretic dissipated energy
559             ax4 = subplot(2,2,4);
560                 p1 = plot(obj.Earthquake.time, obj.eps.tot);
561             %   Plot properties
562                 p1.Color     = '#7E2F8E';
563                 p1.LineWidth = 0.75;
564                 p1.LineStyle = '-';
565             %   Axes properties
566                 ax4.FontSize = 18;
567             %   X axis limits
568                 ax4.XLim = [obj.Earthquake.time(1) obj.Earthquake.time(end)];
569             %   Y axis limits
570                 ax4.YLim = [obj.eps.tot(1),...
571                         (max(abs(obj.eps.tot))+0.15*max(abs(obj.eps.tot)))];
572             %   Grid and Minorgrid
573                 ax4.XGrid =      'on';
574                 ax4.YGrid =      'on';
575                 ax4.XMinorGrid = 'on';
```

```
576                 ax4.YMinorGrid = 'on';
577             %   X label
578                 ax4.XLabel.Interpreter = 'latex';
579                 ax4.XLabel.String = '$t$ $[s]$';
580                 ax4.XLabel.FontSize    = 30;
581             %   Y label
582                 ax4.YLabel.Interpreter = 'latex';
583                 ax4.YLabel.String = '$\varepsilon_{tot}$(t) $[J/kg]$';
584                 ax4.YLabel.FontSize    = 30;
585             %   Title
586                 title(sprintf('Dissipated energy'));
587                 ax4.Title.Interpreter = 'latex';
588                 ax4.Title.FontSize    = 30;
589             %   Legend
590                 legend('$\varepsilon_{tot}(t)$');
591                 ax4.Legend.Interpreter = 'latex';
592                 ax4.Legend.Location = 'southeast';
593                 ax4.Legend.FontSize  = 20;
594             %   Saving graph
595                 if nargin > 1
596                 FigureName =  sprintf('Results');
597                 for i=1:length(FigureName)
598                     if FigureName(i)==' '
599                         FigureName(i)='_';
600                     end
601                 end
602                 saveas(fig, fullfile('..', 'Matlab', 'Figures','pdf', ...
603                       FigureName),'pdf');
604                 saveas(fig, fullfile('..', 'Matlab', 'Figures','fig', ...
605                       FigureName),'fig');
606                 saveas(fig, fullfile('..', 'Matlab', 'Figures','tif', ...
607                       FigureName),'tiffn');
608                 saveas(fig, fullfile('..', 'Matlab', 'Figures','png', ...
609                       FigureName),'png');
610                 end
611     end
612 %% IDENTIFICATION OF PARAMETERS – PATTERN SEARCH ALGORITHM
613     function obj = BWBN_PatternSearch(obj,lb,ub,randSet,u_experimental,...
614                    fr_experimental)
615     % BWBN_PatternSearch identify Bouc–Wen–Baber–Noori system parameters
616     % by Pattern Search algorithm.
617     %
618     %   BWBN_PatternSearch(obj,lb,ub,randSet,u_experimental)
619     %       lb = scrtucture containig low bounds of parameters
620     %       lb = scrtucture containig low bounds of parameters
621     %       ub = scrtucture containig upper bounds of parameters
622     %       u_experimental  = experimental displacement record
623     %       fr_experimental = experimental restoring force record
624     %
625     % See also BoucWenClass
626
627     global J_opt
628     %   Lower bound conditions
629         LB = [lb.ki; lb.beta; lb.gamma; lb.alpha];
630     %   Upper bound conditions
631         UB = [ub.ki; ub.beta; ub.gamma; ub.alpha];
632     %   1st set of random parameters
633         p0 = [randSet.ki; randSet.beta; randSet.gamma; randSet.alpha];
634     %   Correlation matrix A
635         A  = [0  0  0  0;
636               0 -1 -1  0;
637               0 -1  1  0;
638               0  0  0  0];
639     %   Vector b
```

```
640         b  = [0 0 0 0]';
641     %   Solving patternsearch algorithm
642         %options = optimoptions('patternsearch','MaxIterations',1);
643         [p_opt,J_opt(1)] = patternsearch(@(p)objectivefunction(p,obj,...
644                           u_experimental, 'PATTERN SEARCH'),...
645                           p0,A,b,[],[],LB,UB,[]);%,options);
646         %   Getting identified parameters
647             obj.Parameters.ki        = p_opt(1);
648             obj.Parameters.beta      = p_opt(2);
649             obj.Parameters.gamma     = p_opt(3);
650             obj.Parameters.alpha     = p_opt(4);
651             obj.Parameters.delta_eta = 0;
652     %   Printing results on Command Window
653         clc
654         fprintf('<strong> PATTERN SEARCH ALGORITHM </strong>\n\n')
655         fprintf(' Optimal parameters:\n\n');
656         fprintf([' ki = %.0f \n \x3B2  = %.3e \n \x3B3  = %.3e \n \x3B1'...
657                  '  = %.3f \n J  = %.4f\n \n '], obj.Parameters.ki, ...
658                  obj.Parameters.beta, obj.Parameters.gamma, ...
659                  obj.Parameters.alpha, J_opt(1))
660     %   Saving results in a .txt file
661         fileID = fopen(fullfile('..', 'Matlab', 'Output',...
662                       'PatternSearch.txt'), 'w');
663         fprintf(fileID,'PATTERN SEARCH — Optimal parameters:\n\n');
664         fprintf(fileID,['ki        = %.0f\n\nbeta      = %.3f\n\ngamma',...
665            '     = %.3f\n\nalpha     = %.3f\n\nJ         = %.4f\n \n '],...
666                 obj.Parameters.ki, obj.Parameters.beta, ...
667                 obj.Parameters.gamma, obj.Parameters.alpha, J_opt(1));
668         fclose(fileID);
669     %   Solving BWBN with identified parameters by Pattern Search
670         obj = BWBN_solve(obj,'verobose');
671         clc
672     %   Plot Pattern Search results
673         %   Figure properties
674             fig = figure;
675             fig.Position = [0, 0, 1130, 630];
676             fig.PaperUnits = 'points';
677             fig.PaperSize = [1130, 630];
678             fig.PaperPositionMode = 'manual';
679             fig.PaperPosition = [0, 0, 1130, 630];
680         %   Plot experimental restoring force
681             ax1 = subplot(2,3,1);
682                 XMax    = max(abs(u_experimental));
683                 xleft   = — (XMax + 0.15 * XMax);
684                 xright  = xleft * —1;
685                 YMax    = max(abs(fr_experimental));
686                 ybottom = — (YMax + 0.15 * YMax);
687                 ytop    =  ybottom * —1;
688                 xaxis_center = [xleft, xright];
689                 yaxis_center = [ybottom, ytop];
690                 null_axis    = [0, 0];
691                 p1 = plot(xaxis_center,null_axis);
692                 hold on
693                 p2 = plot(null_axis,yaxis_center);
694                 hold on
695                 p3 = plot(u_experimental,fr_experimental);
696             %   Plot properties
697                 p1.Color = 'k';
698                 p2.Color = 'k';
699                 p1.LineWidth = 0.5;
700                 p1.LineStyle = '—';
701                 p2.LineWidth = 0.5;
702                 p2.LineStyle = '—';
703                 p3.Color     = '#0072BD';
```

```
704                 p3.LineWidth = 0.5;
705                 p3.LineStyle = '-';
706             %   Axes properties
707                 ax1.FontSize = 14;
708             %   X axis
709                 ax1.XLim = xaxis_center;
710             %   Y axis
711                 ax1.YLim = yaxis_center;
712             %   Grid and Minorgrid
713                 ax1.XGrid =      'on';
714                 ax1.YGrid =      'on';
715                 ax1.XMinorGrid = 'on';
716                 ax1.YMinorGrid = 'on';
717             %   X label
718                 ax1.XLabel.String = '$u(t)$ $[mm]$';
719                 ax1.XLabel.Interpreter = 'latex';
720                 ax1.XLabel.FontSize    = 23;
721             %   Y label
722                 ax1.YLabel.String = '$f_r(u,z)$ $[kN]$';
723                 ax1.YLabel.Interpreter = 'latex';
724                 ax1.YLabel.FontSize    = 23;
725             %   Title
726                 title(sprintf('Experimental'));
727                 ax1.Title.Interpreter = 'latex';
728                 ax1.Title.FontSize    = 23;
729             %   Legend
730                 legend(p3,'$f_r(u,z)_{exp}$');
731                 ax1.Legend.Interpreter = 'latex';
732                 ax1.Legend.Location = 'southeast';
733                 ax1.Legend.FontSize  = 17;
734         %   Plot restoring force algorithm
735             ax2 = subplot(2,3,2);
736                 XMax    = max(abs(obj.u));
737                 xleft   = -(XMax+0.15*XMax);
738                 xright  = xleft*-1;
739                 YMax    = max(abs(obj.fr.tot));
740                 ybottom = -(YMax+0.15*YMax);
741                 ytop    =  ybottom*-1;
742                 xaxis_center = [xleft, xright];
743                 yaxis_center = [ybottom, ytop];
744                 null_axis    = [0, 0];
745                 p1 = plot(xaxis_center,null_axis);
746                 hold on
747                 p2 = plot(null_axis,yaxis_center);
748                 hold on
749                 p3 = plot(obj.u, obj.fr.tot);
750             %   Plot properties
751                 p1.Color = 'k';
752                 p2.Color = 'k';
753                 p1.LineWidth = 0.5;
754                 p1.LineStyle = '-';
755                 p2.LineWidth = 0.5;
756                 p2.LineStyle = '-';
757                 p3.Color     = '#A2142F';
758                 p3.LineWidth = 0.5;
759                 p3.LineStyle = '-';
760             %   Axes properties
761                 ax2.FontSize = 14;
762             %   X axis
763                 ax2.XLim = xaxis_center;
764             %   Y axis
765                 ax2.YLim = yaxis_center;
766             %   Grid and Minorgrid
767                 ax2.XGrid =      'on';
```

```
768                 ax2.YGrid =      'on';
769                 ax2.XMinorGrid = 'on';
770                 ax2.YMinorGrid = 'on';
771             %   X label
772                 ax2.XLabel.String = '$u(t)$ $[mm]$';
773                 ax2.XLabel.Interpreter = 'latex';
774                 ax2.XLabel.FontSize    = 23;
775             %   Y label
776                 ax2.YLabel.String = '$f_r(u,z)$ $[kN]$';
777                 ax2.YLabel.Interpreter = 'latex';
778                 ax2.YLabel.FontSize    = 23;
779             %   Title
780                 title(sprintf('Identified'));
781                 ax2.Title.Interpreter = 'latex';
782                 ax2.Title.FontSize    = 23;
783             %   Legend
784                 STR = 'PS';
785                 str = sprintf('$f_r(u,z)_{%s}$',STR);
786                 legend(str);
787                 ax2.Legend.Interpreter = 'latex';
788                 ax2.Legend.Location    = 'southeast';
789                 ax2.Legend.FontSize    = 17;
790         %   Plot suverposition
791             ax3 = subplot(2,3,3);
792                 XMax    = max([max(abs(u_experimental)), max(abs(obj.u))]);
793                 xleft   = -(XMax+0.15*XMax);
794                 xright  = xleft*-1;
795                 YMax = max([max(abs(fr_experimental)),max(abs(obj.fr.tot))]);
796                 ybottom = -(YMax+0.15*YMax);
797                 ytop    =  ybottom*-1;
798                 xaxis_center = [xleft, xright];
799                 yaxis_center = [ybottom, ytop];
800                 null_axis    = [0, 0];
801                 p1 = plot(xaxis_center,null_axis);
802                 hold on
803                 p2 = plot(null_axis,yaxis_center);
804                 hold on
805                 p3 = plot(u_experimental,fr_experimental);
806                 hold on
807                 p4 = plot(obj.u, obj.fr.tot);
808             %   Plot properties
809                 p1.Color = 'k';
810                 p2.Color = 'k';
811                 p1.LineWidth = 0.5;
812                 p1.LineStyle = '-';
813                 p2.LineWidth = 0.5;
814                 p2.LineStyle = '-';
815                 p3.Color     = '#0072BD';
816                 p3.LineWidth = 0.25;
817                 p3.LineStyle = '-';
818                 p4.Color     = '#A2142F';
819                 p4.LineWidth = 0.25;
820                 p4.LineStyle = '--';
821             %   Axes properties
822                 ax3.FontSize = 14;
823             %   X axis
824                 ax3.XLim = xaxis_center;
825             %   Y axis
826                 ax3.YLim = yaxis_center;
827             %   Grid and Minorgrid
828                 ax3.XGrid =      'on';
829                 ax3.YGrid =      'on';
830                 ax3.XMinorGrid = 'on';
831                 ax3.YMinorGrid = 'on';
```

```
832             %   X label
833                 ax3.XLabel.String = '$u(t)$ $[mm]$';
834                 ax3.XLabel.Interpreter = 'latex';
835                 ax3.XLabel.FontSize    = 23;
836             %   Y label
837                 ax3.YLabel.String = '$f_r(u,z)$ $[kN]$';
838                 ax3.YLabel.Interpreter = 'latex';
839                 ax3.YLabel.FontSize    = 23;
840             %   Title
841                 title(sprintf('Discrepancy'));
842                 ax3.Title.Interpreter = 'latex';
843                 ax3.Title.FontSize    = 23;
844             %   Legend
845                 STR = 'PS';
846                 str = sprintf('$f_r(u,z)_{%s}$',STR);
847                 legend([p3 p4],'$f_r(u,z)_{exp}$',str);
848                 ax3.Legend.Interpreter = 'latex';
849                 ax3.Legend.Location = 'southeast';
850                 ax3.Legend.FontSize  = 17;
851         %   Plot displacements
852             ax7 = subplot(2,3,[4,6]);
853                 p1 = plot(obj.Earthquake.time,u_experimental);
854                 hold on
855                 p2 = plot(obj.Earthquake.time,obj.u);
856             %   Plot properties
857                 p1.Color     = '#0072BD';
858                 p1.LineWidth = 0.75;
859                 p1.LineStyle = '-';
860                 p2.Color     = '#D95319';
861                 p2.LineWidth = 0.75;
862                 p2.LineStyle = '--';
863             %   Axes properties
864                 ax7.FontSize = 14;
865             %   X axis limits
866                 ax7.XLim = [obj.Earthquake.time(1) obj.Earthquake.time(end)];
867             %   Y axis limits
868                 YMax = max([max(abs(u_experimental)), max(abs(obj.u))]);
869                 ax7.YLim = [-(YMax+0.15*YMax) (YMax+0.15*YMax)];
870             %   Grid and Minorgrid
871                 ax7.XGrid =      'on';
872                 ax7.YGrid =      'on';
873                 ax7.XMinorGrid = 'on';
874                 ax7.YMinorGrid = 'on';
875             %   X label
876                 ax7.XLabel.String = '$t$ $[s]$';
877                 ax7.XLabel.Interpreter = 'latex';
878                 ax7.XLabel.FontSize    = 23;
879             %   Y label
880                 ax7.YLabel.String = '$u(t)$ $[mm]$';
881                 ax7.YLabel.Interpreter = 'latex';
882                 ax7.YLabel.FontSize    = 23;
883             %   Title
884                 title(sprintf('Displacements'));
885                 ax7.Title.Interpreter = 'latex';
886                 ax7.Title.FontSize    = 23;
887             %   Legend
888                 STR = 'PS';
889                 str = sprintf('$u(t)_{%s}$',STR);
890                 legend('$u(t)_{exp}$',str);
891                 ax7.Legend.Interpreter = 'latex';
892                 ax7.Legend.FontSize  = 17;
893         %   Saving graph
894             FigureName =  sprintf(sprintf('Pattern_Search'));
895             saveas(fig, fullfile('..', 'Matlab', 'Figures','pdf',...
```

```
                FigureName),'pdf');
            saveas(fig, fullfile('..', 'Matlab', 'Figures','fig', ...
                FigureName),'fig');
            saveas(fig, fullfile('..', 'Matlab', 'Figures','tif', ...
                FigureName),'tiffn');
            saveas(fig, fullfile('..', 'Matlab', 'Figures','png', ...
                FigureName),'png');
    end
%%  IDENTIFICATION OF PARAMETERS - INTERIOR POINT ALGORITHM
    function obj = BWBN_InteriorPoint(obj,lb,ub,randSet,u_experimental,...
                fr_experimental)
    % BWBN_InteriorPoint identify Bouc-Wen-Baber-Noori system parameters
    % by Interior Point algorithm.
    %
    %   BWBN_InteriorPoint(obj,lb,ub,randSet,u_experimental)
    %       lb = scrtucture containig low bounds of parameters
    %       lb = scrtucture containig low bounds of parameters
    %       ub = scrtucture containig upper bounds of parameters
    %       u_experimental  = experimental displacement record
    %       fr_experimental = experimental restoring force record
    %
    % See also BoucWenClass

    global J_opt
    %   Lower bound conditions
        LB = [lb.ki; lb.beta; lb.gamma; lb.alpha];
    %   Upper bound conditions
        UB = [ub.ki; ub.beta; ub.gamma; ub.alpha];
    %   1st set of random parameters
        p0 = [randSet.ki; randSet.beta; randSet.gamma; randSet.alpha];
    %   Correlation matrix A
        A  = [0  0  0  0;
              0 -1 -1  0;
              0 -1  1  0;
              0  0  0  0];
    %   Vector b
        b  = [0 0 0 0]';
    %   Solving interiorpoint algorithm
        %options = optimoptions(@fmincon,'Display','iter','MaxIterations',1);
        [p_opt,J_opt(2)] = fmincon(@(p)objectivefunction(p,obj,...
                           u_experimental,'INTERIOR POINT'),...
                           p0,A,b,[],[],LB,UB,[]);%,options);
    %   Getting identified parameters
        obj.Parameters.ki        = p_opt(1);
        obj.Parameters.beta      = p_opt(2);
        obj.Parameters.gamma     = p_opt(3);
        obj.Parameters.alpha     = p_opt(4);
        obj.Parameters.delta_eta = 0;
    %   Printing results on Command Window
        clc
        fprintf('<strong> INTERIOR POINT ALGORITHM </strong>\n\n')
        fprintf(' Optimal parameters:\n\n');
        fprintf([' ki = %.0f \n \x3B2  = %.3e \n \x3B3  = %.3e \n \x3B1'...
                 '  = %.3f \n J  = %.4f\n \n '], obj.Parameters.ki, ...
                obj.Parameters.beta, obj.Parameters.gamma, ...
                obj.Parameters.alpha, J_opt(1))
    %   Saving results in a .txt file
        fileID = fopen(fullfile('..', 'Matlab', 'Output', ...
            'InteriorPoint.txt'), 'w');
        fprintf(fileID,'PATTERN SEARCH - Optimal parameters:\n\n');
        fprintf(fileID,['ki        = %.0f\n\nbeta      = %.3f\n\ngamma',...
             '     = %.3f\n\nalpha     = %.3f\n\nJ         = %.4f\n \n '],...
                  obj.Parameters.ki, obj.Parameters.beta,...
                  obj.Parameters.gamma, obj.Parameters.alpha, J_opt(1));
```

```
960         fclose(fileID);
961     %   Solving BWBN with identified parameters by Interior Point
962         obj = BWBN_solve(obj,'verobose');
963         clc
964     %   Plot Interior Point results
965         %   Figure properties
966             fig = figure;
967             fig.Position = [0, 0, 1130, 630];
968             fig.PaperUnits = 'points';
969             fig.PaperSize = [1130, 630];
970             fig.PaperPositionMode = 'manual';
971             fig.PaperPosition = [0, 0, 1130, 630];
972         %   Plot experimental restoring force
973             ax1 = subplot(2,3,1);
974                 XMax    = max(abs(u_experimental));
975                 xleft   = - (XMax + 0.15 * XMax);
976                 xright  = xleft * -1;
977                 YMax    = max(abs(fr_experimental));
978                 ybottom = - (YMax + 0.15 * YMax);
979                 ytop    =  ybottom * -1;
980                 xaxis_center = [xleft, xright];
981                 yaxis_center = [ybottom, ytop];
982                 null_axis    = [0, 0];
983                 p1 = plot(xaxis_center,null_axis);
984                 hold on
985                 p2 = plot(null_axis,yaxis_center);
986                 hold on
987                 p3 = plot(u_experimental,fr_experimental);
988             %   Plot properties
989                 p1.Color = 'k';
990                 p2.Color = 'k';
991                 p1.LineWidth = 0.5;
992                 p1.LineStyle = '-';
993                 p2.LineWidth = 0.5;
994                 p2.LineStyle = '-';
995                 p3.Color     = '#0072BD';
996                 p3.LineWidth = 0.5;
997                 p3.LineStyle = '-';
998             %   Axes properties
999                 ax1.FontSize = 14;
1000            %   X axis
1001                ax1.XLim = xaxis_center;
1002            %   Y axis
1003                ax1.YLim = yaxis_center;
1004            %   Grid and Minorgrid
1005                ax1.XGrid =      'on';
1006                ax1.YGrid =      'on';
1007                ax1.XMinorGrid = 'on';
1008                ax1.YMinorGrid = 'on';
1009            %   X label
1010                ax1.XLabel.String = '$u(t)$ $[mm]$';
1011                ax1.XLabel.Interpreter = 'latex';
1012                ax1.XLabel.FontSize    = 23;
1013            %   Y label
1014                ax1.YLabel.String = '$f_r(u,z)$ $[kN]$';
1015                ax1.YLabel.Interpreter = 'latex';
1016                ax1.YLabel.FontSize    = 23;
1017            %   Title
1018                title(sprintf('Experimental'));
1019                ax1.Title.Interpreter = 'latex';
1020                ax1.Title.FontSize    = 23;
1021            %   Legend
1022                legend(p3,'$f_r(u,z)_{exp}$');
1023                ax1.Legend.Interpreter = 'latex';
```

```
1024                ax1.Legend.Location = 'southeast';
1025                ax1.Legend.FontSize  = 17;
1026            %   Plot restoring force algorithm
1027            ax2 = subplot(2,3,2);
1028                XMax    = max(abs(obj.u));
1029                xleft   = -(XMax+0.15*XMax);
1030                xright  = xleft*-1;
1031                YMax    = max(abs(obj.fr.tot));
1032                ybottom = -(YMax+0.15*YMax);
1033                ytop    =  ybottom*-1;
1034                xaxis_center = [xleft, xright];
1035                yaxis_center = [ybottom, ytop];
1036                null_axis    = [0, 0];
1037                p1 = plot(xaxis_center,null_axis);
1038                hold on
1039                p2 = plot(null_axis,yaxis_center);
1040                hold on
1041                p3 = plot(obj.u, obj.fr.tot);
1042            %   Plot properties
1043                p1.Color = 'k';
1044                p2.Color = 'k';
1045                p1.LineWidth = 0.5;
1046                p1.LineStyle = '-';
1047                p2.LineWidth = 0.5;
1048                p2.LineStyle = '-';
1049                p3.Color     = '#A2142F';
1050                p3.LineWidth = 0.5;
1051                p3.LineStyle = '-';
1052            %   Axes properties
1053                ax2.FontSize = 14;
1054            %   X axis
1055                ax2.XLim = xaxis_center;
1056            %   Y axis
1057                ax2.YLim = yaxis_center;
1058            %   Grid and Minorgrid
1059                ax2.XGrid =      'on';
1060                ax2.YGrid =      'on';
1061                ax2.XMinorGrid = 'on';
1062                ax2.YMinorGrid = 'on';
1063            %   X label
1064                ax2.XLabel.String = '$u(t)$ $[mm]$';
1065                ax2.XLabel.Interpreter = 'latex';
1066                ax2.XLabel.FontSize    = 23;
1067            %   Y label
1068                ax2.YLabel.String = '$f_r(u,z)$ $[kN]$';
1069                ax2.YLabel.Interpreter = 'latex';
1070                ax2.YLabel.FontSize    = 23;
1071            %   Title
1072                title(sprintf('Identified'));
1073                ax2.Title.Interpreter = 'latex';
1074                ax2.Title.FontSize    = 23;
1075            %   Legend
1076                STR = 'IP';
1077                str = sprintf('$f_r(u,z)_{%s}$',STR);
1078                legend(str);
1079                ax2.Legend.Interpreter = 'latex';
1080                ax2.Legend.Location    = 'southeast';
1081                ax2.Legend.FontSize    = 17;
1082            %   Plot suverposition
1083            ax3 = subplot(2,3,3);
1084                XMax    = max([max(abs(u_experimental)), max(abs(obj.u))]);
1085                xleft   = -(XMax+0.15*XMax);
1086                xright  = xleft*-1;
1087                YMax = max([max(abs(fr_experimental)),max(abs(obj.fr.tot))]);
```

```
1088                ybottom = -(YMax+0.15*YMax);
1089                ytop    =  ybottom*-1;
1090                xaxis_center = [xleft, xright];
1091                yaxis_center = [ybottom, ytop];
1092                null_axis    = [0, 0];
1093                p1 = plot(xaxis_center,null_axis);
1094                hold on
1095                p2 = plot(null_axis,yaxis_center);
1096                hold on
1097                p3 = plot(u_experimental,fr_experimental);
1098                hold on
1099                p4 = plot(obj.u, obj.fr.tot);
1100            %   Plot properties
1101                p1.Color = 'k';
1102                p2.Color = 'k';
1103                p1.LineWidth = 0.5;
1104                p1.LineStyle = '-';
1105                p2.LineWidth = 0.5;
1106                p2.LineStyle = '-';
1107                p3.Color     = '#0072BD';
1108                p3.LineWidth = 0.25;
1109                p3.LineStyle = '-';
1110                p4.Color     = '#A2142F';
1111                p4.LineWidth = 0.25;
1112                p4.LineStyle = '--';
1113            %   Axes properties
1114                ax3.FontSize = 14;
1115            %   X axis
1116                ax3.XLim = xaxis_center;
1117            %   Y axis
1118                ax3.YLim = yaxis_center;
1119            %   Grid and Minorgrid
1120                ax3.XGrid =      'on';
1121                ax3.YGrid =      'on';
1122                ax3.XMinorGrid = 'on';
1123                ax3.YMinorGrid = 'on';
1124            %   X label
1125                ax3.XLabel.String = '$u(t)$ $[mm]$';
1126                ax3.XLabel.Interpreter = 'latex';
1127                ax3.XLabel.FontSize    = 23;
1128            %   Y label
1129                ax3.YLabel.String = '$f_r(u,z)$ $[kN]$';
1130                ax3.YLabel.Interpreter = 'latex';
1131                ax3.YLabel.FontSize    = 23;
1132            %   Title
1133                title(sprintf('Discrepancy'));
1134                ax3.Title.Interpreter = 'latex';
1135                ax3.Title.FontSize    = 23;
1136            %   Legend
1137                STR = 'IP';
1138                str = sprintf('$f_r(u,z)_{%s}$',STR);
1139                legend([p3 p4],'$f_r(u,z)_{exp}$',str);
1140                ax3.Legend.Interpreter = 'latex';
1141                ax3.Legend.Location = 'southeast';
1142                ax3.Legend.FontSize  = 17;
1143        %   Plot displacements
1144            ax7 = subplot(2,3,[4,6]);
1145                p1 = plot(obj.Earthquake.time,u_experimental);
1146                hold on
1147                p2 = plot(obj.Earthquake.time,obj.u);
1148            %   Plot properties
1149                p1.Color     = '#0072BD';
1150                p1.LineWidth = 0.75;
1151                p1.LineStyle = '-';
```

```
                p2.Color     = '#D95319';
                p2.LineWidth = 0.75;
                p2.LineStyle = '--';
            %   Axes properties
                ax7.FontSize = 14;
            %   X axis limits
                ax7.XLim = [obj.Earthquake.time(1) obj.Earthquake.time(end)];
            %   Y axis limits
                YMax = max([max(abs(u_experimental)), max(abs(obj.u))]);
                ax7.YLim = [-(YMax+0.15*YMax) (YMax+0.15*YMax)];
            %   Grid and Minorgrid
                ax7.XGrid =      'on';
                ax7.YGrid =      'on';
                ax7.XMinorGrid = 'on';
                ax7.YMinorGrid = 'on';
            %   X label
                ax7.XLabel.String = '$t$ $[s]$';
                ax7.XLabel.Interpreter = 'latex';
                ax7.XLabel.FontSize    = 23;
            %   Y label
                ax7.YLabel.String = '$u(t)$ $[mm]$';
                ax7.YLabel.Interpreter = 'latex';
                ax7.YLabel.FontSize    = 23;
            %   Title
                title(sprintf('Displacements'));
                ax7.Title.Interpreter = 'latex';
                ax7.Title.FontSize    = 23;
            %   Legend
                STR = 'IP';
                str = sprintf('$u(t)_{%s}$',STR);
                legend('$u(t)_{exp}$',str);
                ax7.Legend.Interpreter = 'latex';
                ax7.Legend.FontSize  = 17;
        %   Saving graph
            FigureName =  sprintf(sprintf('Interior_Point'));
            saveas(fig, fullfile('..', 'Matlab', 'Figures','pdf', ...
                  FigureName),'pdf');
            saveas(fig, fullfile('..', 'Matlab', 'Figures','fig', ...
                  FigureName),'fig');
            saveas(fig, fullfile('..', 'Matlab', 'Figures','tif', ...
                  FigureName),'tiffn');
            saveas(fig, fullfile('..', 'Matlab', 'Figures','png', ...
                  FigureName),'png');
    end
  end
end
```

EarthquakeClass.m

```
classdef EarthquakeClass
% Earthquake class
%
%   .name              =  earthquake name
%   .record            =  earthquake record
%   .ag                =  ground acceleration
%   .ts                =  sampling time
%   .filter            =  filter of accelerogrram
%   .filterParameters  =  filter parameters
%   .fs                =  sampling frequency
%   .PGA               =  peak groud acceleration
%   .time              =  time window
%
% See also BoucWenClass, ParametersClass, BWBN_plotFFT.

```

```
16      properties
17          name
18          record
19          ag
20          ts
21          filter
22          filterParameters
23      end
24
25      properties (Dependent)
26          fs
27          PGA
28          time
29      end
30
31      methods
32      %   Get fs
33          function f = get.fs(obj)
34              f = 1 /  obj.ts;
35          end
36      %   Get PGA
37          function pga = get.PGA(obj)
38              pga = max(abs(obj.ag));
39          end
40      %   Get time window
41          function t = get.time(obj)
42              t = (0:length(obj.ag)-1).*obj.ts;
43          end
44      %   Plot FFT of ag
45          function BWBN_plotFFT(obj,~)
46          % BWBN_plotFFT evaluate the FFT of the input acceletogram.
47          %
48          %   BWBN_plotFFT(obj) plot the FFT of the input acceletogram.
49          %   BWBN_plotFFT(obj,'save') plot and save the FFT.
50          %
51          % See also EarthquakeClass, BoucWenClass
52
53              Af   = fft(obj.ag) * obj.ts;
54              Af   = abs(fftshift(Af));
55              NFFT = length(Af);
56              Tf   = NFFT/obj.fs;
57              if mod(NFFT,2)==1
58                  F = - obj.fs/2 : 1/Tf : obj.fs/2-1/Tf;
59              else
60                  F = - obj.fs/2 : 1/Tf : obj.fs/2;
61              end
62              [~, I] = max(Af);
63              fprintf('Frequency of maximum amplitude: %.3f Hz\n',abs(F(I)))
64              %   PLOT FFT
65                  %   Figure properties
66                      fig = figure;
67                      fig.Position          = [0, 0, 800, 450];
68                      fig.PaperUnits        = 'points';
69                      fig.PaperSize         = [800, 450];
70                      fig.PaperPositionMode = 'manual';
71                      fig.PaperPosition     = [0, 0, 800, 450];
72                  %   Plot data
73                      plot(F,Af);
74                  %   Axes properties
75                      ax = gca;
76                      ax.FontSize = 15;
77                  %   X axis Limits
78                      ax.XLim = [0 50];
79                  %   Grid and Minorgrid
```

```
                ax.XGrid      = 'on';
                ax.YGrid      = 'on';
                ax.XMinorGrid = 'on';
                ax.YMinorGrid = 'on';
            %   X label
                ax.XLabel.String      = '$f$ $[Hz]$';
                ax.XLabel.Interpreter = 'latex';
                ax.XLabel.FontSize    = 26;
            %   Y label
                ax.YLabel.String      = '$A(f)$ $[m/s^2]$';
                ax.YLabel.Interpreter = 'latex';
                ax.YLabel.FontSize    = 26;
            %   Title
                title(sprintf('%s FFT', obj.name));
                ax.Title.Interpreter = 'latex';
                ax.Title.FontSize    = 26;
            %   Legend
                legend('$A(f)$')
                ax.Legend.Interpreter = 'latex';
                ax.Legend.FontSize  = 18;
            %   Save graph
                if nargin > 1
                    fprintf('Saving graph...\n')
                    FigureName =  sprintf('FFT %s',obj.name);
                    for i = 1 : length(FigureName)
                        if FigureName(i) ==' '
                            FigureName(i) ='_';
                        end
                    end
                    saveas(fig, fullfile('..', 'Matlab', 'Figures','pdf', ...
                            FigureName),'pdf');
                    saveas(fig, fullfile('..', 'Matlab', 'Figures','fig', ...
                            FigureName),'fig');
                    saveas(fig, fullfile('..', 'Matlab', 'Figures','tif', ...
                            FigureName),'tiffn');
                    saveas(fig, fullfile('..', 'Matlab', 'Figures','png', ...
                    FigureName),'png');
                end
        end
    end
end
```

ParametersClass.m

```
classdef ParametersClass
% Bouc–Wen–Baber–Noori system parameters class
%
%   .alpha      =  sub–class containing earthquake data
%   .ki         =  system initial stiffness
%   .beta       =  BWBN parameter
%   .gamma      =  BWBN parameter
%   .N          =  BWBN parameter
%   .delta_eta  =  BWBN parameter
%   .delta_A    =  BWBN parameter
%   .delta_nu   =  BWBN parameter
%   .hz         =  BWBN pinching parameter
%   .p          =  BWBN pinching parameter
%   .z0         =  BWBN pinching parameter
%   .psi0       =  BWBN pinching parameter
%   .delta_psi0 =  BWBN pinching parameter
%   .lambda     =  BWBN pinching parameter
%   .q          =  BWBN pinching parameter
%   .mass       =  system mass
%   .zita       =  system damping ratio
```

```
%   .frequency  =  system natural frequency
%   .T          =  system period
%   .c          =  system viscous damping
%   .w0         =  system natural pulsation
%   .wd         =  system damped pulsation
%
% See also BoucWenClass, EarthquakeClass.

    properties
        alpha
        ki
        beta
        gamma
        N
        delta_eta
        delta_A
        delta_nu
        hz
        mass
        zita
    end

    properties (Dependent)
        frequency
        T
        c
        w0
        wd
    end

    properties (Hidden)
        p
        z0
        psi0
        delta_psi0
        lambda
        q
    end

    methods
    %   Get system natural pulsation
        function w0 = get.w0(obj)
           w0 = sqrt(obj.ki * 1e6/(obj.mass));
        end
    %   Get system damped pulsation
        function wd = get.wd(obj)
           wd = obj.w0 * sqrt(1-(obj.zita)^2);
        end
    %   Get system natural frequency
        function f = get.frequency(obj)
            f = obj.w0 / (2*pi);
        end
    %   Get system natural period
        function period = get.T(obj)
            period = 1/obj.frequency;
        end
    %   Get linear viscous damping coefficient
        function C = get.c(obj)
            C = obj.zita * 2 * obj.mass * obj.w0;
        end
     end
end
```

BWBN experimental data.m

```
%% BOUC-WEN-BABER-NOORI SDF SYSTEM 'EXPERIMENTAL DATA' AND MODEL VALIDATION

% INFO SCRIPT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                         %
%   The script simulates and generates 'experimental data' of a hysteretic SDF %
%   Bouc-Wen-Baber-Noori system with degradation                          %
%                                                                         %
% STRUCTURES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                         %
%   .Earthquake      sub-structure containing earthquake data             %
%      .name              =  ['']       earthquake's name                 %
%      .record            =  [1/g]      .txt earthquake record data       %
%      .ts                =  [sec]      earthquake record sampling time   %
%      .filter            =  ['']       'YES'/'NO' to filter ground motion acc. %
%      .filterParameters  =  [-]        filter parameters                 %
%      .fs                =  [Hz]       sampling frequency                %
%      .ag                =  [m/s^2]    ground motion acceleration        %
%      .PGA               =  [m/s^2]    peak ground acceleration          %
%      .time              =  [sec]      earthquake time window            %
%                                                                         %
%   .Parameters      sub-structure containing BWBN parameters             %
%        .alpha           =  [-]        stiffness ratio (ku/ki)           %
%        .ki              =  [kN/mm]    BWBN initial stiffness parameter  %
%        .beta            =  [-]        BWBN parameter                    %
%        .gamma           =  [-]        BWBN parameter                    %
%        .N               =  [-]        BWBN hardening-softening parameter %
%        .delta_eta       =  [-]        BWBN stiffness degradation parameter %
%        .delta_A         =  [-]        BWBN linear variant parameter     %
%        .delta_nu        =  [-]        BWBN strength deterioration parameter %
%        .hz              =  [-]        BWBN pinching function            %
%        .p               =  [-]        BWBN pinching parameter           %
%        .z0              =  [-]        BWBN pinching parameter           %
%        .psi0            =  [-]        BWBN pinching parameter           %
%        .delta_psi0      =  [-]        BWBN pinching parameter           %
%        .lambda          =  [-]        BWBN pinching parameter           %
%        .q               =  [-]        BWBN pinching parameter           %
%        .frequency       =  [Hz]       system's natural frequency        %
%        .zita            =  [-]        system's damping ratio            %
%        .mass            =  [kg]       system's mass                     %
%        .c               =  [-]        system's viscous damping coefficient %
%        .w0              =  [rad/sec]  system's natural frequency        %
%        .wd              =  [rad/sec]  system's damping frequency        %
%                                                                         %
%   .u                    =  [mm]       displacement response             %
%   .v                    =  [m/s]      velocity response                 %
%   .z                    =  [mm]       hysteretic displacement           %
%                                                                         %
%   .fr               restoring force of the BWBN SDF system              %
%      .hys               =  [kN]       restoring force hysteretic component %
%      .el                =  [kN]       restoring force elastic component %
%      .tot               =  [kN]       total restoring force of the system %
%                                                                         %
%   .eps              dissipated energy of the BWBN SDF system            %
%      .hys               =  [J/kg]     normalized hysterertic energy     %
%      .el                =  [J/kg]     normalized elastic energy         %
%      .tot               =  [J/kg]     normalized total energy           %
%                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   Clear all variables from the workspace, close all figures, and add path
    clearvars; close all
    addpath('Input', 'Output', 'Functions', 'Classes', 'Sessions')
%   Clear command window
```

```
64    clc; fprintf(['<strong>SIMULATION OF EXPERIMENTAL RECORD DATA FOR A '...
65                  'NONLINEAR BOUC-WEN-BABER-NOORI SDF SYSTEM:</strong>\n\n'])
66
67 %% DEFINING BOUC-WEN-BABER-NOORI SYSTEM CLASS
68    BoucWen = BoucWenClass;
69
70 %% EARTHQUAKE LOAD
71    BoucWen.Earthquake.name    = 'Montenegro (1979)';
72    BoucWen.Earthquake.record  = load('RSN4451_MONTENE.GRO_BSO000.txt')';
73    BoucWen.Earthquake.ag      = BoucWen.Earthquake.record * 9.81;
74    BoucWen.Earthquake.ts      = 0.01;
75    %  Filtering ground motion acceleration
76       BoucWen.Earthquake.filter  = 'YES';
77       %  .filterParameters = [WpMax, Rp, Rs, Tuning];
78          BoucWen.Earthquake.filterParameters =  [20, 3, 8, 0];
79          if strcmp('YES',BoucWen.Earthquake.filter) == 1
80             BoucWen = EarthquakeFiltering(BoucWen);
81             if BoucWen.Earthquake.filterParameters(end) == 1
82                return
83             end
84          end
85    %  Plot Accelerogram
86       BWBN_accelerogram(BoucWen,'save');
87
88 %% SYSTEM'S PARAMETERS
89    BoucWen.Parameters.alpha     = 0.05;
90    BoucWen.Parameters.ki        = 7.6;
91    BoucWen.Parameters.beta      = 63e-3;
92    BoucWen.Parameters.gamma     = BoucWen.Parameters.beta * 1;
93    BoucWen.Parameters.N         = 1;
94    BoucWen.Parameters.delta_eta = 6.5;
95    BoucWen.Parameters.delta_A   = 0;
96    BoucWen.Parameters.delta_nu  = 2.3;
97    BoucWen.Parameters.mass      = 12000;
98    BoucWen.Parameters.zita      = 0.03;
99    %  Pinching (.hz == 1 no pinching /.hz == [] pinching)
100      BoucWen.Parameters.hz = 1;
101      %  Pinching parameters
102         if isempty(BoucWen.Parameters.hz) == 1
103            BoucWen.Parameters.p          = 2;
104            BoucWen.Parameters.z0         = 1;
105            BoucWen.Parameters.psi0       = 0.5;
106            BoucWen.Parameters.delta_psi0 = 0.6;
107            BoucWen.Parameters.lambda     = 0.9;
108            BoucWen.Parameters.q          = 1;
109         end
110
111 %% MODEL VALIDATION IN LINEAR FIELD
112    BWBN_linearValidation(BoucWen,'save');
113
114 %% EXPERIMENTAL DATA RECORDS SIMULATION
115    %  Solve Bouc-Wen-Baber-Noori system with ode45
116       BoucWen = BWBN_solve(BoucWen,'verbose');
117    %  Save experimental data simulation
118       BWBN_saveExperimentalData(BoucWen);
119    %  Print Bouc-Wen-BAber-Noori system info
120       BWBN_print(BoucWen);
121    %  Plot Bouc-Wen-Baber-Noori system response
122       BWBN_plot(BoucWen,'savePlot');
123
124 %% SAVE WORKSPACE
125    save(fullfile('..', 'Matlab', 'Sessions', 'BWBN_experimental_data.mat'));
```

BWBN identification.m

```
%% IDENTIFICATION OF BOUC-WEN-BABER-NOORI SDF SYSTEM'S PARAMETERS WITH PS & IP

% INFO SCRIPT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
%  The script identify the parameters of the hysteretic Bouc-Wen-Baber-Noori   %
%  system using Pattern Search (PS) & Interior Point (IP) algorithms.          %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  Clear all variables from the workspace, close all figures, and add path
   clearvars; close all
   addpath('Input', 'Output', 'Functions', 'Classes', 'Sessions')
%  Clear command window
   clc; fprintf(['<strong>IDENTIFICATION OF BOUC-WEN-BABER-NOORI SDF'...
                 'SYSTEM''S PARAMETERS\nWITH PATTERN SEARCH & INTERIOR '...
                 'POINT ALGORITHMS:</strong>\n\n']);

%% DEFINING BOUC WEN SYSTEM CLASS
   BoucWen = BoucWenClass;

%% EARTHQUAKE LOAD
   BoucWen.Earthquake.name    = 'Montenegro (1979)';
   BoucWen.Earthquake.record  = load('RSN4451_MONTENE.GRO_BSO000.txt')';
   BoucWen.Earthquake.ag      = BoucWen.Earthquake.record * 9.81;
   BoucWen.Earthquake.ts      = 0.01;
   %  Filtering ground motion acceleration
      BoucWen.Earthquake.filter  = 'YES';
      %  .filterParameters = [WpMax, Rp, Rs, Tuning];
         BoucWen.Earthquake.filterParameters =  [20, 3, 8, 0];
         if strcmp('YES',BoucWen.Earthquake.filter) == 1
            BoucWen = EarthquakeFiltering(BoucWen);
            if BoucWen.Earthquake.filterParameters(end) == 1
               return
            end
         end

%% SYSTEM'S PARAMETERS
   BoucWen.Parameters.N         = 1;
   BoucWen.Parameters.delta_eta = 0;
   BoucWen.Parameters.delta_A   = 0;
   BoucWen.Parameters.delta_nu  = 0;
   BoucWen.Parameters.mass      = 12000;
   BoucWen.Parameters.zita      = 0.03;
   BoucWen.Parameters.hz        = 1;

%% LOAD 'EXPERIMENTAL DATA'
   u_experimental  = load('u_experimental.txt');
   fr_experimental = load('fr_tot_experimental.txt');

%% PARAMETERS IDENTIFICATION
   %  Lower bound (lb), upper bound (up) of BWBN parameters
   %  and selection of 1st set of random parameters (randSet)
   lb.alpha = 0;              ub.alpha = 1;             randSet.alpha = 0.5;
   lb.ki    = 4.27;           ub.ki    = 11.85;         randSet.ki    = 5;
   lb.beta  = 55e-3;          ub.beta  = 65e-3;         randSet.beta  = 57e-3;
   lb.gamma = -randSet.beta;  ub.gamma = randSet.beta;  randSet.gamma = 0;

   %  Pattern Search algorithm to evaluate BWBN system with no degradation
      BoucWen(1) = BWBN_PatternSearch(BoucWen(1), lb, ub, randSet,...
                   u_experimental, fr_experimental);

   %  Interior Point algorithm to evaluate BWBN system with no degradation
      BoucWen(2) = BWBN_InteriorPoint(BoucWen(1), lb, ub, randSet,...
```

```
64                      u_experimental, fr_experimental);
65
66  %% SAVE
67      save(fullfile('..', 'Matlab', 'Sessions', 'BWBN_identification.mat'));
```

Acknowledgements

Bibliography

[1] Abbiati, G., Bursi, O., Caperan, P., Di Sarno, L., Molina, F., Paolacci, F., & Pegon, P. (2015). Hybrid simulation of a multi-span rc viaduct with plain bars and sliding bearings. *Earthquake Engineering & Structural Dynamics*, *44*, 2221–2240.

[2] Abbiati, G., Miraglia, G., Mojsilovic, N., & Stojadinovic, B. (2017). Hybrid simulation with dynamic substructuring of masonry structures: A numerical study.

[3] Alanqar, A., Durand, H., & Christofides, P. D. (2017). Error-triggered on-line model identification for model-based feedback control. *AIChE Journal*, *63*(3), 949–966. doi:10.1002/aic.15430

[4] Andersen, P., & Kirkegaard, P. (1997). *Statistical damage detection of civil engineering structures using armav models*. Presented at the 16th International Modal Analysis Conference, Santa Barbara, California, USA, February 2-5, 1998 PDF for print: 12 pp. Denmark: Dept. of Building Technology and Structural Engineering, Aalborg University.

[5] Andronikou, A., Bekey, G., & Masri, S. (1982). Identification of nonlinear hysteretic systems using random search. *IFAC Proceedings Volumes*, *15*(4), 331–336. doi:10.1016/s1474-6670(17)63010-6

[6] Baber, T. T., & Noori, M. N. (1985). Random vibration of degrading, pinching systems. *Journal of Engineering Mechanics*, *111*(8), 1010–1026. Retrieved from 10.1061/(ASCE)0733-9399(1985)111:8(1010)

[7] Benedettini, F., Capecchi, D., & Vestroni, F. (1995). Identification of hysteretic oscillators under earthquake loading by nonparametric models. *Journal of Engineering Mechanics*, *121*(5).

[8] Billings, S. (2013). *Nonlinear system identification: Narmax methods in the time, frequency, and spatio-temporal domains*. Wiley. Retrieved from https://books.google.it/books?id=SaQ2AAAAQBAJ

[9] Blatman, G., & Sudret, B. (2011). Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics*, *230*(6), 2345–2367. doi:https://doi.org/10.1016/j.jcp.2010.12.021

[10] Bodeux, J. B., & Golinval, J. C. (2001). Application of ARMAV models to the identification and damage detection of mechanical and civil engineering structures. *Smart Materials and Structures*, *10*(3), 479–489. doi:10.1088/0964-1726/10/3/309

[11] Bouc, R. (1969). Modèle mathématique d'hystérésis: Application aux systèmes à un degré de liberté. *Journal Acustica*, *24*, 16–25.

[12] Brewick, P. T., Masri, S. F., Chassiakos, A. G., & Kosmatopoulos, E. B. (2016). A probabilistic study of the robustness of an adaptive neural estimation method for hysteretic internal forces in nonlinear mdof systems. *Probabilistic Engineering Mechanics*, *45*, 140–156. doi:10.1016/j.probengmech.2016.04.002

[13] Brun, M., Batti, A., Combescure, A., & Gravouil, A. (2014). External coupling software based on macro- and micro-time scales for explicit/implicit multi-time-step co-computations in structural dynamics. *Finite Elements in Analysis and Design*, *86*, 101–119. doi:https:

//doi.org/10.1016/j.finel.2014.04.005

[14] Bursi, O. S., Ceravolo, R., Erlicher, S., & Zanotti Fragonara, L. (2012). Identification of the hysteretic behaviour of a partial-strength steel-concrete moment-resisting frame structure subject to pseudodynamic tests. *Earthquake Engineering & Structural Dynamics*, *41*(14), 1883–1903. doi:10.1002/eqe.2163

[15] Calabrese, A., Strano, S., & Terzo, M. (2018). Adaptive constrained unscented kalman filtering for real-time nonlinear structural system identification. *Structural Control and Health Monitoring*, *25*(2), e2084. doi:10.1002/stc.2084

[16] Catbas, N., Kijewski-Correa, T., & Aktan, A. (2013). *Structural identification of constructed systems - approaches, methods, and technologies for effective practice of st-id*. doi:10.1061/9780784411971

[17] Ceravolo, R., Demarie, G. V., & Erlicher, S. (2010). Instantaneous identification of degrading hysteretic oscillators under earthquake excitation. *Structural Health Monitoring*, *9*(5), 447–464. doi:10.1177/1475921710368202

[18] Ceravolo, R., Erlicher, S., & Fragonara, L. Z. (2013). Comparison of restoring force models for the identification of structures with hysteresis and degradation. *Journal of Sound and Vibration*, *332*(26), 6982–6999. doi:10.1016/j.jsv.2013.08.019

[19] Chassiakos, A., Masri, S., Smyth, A., & Caughey, T. (1998). On-line identification of hysteretic systems. *Journal of applied mechanics*, *65*(1), 194–203.

[20] Chassiakos, A., Masri, S., Smyth, A., & Anderson, J. (1995). Adaptive methods for identification of hysteretic structures, 2349–2353 vol.3. doi:10.1109/ACC.1995.531392

[21] Chatzi, E. N., Smyth, A. W., & Masri, S. F. (2010). Experimental application of on-line parametric identification for nonlinear hysteretic systems with model uncertainty. *Structural Safety*, *32*(5), 326–337. doi:10.1016/j.strusafe.2010.03.008

[22] Chatzi, E., & Papadimitriou, C. (2016). *Identification methods for structural health monitoring*. doi:10.1007/978-3-319-32077-9

[23] Chatzis, M., Chatzi, E., & Triantafyllou, S. (2017). A discontinuous extended kalman filter for non-smooth dynamic problems. *Mechanical Systems and Signal Processing*, *92*, 13–29. doi:10.1016/j.ymssp.2017.01.021

[24] Distefano, N., & Rath, A. (1975). System identification in nonlinear structural seismic dynamics. *Computer Methods in Applied Mechanics and Engineering*, *5*(3), 353–372. doi:10.1016/0045-7825(75)90007-9

[25] Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Ann. Statist.* *32*(2), 407–499. doi:10.1214/009053604000000067

[26] Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis* (2nd ed.). Chapman and Hall/CRC.

[27] Ghobadi, M., Majji, M., & Esfahani, E. T. (2017). Aosid: An analytical solution to the output-only system identification problem to estimate physical parameters and unknown input simultaneously. *Structural Control and Health Monitoring*, *24*(8), e1951. doi:10.1002/stc.1951

[28] Goerens, E. (2018). *Hybrid testing of masonry walls* (Doctoral dissertation, ETH Zürich).

[29] Goodman, J., & Weare, J. (2010). Ensemble samplers with affine invariance. *Communications in Applied Mathematics and Computational Science*, *5*. doi:10.2140/camcos.2010.5.65

[30] Hou, Z., Noori, M., & St. Amand, R. (2000). Wavelet-based approach for structural damage detection. *Journal of Engineering Mechanics-asce - J ENG MECH-ASCE*, *126*. doi:10.1061/(ASCE)0733-9399(2000)126:7(677)

[31] Huh, J., & Haldar, A. (2001). Stochastic finite-element-based seismic risk of nonlinear

structures. *Journal of Structural Engineering*, *127*(3), 323–329. doi:10.1061/(ASCE)0733-9445(2001)127:3(323)

[32] Ikhouane, F., & Rodellar, J. (2007). *Systems with hysteresis: Analysis, identification and control using the bouc-wen model*. doi:10.1002/9780470513200

[33] Kaipio, J., & Somersalo, E. (2005). *Statistical and computational inverse problems*. doi:10.1007/b138659

[34] Kapania, R. K., & Park, S. (1997). Parametric identification of nonlinear structural dynamic systems using time finite element method. *AIAA Journal*, *35*(4), 719–726. doi:10.2514/2.163

[35] Kijewski, T., & Kareem, A. (2003). Wavelet transforms for system identification in civil engineering. *Computer-Aided Civil and Infrastructure Engineering*, *18*(5), 339–355. doi:10.1111/1467-8667.t01-1-00312

[36] Kontoroupi, T., & Smyth, A. W. [Andrew W.]. (2017). Online bayesian model assessment using nonlinear filters. *Structural Control and Health Monitoring*, *24*(3), e1880. doi:10.1002/stc.1880

[37] Li, L., & Qin, H. (2018). An ukf-based nonlinear system identification method using interpolation models and backward integration. *Structural Control and Health Monitoring*, *25*(4), e2129. doi:10.1002/stc.2129

[38] Liu, Y., Goorts, K., Ashasi-Sorkhabi, A., Mercan, O., & Narasimhan, S. (2016). A state space-based explicit integration method for real-time hybrid simulation. *Structural Control and Health Monitoring*, *23*(4), 641–658. doi:10.1002/stc.1798

[39] Lu, Z.-R., Yao, R., Wang, L., & Liu, J. (2017). Identification of nonlinear hysteretic parameters by enhanced response sensitivity approach. *International Journal of Non-Linear Mechanics*, *96*, 1–11. doi:10.1016/j.ijnonlinmec.2017.07.012

[40] Mai, C. V. (2016). *Polynomial chaos expansions for uncertain dynamical systems – applications in earthquake engineering* (Doctoral dissertation, ETH Zürich, Zürich, Switzerland).

[41] Mai, C., Konakli, K., & Sudret, B. (2017). Seismic fragility curves for structures using non-parametric representations. *Frontiers of Structural and Civil Engineering*, *11*(2), 169–186. doi:10.1007/s11709-017-0385-y

[42] Mai, C., Spiridonakos, M., Chatzi, E., & Sudret, B. (2016). Surrogate modelling for stochastic dynamical systems by combining narx models and polynomial chaos expansions. *International Journal for Uncertainty Quantification*, *6*. doi:10.1615/Int.J.UQ.2016016603

[43] Mairtin, E., Parry, G., Beltz, G., & Mcgarry, P. (2014). Potential-based and non-potential-based cohesive zone formulations under mixed-mode separation and over-closure–part ii: Finite element applications. *Journal of the Mechanics and Physics of Solids*, *63*, 363–385. doi:10.1016/j.jmps.2013.08.019

[44] Marelli, S., & Sudret, B. (2019). Uqlab: A framework for uncertainty quantification in matlab. In *Vulnerability, uncertainty, and risk* (pp. 2554–2563). doi:10.1061/9780784413609.257. eprint: https://ascelibrary.org/doi/pdf/10.1061/9780784413609.257

[45] Masri, S. F., Bekey, G. A., Sassi, H., & Caughey, T. K. (1982). Non-parametric identification of a class of nonlinear multidegree dynamic systems. *Earthquake Engineering & Structural Dynamics*, *10*(1), 1–30. doi:10.1002/eqe.4290100102

[46] Masri, S. F., Caffrey, J. P., Caughey, T. K., Smyth, A. W., & Chassiakos, A. G. (2004). Identification of the state equation in complex non-linear systems. *International Journal of Non-Linear Mechanics*, *39*(7), 1111–1127. doi:10.1016/S0020-7462(03)00109-4

[47] Masri, S. F., Nakamura, M., Chassiakos, A. G., & Caughey, T. K. (1996). Neural network approach to detection of changes in structural parameters. *Journal of Engineering Mechanics*, *122*(4), 350–360.

[48] Masri, S., Miller, R., Saud, A., & Caughey, T. (1987a). Identification of nonlinear vibrating structures: Part i. *Journal of Applied Mechanics*, *54*. doi:10.1115/1.3173140

[49] Masri, S., Miller, R., Saud, A., & Caughey, T. (1987b). Identification of nonlinear vibrating structures: Part ii - applications. *Journal of Applied Mechanics*, *54*.

[50] *Matlab version 9.7.0.11 (r2019b)*. (2019). Natick, Massachusetts: The MathWorks Inc.

[51] Mcgarry, P., Mairtin, E., Parry, G., & Beltz, G. (2014). Potential-based and non-potential-based cohesive zone formulations under mixed-mode separation and over-closure. part i: Theoretical analysis. *Journal of the Mechanics and Physics of Solids*, *63*, 336–362. doi:10.1016/j.jmps.2013.08.020

[52] Milani, G., & Bertolesi, E. (2017). Quasi-analytical homogenization approach for the non-linear analysis of in-plane loaded masonry panels. *Construction and Building Materials*, *146*, 723–743. doi:10.1016/j.conbuildmat.2017.04.008

[53] Miraglia, G. (2019). *Hybrid simulation techniques in the structural analysis and testing of architectural heritage* (Doctoral Dissertation, Supervisor Ceravolo, R., Politecnico di Torino, Doctoral Program in Architectural and Landscape Heritage (31th Cycle)).

[54] Ogrizovic, J., Abbiati, G., Stojadinovic, B., & Frangi, A. (2018). Hybrid simulation of a two-storey two-bay post-tensioned timber frame. In *6th european conference on earthquake engineering, thessaloniki*.

[55] Patel, S., Chourasia, A., Panigrahi, S., Parashar, J., Parvez, N., & Kumar, M. (2016). Damage identification of rc structures using wavelet transformation. *Procedia Engineering*, *144*, 336–342. International Conference on Vibration Problems 2015. doi:10.1016/j.proeng.2016.05.141

[56] Pathirage, C. S. N., Li, J., Li, L., Hao, H., Liu, W., & Ni, P. (2018). Structural damage identification based on autoencoder neural networks and deep learning. *Engineering Structures*, *172*, 13–28. doi:doi.org/10.1016/j.engstruct.2018.05.109

[57] Patterson, M. A., Weinstein, M., & Rao, A. V. (2013). An efficient overloaded method for computing derivatives of mathematical functions in matlab. *ACM Trans. Math. Softw.* *39*(3), 17:1–17:36. doi:10.1145/2450153.2450155

[58] Pei, J. S., Smyth, A., & Kosmatopoulos, E. (2004). Analysis and modification of volterra / wiener neural networks for the adaptive identification of non-linear hysteretic dynamic systems. *Journal of Sound and Vibration*, *275*(3), 693–718. doi:10.1016/j.jsv.2003.06.005

[59] Risorse di calcolo fornite da HPC@polito, progetto di Academic Computing del Dipartimento di Automatica e Informatica presso il Politecnico di Torino. (n.d.).

[60] Robert, C., & Casella, G. (2004). *Monte carlo statistical methods*. Springer Texts in Statistics. Springer New York.

[61] Roberts, G. O., Gelman, A., & Gilks, W. R. (1997). Weak convergence and optimal scaling of random walk metropolis algorithms. *The Annals of Applied Probability*, *7*(1), 110–120. Retrieved from http://www.jstor.org/stable/2245134

[62] Saha, S. K., Sepahvand, K., Matsagar, V. A., Jain, A. K., & Marburg, S. (2016). Fragility analysis of base-isolated liquid storage tanks under random sinusoidal base excitation using generalized polynomial chaos expansion–based simulation. *Journal of Structural Engineering*, *142*(10), 04016059. Retrieved from 10.1061/(ASCE)ST.1943-541X.0001518

[63] Simpson, T., Poplinski, J., Koch, P. N., & Allen, J. (2001). Metamodels for computer-based engineering design: Survey and recommendations. *Engineering with Computers*, *17*(2), 129–150. doi:10.1007/PL00007198

[64] Smyth, A. W. [A. W.], Masri, S. F., Chassiakos, A. G., & Caughey, T. K. (1999). On-line parametric identification of mdof nonlinear hysteretic systems. *Journal of Engineering Mechanics*, *125*(2), 133–142.

[65] Spiridonakos, M., & Chatzi, E. (2015). Metamodeling of nonlinear structural systems with parametric uncertainty subject to stochastic dynamic excitation. *Earthquakes and Structures*, *8*, 915–934. doi:10.12989/eas.2015.8.4.915

[66] Tsou, P., & Shen, M.-H. H. (1994). Structural damage detection and identification using neural networks. *AIAA Journal*, *32*(1), 176–183. doi:10.2514/3.11964

[67] Wagner, P.-R., Nagel, J., Marelli, S., & Sudret, B. (2019). *UQLab user manual - Bayesian inversion for model calibration and validation*. Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland. Report # UQLab-V1.3-113.

[68] Wan, X., & Karniadakis, G. (2006). Long-term behavior of polynomial chaos in stochastic flow simulations. *Computer Methods in Applied Mechanics and Engineering*, *195*, 5582–5596. doi:10.1016/j.cma.2005.10.016

[69] Wen, Y.-K. (1976). Method for random vibration of hysteretic systems. *Journal of Engineering Mechanics Division ASCE*, *102*(2), 249–263.

[70] Worden, K., Becker, W., Rogers, T., & Cross, E. (2018). On the confidence bounds of gaussian process narx models and their higher-order frequency response functions. *Mechanical Systems and Signal Processing*, *104*, 188–223. doi:10.1016/j.ymssp.2017.09.032

[71] Wu, M., & Smyth, A. (2008). Real-time parameter estimation for degrading and pinching hysteretic models. *International Journal of Non-Linear Mechanics*, *43*(9), 822–833. doi:10.1016/j.ijnonlinmec.2008.05.010

[72] Xiuli, D., & Fengquan, W. (2010). Modal identification based on gaussian continuous time autoregressive moving average model. *Journal of Sound and Vibration*, *329*(20), 4294–4312. doi:10.1016/j.jsv.2010.04.018

[73] Xu, X., & Needleman, A. (1999). Void nucleation by inclusion debonding in a crystal matrix. *Modelling and Simulation in Materials Science and Engineering*, *1*, 111. doi:10.1088/0965-0393/1/2/001

[74] Yun, C.-B., & Shinozuka, M. (1980). Identification of nonlinear structural dynamic systems. *Journal of Structural Mechanics*, *8*(2), 187–203. doi:10.1080/03601218008907359

[75] Zhang, H., Foliente, G. C., Yang, Y., & Ma, F. (2002). Parameter identification of inelastic structures under dynamic loads. *Earthquake Engineering & Structural Dynamics*, *31*(5), 1113–1130. doi:10.1002/eqe.151