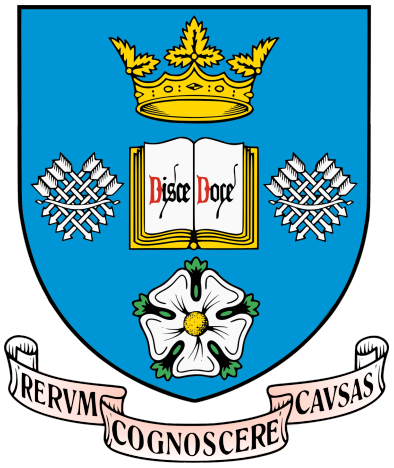
THE UNIVERSITY OF SHEFFIELD

PhD Mechanical Engineering



A data-driven approach to modelling structures

by

George **TSIALIAMANIS**

August 2022

Prof K. Worden, Prof D. Wagg

Abstract

This thesis is focussed on machine-learning approaches to defining accurate models for structural dynamics. The work is motivated by the concept of 'digital twin' and is an attempt to build tools that could be included in a modelling campaign for structures or within the context of a digital twin used for structural health monitoring (or more broadly, asset management).

In recent years, machine learning has provided solutions to many modelling problems, offering solutions that do not require exact knowledge of the physics of the phenomena that are modelled. For structural dynamics this approach can be quite useful, since accurate mathematical representations of the physics of several structures are often not available. Moreover, for performing *structural health monitoring* SHM of structures, data should be used, making machine learning a straightforward way to deal with such problems. The thesis attempts to exploit powerful machine learning algorithms to perform inference for structures in situations that traditional methodologies might fail. The attempts concern several fields of structural dynamics, such as population-based structural health monitoring, modelling under uncertainty and with a combination of known and unknown environmental conditions, performing modal decomposition for structures with nonlinear elements and defining the remaining useful life of a structure within a population of similar structures. The methodologies presented yield very promising results and reinforce the idea that machine learning, in some cases combined with physics, can be used as a tool to define accurate models of structures.

As described in the first chapters of the thesis, an efficient modelling strategy for structures is to use various different models in order to model different parts, substructures or functionalities of a structure. Therefore, an organising technique for all the available data and models that are used is required. For this reason, an *ontological approach* is proposed herein to include all the aforementioned elements and in order to facilitate knowledge sharing.

After defining an organising technique for such a project, some data-driven schemes using novel machine-learning algorithms are presented. Initially, a method to define nonlinear normal modes of oscillations of structures is presented. The method is based on the use of a variation of a *generative adversarial network* (GAN), and proves to provide quite efficient modal decomposition, under specific assumptions. The generative adversarial network algorithm is further explored and an algorithm is developed to define *generative mirror models* of structures. The algorithm is developed to perform in an environment where both known/measured and unknown variables affect a structure. The algorithm, being a generative algorithm, provides a probability distribution of potential outcomes, rather than single-point predictions, allowing probabilistic assessment and planning about a structure to be undertaken.

Moreover, *population-based structural health monitoring* (PBSHM) is addressed using machine-learning algorithms. Performing inference in heterogeneous populations can be complicated, because of the big differences between structures within such populations. In the current thesis, a *graph neural network* (GNN) approach is combined with the transformation of structures into graphs, to perform inference in such a population. The novel GNN algorithm proves able to learn efficiently the interaction physics between structural members and their environment.

Finally, a generative model is used to deal with the problem of estimating the remaining useful life of structures within a population. This algorithm is also a variation of the GAN and is built to generate time series. Using this method, a probability density is defined over the remaining lifetime of a structure, exploiting information available from other structures of the population, for which data are available and which have reached their total lifetime.

The new contribution of this research is the use of currently-state-of-the-art machine learning models for the purposes of structural dynamics. GANs are used for purposes other than their original purpose (artificial data generation), i.e. to perform nonlinear modal analysis and to define generative digital twins of structures. Such models are also used with a view to defining a generative time-series model, which is exploited to estimate the remaining useful lifetime of structures within a population. A second novel type of model that is exploited in the current thesis for the purposes of structural dynamics is that of graph neural networks, which are used to infer the normal condition characteristics of structures within a population.

Table of Contents

Acknowledgements

First and foremost I want to thank my supervisor Keith Worden. The current work was largely made possible by his supervision. I have to thank him for giving me the unlimited freedom to work on any subject that I wanted throughout the years of my PhD and then spending time to correct my work and provide suggestions about it. I really thank him for being a supervisor as well as a friend, for having a pint or a coffee with me (during the pandemic this proved absolutely necessary), which helped a lot to moving and adapting to the UK.

Second, I would like to thank my second supervisor, David Wagg. I want to thank him for being an excellent manager and helping me a lot with my PhD by introducing me to the idea of digital twins and how to move in such a vague field. Our talks mainly during the beginning of my research carrier but also later, although he would call them 'hand-waving', have really helped me.

I would also like to thank the PhD students of the DyVirt program. We all started our PhDs together and we all shared the same anxiety and insecurity and meeting with them and talking about these issues has really helped.

My special thanks go to Eleni Chatzi, who was my supervisor during my secondment at ETH Zurich. I want to thank her for having me there and also because essentially she introduced me to the DyVirt program and an opportunity to pursue a PhD.

I would like to thank everyone in the Dynamics Research Group (too many people to list their names here). They have been great company, quite accepting and have successfully

helped me develop the feeling of belonging to the same group. And, of course, they are great people to go to the pub with.

Finally, I want to thank my family. First, I would like to thank Mara for being a great support, withstanding all my whining, anxiety and pessimism and largely helping me continue with my work. I would also like to thank my parents and my sister, for giving me the opportunity and assistance to move to a foreign country, as well as for being very supportive with my choices, although totally outside their field. Last but not least, I want to mention the pets of the family, which, unknowingly, have helped a lot relieve anxiety.

Chapter 1

Introduction

1.1 Motivation

Infrastructure is an extremely important part of everyday life. In the UK only, for the decade 2018-2028, it is expected that £600bn [1] will be invested on infrastructure for energy, transportation, digital infrastructure, social infrastructure, protection from flood and coastal erosion etc. All these assets are expected to be, first of all safe for the users, but are also expected to benefit society and investors. In order for such structures to yield profit, they should be carefully managed and extensively modelled. Modelling should be aimed at reducing the risk of failure during design as well as during operation of the structures.

Structures often fail, and their failure has very high cost for society. A major cost of structural failure is that of human lives. On the 14^{th} of August 2018, the Morandi Bridge in Genova Italy collapsed, and as a result 43 lives were lost. A study after the catastrophe, showed that in France, a third of the 12000 state-maintained bridges needed repair and that 840 of them were in danger of collapse [2]. In the US it is estimated that maintaining ageing structures will have an economic impact of $10.3tn between 2020 and 2039 [3].

Another field of structures that needs attention is that of wind energy. Electricity generation using wind has increased by 715% in the UK from 2009 to 2020 [4]. The wind energy

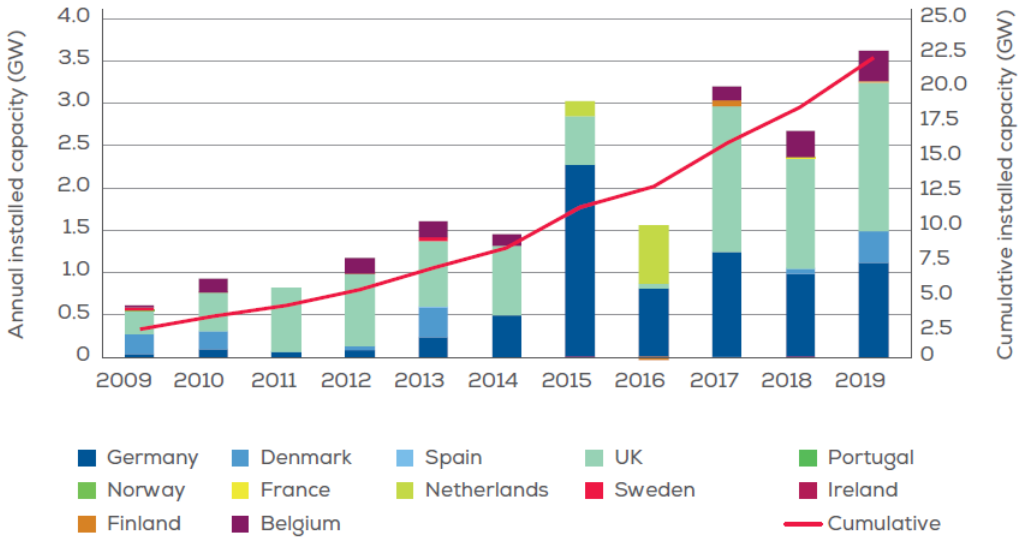


Figure 1.1: Annual offshore installations by country (left axis) and cumulative capacity (right axes) [5]

generated in the UK accounted for 24% of the total electricity generation for 2020. In Europe, in 2019 only, 3623 MW of net offshore capacity were installed; this corresponds to 502 offshore wind turbines [5]. In total, 22 GW of offshore wind capacity is installed in Europe. In Figure 1.1, the installed offshore capacity per year for some countries in Europe is shown, and in Figure 1.2, the corresponding invested amounts are shown for years 2010-2019. The inconsistency regarding the installations in 2016 in the later figure might be because of reductions of the funding by the EU or the governments of the EU members.

Taking into account the "Net Zero by 2050" goals of many countries [6], wind turbines shall be one of the most important aspects of infrastructure in the years to come. Their role is extremely important as a means to achieve the aforementioned environmental goals, as well as investments that should yield profit to society. They shall need careful design, and attention should be paid in order to maintain their condition as efficiently as possible. However, wind turbines face many challenges. They are exposed to extreme environmental conditions and complicated materials are used for their construction. Therefore, designing them and maintaining them becomes a complicated task.

Modelling such structures is necessary in order to make them live up to their expectations; i.e. to perform as they should throughout the years of their operation. During the design

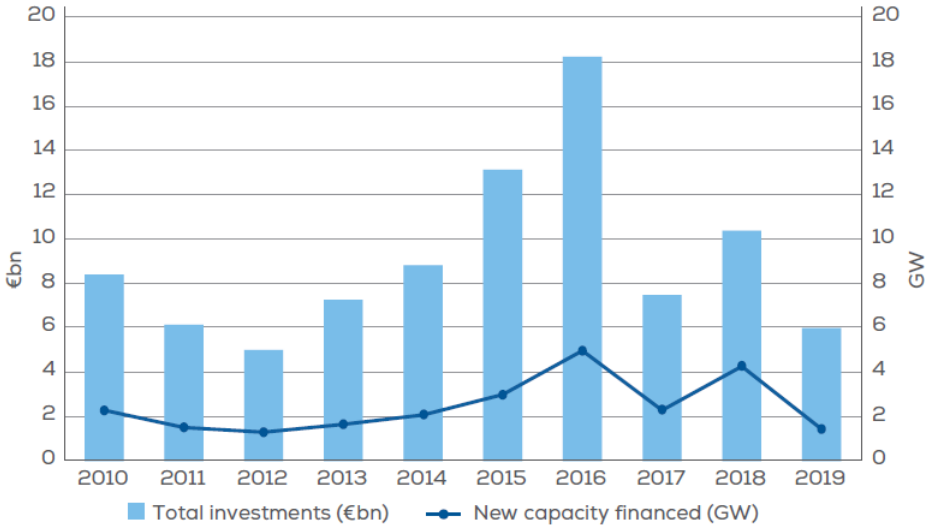


Figure 1.2: Offshore wind energy investments and capacity financed in Europe 2010-2019 [5]

phase of the structures, as well as during operation, modelling is needed to avoid failure and to maximise efficiency. For design, numerous hypothetical scenarios regarding the influence of the environment need to be tested, and how the structure would behave should be analysed. Consequently, engineers need to select the proper features for wind turbines so that they do not fail under the projected environmental conditions. During operation, maintenance should be performed often enough to avoid failure of the wind turbine. Monitoring the condition of the structure is also a means to avoid failure and to locate potential damage early enough to reduce the costs of extensive repairs.

All the aforementioned tasks can be performed using traditional structural analysis tools, such as *finite element models* [7]. However, such methods often fail because of the lack of knowledge one has about the behaviour of the materials and the exact stimuli of the structures by the environment. To satisfy the need of modelling options in cases where traditional methods fail, data-driven methods are often deployed. Such methods are based on observations of the physical phenomena that one wants to model, and look for patterns in them to define relationships between quantities.

Development of data-driven methods has skyrocketed in recent years for many disciplines. Data are becoming available more widely and their exploitation is being introduced into numerous fields. Major applications include imitating human behaviour and automating

procedures that would normally be performed by humans. The fields of *machine learning* [8–10] and artificial intelligence (AI), have offered great benefits to society and facilitated many aspects of everyday life. The independence of these methods from extensive knowledge of the entities which they are called to model, makes them extremely convenient, and provides quick and sometimes very successful solutions to problems, which may have been really difficult to solve over the years.

For structural dynamics, such methods can be used to perform analysis of structures, as well as the functionalities of *structural health monitoring* (SHM) [11], that are much needed for wind turbines. In the current work, several such methods are developed and presented. The aim of these methods is to provide new modelling capabilities and to define SHM methods that can assist in maintaining the healthy condition of structures.

1.2 Brief introduction

Modelling of systems has been a pursuit of researchers for many years. The desire to model systems is brought out by the need to avoid danger, and to efficiently manage available resources. Another reason why modelling of systems is studied is because it costs much less than performing experiments on the actual systems and that simulations can be repeated for many different values of the parameters of the systems. If studied more thoroughly, the need of humans to predict the behaviour of various objects, or the outcome of situations, is quite primal. Humans have been trying to predict the future in many situations in the past. Many groups (usually affiliated to some religion), throughout human history have had the belief that some humans are able, via some clairvoyant procedure, to predict the future.

Naturally, most of these claims about future-predictive abilities have been proven false. However, the need to predict the outcome of future situations, or at least to provide some insight about the outcome, still exists. Everyday life activities have become safer than in previous years, and this is largely because of efficiently modelling systems. In almost every aspect of life, some system is involved, which has been studied and modelled in order to ensure its efficiency and safety. A large class of such systems is that of structural systems.

Structures participate in almost every aspect of modern life. Life in cities includes interaction with many different types of structures, including buildings, cars, roads, machinery etc. Although such structures have existed for a long time, simulation of their behaviour has not been always feasible. Today, the majority of these structures are modelled in order to be deployed into the world with maximum safety. Before acquiring the knowledge and the tools to analyse and predict the behaviour of these structures, humans mostly used personal intuition and experiences in order to make them as safe as possible.

Years of studying the physics of these structures, their materials and their environment together with modern technology have created the proper conditions in order to build very efficient and accurate models of them. These models are used widely during the design phase of the structures. Engineers try to model potential situations, in which underdesign structures might be found, and design it in order to avoid failure and to maintain the quality of the structures' service to the users.

While modelling methods are developed further in order to more accurately simulate structures, materials which are used in order to build the structures, are becoming more complicated. As mentioned, an important tool to model structural behaviour is studying of the physics of the materials and the structures. However, some new materials have not been studied yet. Furthermore, more complicated structure layout and connectivity tends to cause problems in the modelling procedure, since interactions between different structural members might cause alterations in the behaviour of each member.

Because of the lack of physical knowledge and the difficulty in acquiring it, in many scientific fields, data-driven modelling has emerged. Using data to define models of some phenomenon is a technique that has, of course, been used in many applications, before it was considered a discipline of its own. For example, in order to define Young's modulus of materials, data from experiments were used and some linear or otherwise model was fitted according to the observations. Although the concept of the scientific method is fundamentally a data-driven process -i.e. observations (data), hypothesise (model), test (data)- in the current thesis data-driven models refer to models which do not require development of physical theory in order to define the model (i.e. to hypothesise).

In recent years, the field of data science has been developed even further. The discipline

of *machine learning* is based on learning relationships between data mainly based on observations and using no knowledge about the underlying connections. Such methods are used to replace humans in tasks because of the efficiency and the accuracy the methods have in performing their goal, and numerous applications reveal that the algorithms are indeed able to substitute human involvement in several tasks. Although the algorithms and methods of the machine learning discipline which get the most promotion are focussed on performing tasks -such as image recognition or natural language processing- which are functionalities of humans, such algorithms are also exploited quite successfully for the purposes of physical sciences and engineering.

As mentioned, the behaviour of some materials is quite complicated and their modelling solely via physical knowledge is difficult. In such cases, it its natural to seek solutions via *uninformed* data-driven methods. The materials can be tested and studied in a laboratory, or directly from the field of their deployment, and using these data one can build machine learning models, in order to model their performance. Such methods could be considered 'lazy' in the sense that one does not spend any time or effort to define the underlying physics of the modelled phenomenon. However, building such models requires extensive study, since in order to deploy it in action, one needs to be completely sure that it will perform as expected.

Another emerging field of study is that of defining *digital twins* of systems. The term digital twin probably comes from the desire of humans to create exact replicas of physical systems in the digital world, with emphasis on the word 'exact'. The desire comes from the need to acquire a digital copy of a real-world entity, which humans will be able to analyse and make inference about the real entity, to monitor the condition of its real world counterpart, to predict the future of the physical twin and even in hypothetical scenarios, to see how the physical twin would behave. All these potential uses of such digital twins might be motivated by science fiction, but they would indeed be very useful in modern life. By using such simulation tools, engineers would be able to have a holistic view of a structure and be able to maximise safety and profit from it. Therefore, a lot of effort has been spent on defining such models or objects.

As expected, the digital twin should be 'aware' of the situation of the physical twin, and therefore the real world structure should be monitored. As a consequence, data

are acquired form the real-world structure. Data acquisition makes the use of machine learning, and data-driven modelling in general, indisputably connected to the philosophy of digital twins. As a matter of fact, data-driven modelling may be even more powerful than traditional physics-based modelling techniques, when it comes to modelling complicated systems with many *uncertainties* and when one has to deal with varying length and time scales.

A field where machine learning has been the major strategy to follow is that of *structural health monitoring* (SHM). The discipline of SHM is the subfield of structural dynamics, which as its name states, is targeted at observing the condition/health of a structure. Maintaining the healthy condition of structures is naturally wanted by engineers and beneficiaries of the structure. Moreover, SHM is about real-time monitoring of structures, as well as predicting future states and potential future damage of the structure. Therefore, SHM can be integrated within the digital twin concept and be an important part of it.

Already, by including SHM into the digital twin, one can understand that it becomes difficult for the digital twin to be a single holistic model of the real-world entity. Therefore, an approach to achieve the creation of a digital twin could be the collection of many models. As will be discussed further in this thesis, the digital twin can actually be a *collection* of models, data, structural members, structures and modelling techniques. Although a digital twin has no universally agreed definition, the goal of such an object is to model as accurately as possible, the current, future and potential state of a system. The states may refer to the health condition of the structure, its behaviour under external excitation, control parameters or even action policies that have to be followed in order to increase the safety and the economical benefit from the structure. In the current thesis, several methods with such an intent are described.

1.3 Overview

In the current work, data-driven models for structural modelling are presented. The aims are based on the admission that a digital twin is a collection of models, data etc. as mentioned earlier. The models used herein are *neural networks*, a type of model that has

dominated the machine learning discipline in recent years. Variations of classic machine learning models are used, aiming to match the properties of the algorithms to the physics of the phenomenon that is studied, and the nature of the data.

In Chapter 2, an introduction is given to data-driven modelling. An introduction to *mirror* models is given, together with arguments about the use of generative models as mirrors. Some basic theory about machine learning is presented and of neural network algorithms, which has dominated the discipline of machine learning. Some aspects of the algorithm are discussed, and a new type of neural network, the *generative adversarial network* (GAN) [12], is presented and explained.

In Chapter 3, structural health monitoring (SHM) is discussed. The various levels at which SHM is performed on structures are presented and some examples of applications are discussed. Moreover, the new discipline of population-based SHM (PBSHM) is also introduced. The reasons, for which this novel discipline is developed and some initial works on the subject are presented.

In Chapter 4, an ontological scheme is presented for structural health monitoring. The ontology is considered an appropriate way of organising data and knowledge about a topic. Therefore, its definition including the data and the methods used in an digital twin scheme or an SHM campaign shall prove useful when one needs to use them. The ontology is built in a way which can help it include different structures and structural members allowing potential transfer of knowledge between them, and the extension of knowledge. Such an ontology can be of great importance for multidisciplinary applications, since it facilitates knowledge sharing and can be used as a guide for implementation of the methods in terms of creating software.

In Chapter 5, machine learning methods based on the generative adversarial network (GAN) are presented. The newly-introduced method of adversarial training for generative models is exploited for the purposes of structural dynamics. A method based on machine learning that achieves a 'modal' decomposition of the dynamics of structures is presented. Such a decomposition is very important when modelling structures, because it gives the users ways of studying independent components of the movement, rather than the whole movement of the structure, and making inference based on simple quantities.

In Chapter 6, another variation of neural networks, which considers graphs as inputs to the inference algorithm, is exploited in order to define the first natural frequency of structures within a population. The natural frequencies of structures are important for the procedure of monitoring, but also during the design procedure. The proposed algorithm is data-based and seeks to exploit an algorithm that is able to take into account the layout of structures. A great advantage of the algorithm is that it is able to perform efficiently in a quite diverse population of structures.

In Chapter 7 a GAN is used to generate artificial structural data. Taking into account the original goal of such an algorithm, which was to generate artificial images, one can use it to generate artificial data regarding the behaviour of a structure. Rather than generating data for conditions for which data have already been recorded, it was chosen to use a variation of the algorithm and learn how to generate data that correspond to conditions of the environment for which data have not been recorded.

In the same chapter, the same algorithm is also exploited in order to define *generative* digital twin models, or mirror models of a structure. Since digital twins might be used to model structures under uncertainty, it is natural to seek models that can take into account such uncertainties. The variation of the GAN is able to learn transformations of probability distributions according to various parameters, and can therefore be used to define such a generative model. The algorithm is compared with the widely-used *stochastic finite element method* (SFEM), which does not perform very well when the finite element formulation physics do not match the physics of the structure. A combination of the two algorithms -a *hybrid* model- is also presented, and is found to slightly outperform both approaches.

In Chapter 8, the problem of damage prognosis is addressed via the use of another variation of GAN. The approach is based on a population-based scheme, and the algorithm is used in order to learn the physics of some type of damage, common in the structures of the population. The proposed algorithm is a generative model and as a result, it provides a probability over the remaining useful life of the structures rather than a single point estimate. The method is tested on a simulated dataset, and the results reveal that it is able to learn the damage evolution process and to provide estimates of the remaining useful life, which are more confident than using a single probability density function according

to the total lifetimes of the structures of the population.

Finally, in Chapter 9, conclusions are drawn regarding the presented methods, and future research is proposed in order to further develop and validate current methods, but also regarding the definition of new methods for digital twins.

Chapter 2

Data-driven modelling

2.1 What is a model?

A model means different things for different disciplines. A model for art or architecture can be a three-dimensional miniature of an object, or even plans of a building; for some commercial company, it could mean the different versions of a product; for graphic designers, a model is a mathematical representation of real or imaginary objects via some software. For engineering, a model is often a mathematical description or a representation of some phenomenon or some object. Models in engineering are created in order to describe objects -such as structures- and to share their design with others, but also in order to formulate the conceptualisation of the physics of how these objects behave and react under certain situations, even for situations that one might not be able to evaluate with an experiment.

The latter type of model has been the object of research for many years in many disciplines including: mechanical and civil engineering, computer science, social sciences, biology, chemistry, medicine etc. In every discipline, different objects are modelled and inferences are made about various aspects of these objects. Similarly to creating a model for architecture or art, a model of the aforementioned disciplines offers a way to have supervision over an object or a phenomenon. For example, a three-dimensional miniature of a larger structure gives the architect the chance to rotate it, examine it, and have a better

opinion about how it would look like after it is 'embodied'. Similarly, having plans of a building reduces the information given by the model, but offers the chance to pay attention to different aspects of the structure than in the case of observing some three-dimensional miniature. Respectively, models in more technical disciplines offer an abstraction of the modelled object and phenomenon, with a view to better understanding them and also to remove redundant information and focus on aspects of the phenomenon that are of interest of the user of the model.

A major use of models in technical disciplines is to perform inference in hypothetical ('what if') scenarios. Models are created using assumptions, limitations and understanding about the *underlying physics* of the objects and the phenomena that are modelled. The creators of the models use observations of the phenomena, understanding and intuition, in order to build mathematical models which describe interactions of objects and predict how the system behaves, given some stimulus or input. These models are often used during design of objects in order to test what-if scenarios and avoid unwanted situations. They have offered great capabilities during the design of structures, since one can use them to predict how a structure (which has not been materialised yet) would behave under the conditions of the environment in which it will operate. Furthermore, such models are used to model existing objects and make inferences on how they might behave given different inputs. It becomes clear that models are an attempt to predict the future, combining understanding of phenomena and observations.

The performance of the models is based on the creators' understanding of the phenomena and the objects that are modelled. During the formulation of the mathematical model, the creator translates his understanding of the underlying physics into mathematical operations. It is common that one does not fully understand the underlying physics or formulates this understanding mistakenly. This lack of knowledge is a common problem when one models a phenomenon and is referred to as *epistemic uncertainty*. Sometimes, complete knowledge of the underlying physics is not required in order to build accurate models, since only a subset of the operations of the phenomenon might need modelling in order to get the desired results. In every case, the predictive capabilities of the model depend largely on whether one has captured correctly every part of the phenomenon that one tries to predict via the model. Paradoxically, some models may predict quantities of

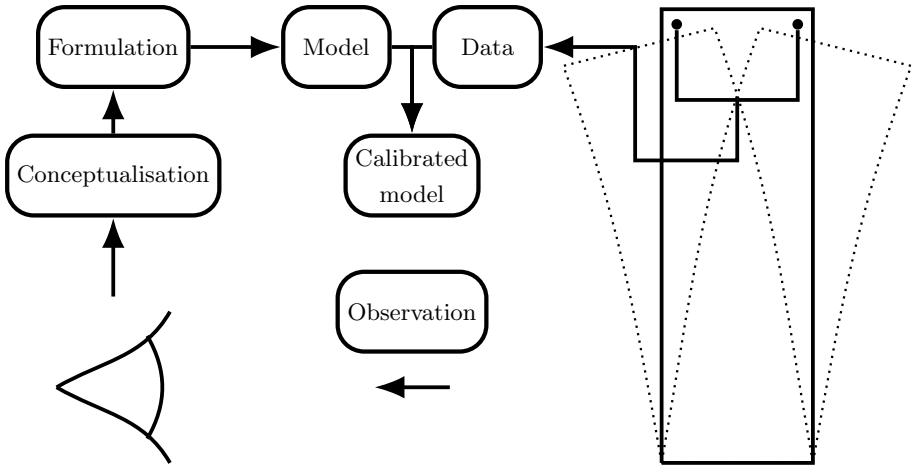


Figure 2.1: Schematic representation of the physics-based modelling procedure of some phenomenon.

interest quite well even if they have not at all capture the underlying physics, e.g. the earth-centered solar system model.

The procedure of understanding, formulating and developing models is considered the *physics-based* way of modelling. A schematic representation of the procedure followed to create a physics-based model of some phenomenon is shown in Figure 2.1. Physics-based models have been monopolising the model-development discipline for a long time. Most models are based on physical theories, which in turn are based on understanding of fundamental operations of nature, or in general, the mechanisms that dictate the phenomenon of interest. A major example of such models in structural engineering is the *theory of elasticity* [13]. The specific theory provides a tool to analyse bodies under external loading and to define the stresses and strains within their volume. Elasticity theory is the basis for many methods used in structural engineering to define models and perform analysis or design of structures. A physics-based method that has been widely and successfully used in the aforementioned discipline is the *finite element method* (FEM) [7]. The finite element method essentially allows the reduction an infinite-dimensional problem of elasticity into a finite set of equations that need to be solved in order to define the state of a structure.

Finite elements are widely used in civil and mechanical engineering and provide efficient solutions when one wants to study the behaviour of a structure. Of course, linear theory of materials does not suffice to explain the behaviour of every structure. It is common that materials have nonlinear stress-strain or load-deformation relationships, strain-rate

effects, time-dependent effects, thermal effects, radiation effects, and so the finite element formulation has been progressed in order to include and explain such behaviours. However, finite elements are not always the solution to efficiently modelling structures. Quite often, one has no knowledge about the exact behaviour of some materials and, even if experiments are carried out in some laboratory to investigate the behaviour, the environmental conditions to which the material will be exposed during deployment of the structure may largely alter its behaviour.

The problem of missing information about the underlying physics of some structure becomes evident when *calibration* of a physics-based model is attempted. Calibration is the procedure of defining the model parameters of a model of an existing system so that the predictions match one's observations of the system. The procedure of calibration can vary from being as simple as an exhaustive search over some parameter space to being more sophisticated; for example, using gradient descent methods [14–16] or genetic algorithms [17, 18]. Matching of different quantities is attempted when one tries to calibrate a model according to observations/acquired data from a system. For example, in the case of structures, attempts might be made to match some spectra or the maximum acceleration, or the first n natural frequencies of the structure and the model. If such a model is created using physics that do not correspond to the underlying physics of the structure, calibrating the model could be possible and the model might predict quite accurately for the calibration data, but its predictive capabilities for other regimes could be minimal. Epistemic uncertainty will be discussed further in next sections.

A common strategy to follow when one is concerned that the underlying physics of some phenomenon are not fully understood, is to use *data-driven* models. Such models do not require understanding of the underlying physics and are based only on the observation of the phenomenon which is to be modelled. This time, the modelling procedure does not require conceptualisation of the phenomenon and mathematical formulation of the theory; a schematic representation, is shown in Figure 2.2. Data-driven models are defined based on data, which are collected from some kind of sensor. The data should be quantities that are varying during the phenomenon and affected by it. These quantities should in general have some causality relationship. Variation of some of the quantities should also mean variation of the others; i.e. correlation. Moreover, in order for the techniques to be

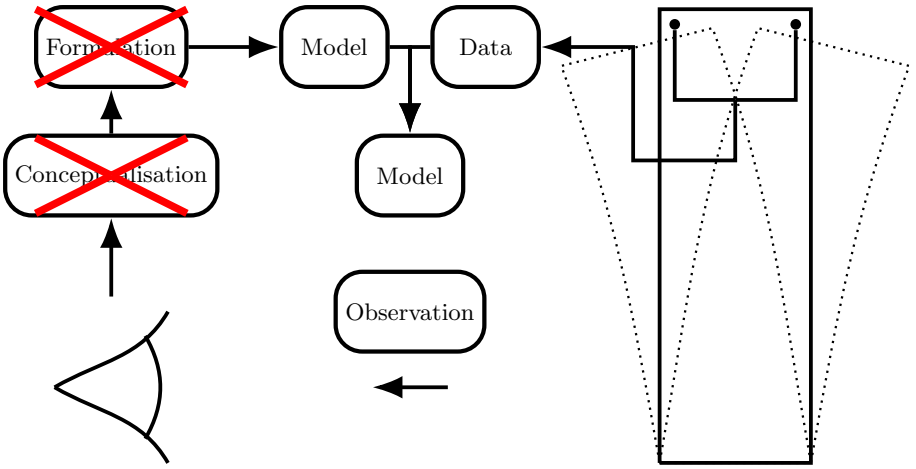


Figure 2.2: Schematic representation of the data-driven modelling procedure of some phenomenon.

successfully applied, every variable that causes variation of the quantity that one wants to predict, should be recorded and used to define the data-driven model.

Data-driven models are often referred to as *black-box* models. This name comes from the fact that it is common that one has no knowledge about how the model performs inference, i.e. the model is not transparent. A common approach to using such models is to collect every possible variable that is monitored during a phenomenon and use it as an input to the black-box model. For every set of input variables, the quantity, for whose prediction the model is created, is also monitored and is considered the output of the model. Subsequently, the model is *fitted* to the data. Fitting the model is also called *training*, and can be a deterministic procedure; for example, in linear regression, or a stochastic one, as in the case of neural networks (described in next section). The result of fitting is a model that has *learnt* some relationship between the input and the output quantities. Having fitted the model, one expects that it will be able to *generalise*, i.e. to approximate the quantity of interest for "unseen" values of the input variables.

Data-driven methods, and more specifically *machine learning* [8, 9], offer quite powerful approaches when knowledge about a phenomenon does not exist. For structural dynamics, a major example is not knowing details about the behaviour of some material. Given that a material is nonlinear [19], for example, the exact type of nonlinearity should be defined in order to properly model the behaviour of structures that have members made from the specific material. Even if this result is achieved, temperature or loading variations may

change the behaviour of the material.

Machine learning offers generic models that can be calibrated on almost any dataset, which is based on some rules and relationships between the input and the output quantities. Using such models, the demand for understanding the underlying physics of phenomena, systems and members of the system is bypassed. Models are created only based on data; it has been proven that machine learning algorithms have great predictive capabilities (mostly interpolation capabilities), in some cases even outperforming classic modelling methods or even human performance in tasks which are based on imitating human behaviour (image recognition [20], natural language processing [21]).

Modelling methods based on data offer great capabilities especially when *big data* [22], are concerned. When vast quantities of data need processing and models need to be created based on them, classic modelling techniques may be computationally inefficient or even intractable. Machine learning also offers ways to select the best data in order to build the models (such techniques are included in the *active learning* discipline [23, 24]). Another advantage of using such methods, when large quantities of data are available, is the speed of inference. Machine learning methods are able to perform inference on large quantities of data very fast, since their operations are usually simple matrix addition and multiplication.

From what has been discussed so far, the impression might have been created that data-driven methods is some kind of panacea to the problem of modelling systems (of course this might be a result of the author's preferences and the fact that the current work is a thesis on data-driven methods for structural dynamics). However, machine learning, and data-driven modelling in general, are far from being the cure to every problem one might encounter when trying to model a system.

As mentioned, when using such models, the step of defining the physics of the problem, the conceptualisation, is bypassed and this bypass is accompanied by drawbacks. First and foremost, in order to define any data-driven model, naturally, data should be available. A data-driven model is as good as the data used to define it. The quantity of data required to define such models tends to be larger than the quantity required when a physics-based model is to be calibrated. Secondly, data-driven models lack *extrapolation capabilities*

(something that quite often happens with physics-based models as well). Training a data-driven model on data coming from a specific distribution, can only ensure the ability of the model to perform inference on data coming from that distribution. In contrast, physics-based models are based on physics that can be global (although quite often the assumptions made in order to define the model are restrictive, e.g. small deformations for various beam models) and can dictate the behaviour of the system in-hand under every potential input or condition of operation. This difference can be thought as the element that equalises the "energy" one needs to offer in order to define either a physics-based or a data-driven model. In the case of data-driven models, there is no requirement of knowledge of the underlying physics, and in return, the predictive capabilities of the model outside the distribution of the training data are essentially minimised. Thirdly, one has no supervision on how a black-box model performs inference. Evaluation of the performance of such models is performed in terms of percentage of accuracy or some error metric over a dataset of unseen data. These metrics may often mislead and create the impression that a model performs very well, when in fact the model may be making elementary mistakes. Such models may have good accuracy, but may also be quite unstable in their predictions, and have catastrophic results if used for decision making without any human intervention.

The aforementioned problems of machine learning, and many others, have been the subject of research of many researchers for many years. Especially in structural dynamics, an attempt to deal with these problems is to try and combine the advantages of physics-based models (white-box models) and data-driven models (black-box models). This strategy leads to the creation of hybrid or *grey-box* models. Creation of hybrid models can be attempted by inducing physics in the inference procedure of a data-driven model, by using black-box models to correct the predictions of a physics-based model which is not performing as expected, by favouring specific solutions of the black-box model according to one's knowledge of the underlying physics or by using some prior knowledge about some quantities, in order to affect the training procedure. At the same time, these techniques are preferred by engineers, since they are more interpretable than plain black-box models, and malfunctions can more easily be addressed and fixed.

Another important aspect of applying data-based models is that of *feature engineering*. Selection of appropriate features of some phenomenon before training a machine learning

algorithm is a very important procedure, and may greatly affect the performance of the model. The most basic requirement of feature selection is to define quantities as inputs and outputs that have a cause and effect relationship. A data-driven model is able to perform both forward and inverse inference between such quantities if the relationship exists, otherwise the variables are just random and no model can define a relationship between them. During feature selection, the user can induce some physical knowledge in the model by preprocessing the data in a convenient way. For example, instead of trying to model raw acceleration data coming from a sensor, one might try to model an aggregative quantity of the data, such as the mean or maximum acceleration or the *power spectral density* of the signal.

Various attempts have been made using many different modelling techniques and algorithms to model structural systems. In some cases, the procedure has been simple, or some simple model yields accurate results and more sophisticated approaches are not needed and therefore not attempted. There are cases, however, especially when new materials are used, or have not been sufficiently studied, where more sophisticated approaches are needed. In some cases, physics-blind methods, i.e. data-driven methods are needed, since the physics of the materials are extremely complicated to conceptualise. As materials evolve and structures become more complicated, more accurate models are needed, and sometimes the need emerges to model structures in real time to make decisions. To deal with such issues, the field of *digital twins* has emerged. The term could mean many different things and serve many different expediencies according to the needs and goals of the users of the model. In the next section, the term 'digital twins' is discussed, an alternative term is presented and some terminology about such models is given.

2.2 Digital twins - mirrors

The need to model systems with the highest possible accuracy, has lead to the creation of the concept of a *digital twin*. A digital twin could essentially be defined as a digital copy of some system, which is called the *physical twin*. Such a model would, ideally, react the same way to inputs as the system that it models. The whole procedure of defining a digital twin would naturally include calibration of the model according to data acquired

from the system for different inputs and for varying parameters affecting the behaviour. A digital twin should be able to yield accurate and trustworthy predictions, even in real time, about the current and future condition of the structure. The goal of such models is to operate the system safely and to maximise profit. The term 'digital twin' has been used in many scientific disciplines and they have been a highly desired object especially in industry. There have been examples of attempts to define such tools in manufacturing [25, 26], control systems & the internet of things [27], smart cities [28], social networks, management [29], structural engineering [30] etc [31, 32].

An initial approach to defining a digital twin would be to follow a holistic approach and try to simulate the whole system via a single model. Such a strategy has been followed in modelling structures using the finite element method [7]. Whole structures can be modelled using FEM models, taking into account interactions between various structural members as well as between environmental conditions and the parts of the structure. Although such approaches are able to account for interactions between different elements of the system, even in a recurrent way, they often fail.

The main reason that causes such models to fail to predict accurately the behaviour of systems, is that the relationships between the inputs and the outputs of the system are often quite complicated or the inability to practically model behavior over widely varying length and time scales. In order for a model of the whole system to be defined, extended knowledge about the mechanisms of the modelled system is required. In the case of structures, the knowledge refers to the materials of the members, the loading, the degradation processes of the materials through time, etc. This knowledge is the physics of the structure. By building models of the whole system, errors -because of insufficient knowledge of the physics of members- accumulate and affect the final predictions. In addition, multi-scale and multi-physics makes the creation of a single model very difficult and simulation and calibration of such models are usually computationally-expensive procedures increasing the operation cost.

For complex engineering applications, a holistic model of a system may not even be definable. A simpler modelling solution is to define models that just govern limited environmental and operational regimes or to define smaller-scale models for different parts of the system [33]. A first advantage of modelling smaller parts of a structure is that the

need to model interactions between structural members is reduced. A second advantage is that more modelling techniques are made available in order to make predictions. Techniques like machine learning [8, 34] and *deep learning* [10] can be exploited to increase the modelling capabilities of the digital twins. Data-driven models are structurally able to define relationships based on acquired data and their impact in the modelling procedure of structures is of great importance.

By splitting the modelling procedure into smaller models that perform inference on a specific part or a specific functionality of the structure, the digital twin becomes something more than just a single model. A digital twin can be a collection of submodels, each one modelling a different aspect of the structure. This strategy allows inclusion of models performing different functionalities than simply predicting accelerations or displacements of the structural members. An important functionality that can be included in the modelling process is that of structural health monitoring [11], which will be presented in later chapters. Different submodels may be directly focussed on maximising the profit of the structure's operation; for example, by modelling the generated energy from a wind-turbine.

The submodel approach appears to be more appealing and applicable than the holistic approach; however, uncertainty affects the modelling procedure even for smaller members of the system. Uncertainty has two different main forms -aleatory and epistemic uncertainty. Aleatory uncertainty refers to phenomena that are inherently random. Varying material parameters because of imperfections during the manufacturing process is a common source of aleatory uncertainty. Variations of the loading might have some deterministic basis, but trying to predict them could be really complicated, so they are also often considered as random. This type of uncertainty cannot be reduced and has to be taken into account when one models a procedure that is affected by such randomness.

The second type of uncertainty -epistemic- refers to lack of knowledge about the mechanisms that are modelled. If one does not completely understand the physics that need to be included in the model, then errors will be observed in the predictions, in comparison with the real behaviour of the system. In structural engineering, a common source of such uncertainty is the exact type of *nonlinearity*. Structures are often modelled as linear, which may yield accurate enough predictions in order to design and perform maintenance, because structures are often designed to operate in the linear region of their materials,

for safety reasons. However, nonlinearity is present [19] and affects the performance of models. Another source of such a type of uncertainty is to not know exactly which parameters affect the quantity of interest. In machine learning, these parameters are also called *lurking variables*.

In order to deal with issues of uncertainty during prediction, the use of *generative models* as mirrors is proposed in the current work. Generative models are models that either provide a probability distribution or are able to generate samples of the quantities of interest. Generative models, such as generative adversarial networks [12], learn to generate samples according to acquired data and are able provide predictions taking uncertainties into account. An approach to the digital twin concept taking into account uncertainty seems more complete: on the one hand, it offers a way to deal with epistemic uncertainty, treating it as variations in the observations, and on the other hand, takes aleatory uncertainty into account. In a later chapter, a completely data-driven approach to defining such a model will be presented.

2.2.1 Mirror terminology

In the current work, a different term for the aforementioned type of model is used. The preferred term is that of a *mirror* model of a system, and more specifically a mirror model (or just a mirror) of a structure; the term was introduced in [35]. A mirror refers to some physical structure (physical twin) S, which has N different states $\underline{s} = \{s_1, s_2, ...s_N\}$. At the same time, the structure exists within an environment E, which has N corresponding environmental states $\underline{e} = \{e_1, e_2, ...e_N\}$. Different operational conditions may also be included in $\underline{e}$. The environmental states, of course, affect the structural states and play a very important role in defining a proper mirror of the structure. Such environmental states might be the temperature of the environment, the humidity, the wind speed and direction, the ground conditions of the foundation etc. The corresponding structural states can be anything that one monitors during the operation of a structure, such as displacements, accelerations, strains etc.

The approach followed herein is to use models predict some quantities of interest instead of defining a global model for a system. The collection of these quantities comprise the

context C of the mirror, given by,

$$C = \{s_j^C \in \underline{s}, e_i^C \in E; i, j\} \tag{2.1}$$

where s_j^C are the quantities to be predicted or the *response* or *predictive context* and e_i^C are the environmental parameters that affect the quantities of interest and therefore the prediction process or the *environmental context*. The context C defines clearly the capabilities of the mirror that will be defined. It clarifies, first of all, the quantities that can be predicted using the specific mirror. Such a clarification is fundamental when one needs accurate predictions about some aspect of the structure, since a common phenomenon is to use models to perform inference about specific quantities but are built according to their performance on different ones. The second important aspect of the context is the specification of the environment into which the mirror can be used. The environment defines the domain that the model has been tested on and proved to perform efficiently. Using such a model for different environmental parameters would be similar to trying to extrapolate, which models are often not able to perform.

According to the ability of a mirror to predict the behaviour of some structure, they can be considered either ϵ-*mirrors* or α-*mirrors* [35]. The two different types of mirrors refer to different ways of measuring their performance compared to data acquired from a structure. An ϵ-mirror needs the definition of a proper distance metric d^C so that,

$$d^C(\underline{p}^C, \underline{r}^C) \leq \epsilon \tag{2.2}$$

where $\underline{p}^C$ are the predictions of the model within the context C and $\underline{r}^C$ are the measured data from the structure for the same context. The parameter ϵ is the tolerance of the mirror and characterises its performance within the context C. Essentially, the tolerance ϵ defines how close are the mirror's predictions to the observations. For the case of generative models, which generate distributions of data, the metric $d^C(\underline{p}^C)$ should be a probability density function metric. The tolerance ϵ in that case describes how close the real and the generated distributions of data are.

The second type of mirrors, the α-mirrors, refer explicitly to generative models and their ability to include the observations within a predefined interval. More specifically, the

function that describes the functionality of an α-mirror is given by,

$$P(r_i^C \in [\bar{m}_M^C - \alpha\sigma_M^C, \bar{m}_M^C + \alpha\sigma_M^C]) = f(\alpha) \tag{2.3}$$

where M is the generative model used as a mirror, $\underline{r}^C$ is the observation, $\bar{m}_M^C$ is the mean value of the distribution of samples generated by M, σ_M^C is the corresponding standard deviation, P is the probability of $\underline{r}^C$ to fall into the defined interval and α is a parameter that controls the width of the interval. Clearly, the wider the interval, the more probable it is for a sample to fall into it and therefore the probability $P(\alpha)$ becomes larger. An α-mirror is a simpler way of evaluating a mirror model, since one could use it by adjusting the parameter α in order to get prediction intervals, so that the probability of exceeding the bounds fits one's needs.

As far as the environmental parameters are concerned, they can be separated in two categories; the first is the *known* or *controlled variables* ($\underline{e}_c^C$). This group includes every parameter that one has records of and influences the behaviour of the structure. They are the variables that will most definitely be used as inputs to the mirrors in order to predict the behaviour of the structure. The second type of environmental variables are the *unknown* or *uncontrolled variables* ($\underline{e}_u^C$). These variables are often present during the operation of a structure and could refer to uncertain parameters, such as noise or some stochastic material property, or even environmental parameters that one does not monitor or does not know, that affect the performance of the system.

The unknown variables should also be included in the prediction procedure. To do so, a generative model M_u^{EC} should be used in order to provide the best estimate $\hat{\underline{e}}_u^C$ for $\underline{e}_u^C$, as far as the stochastic parameters are concerned. Consequently, a mirror M, which in the current work is also considered a generative model, provides predictions given by,

$$P_{\underline{p}} = M(\underline{e}_c^C, \hat{\underline{e}}_u^C = M_u^{EC}). \tag{2.4}$$

where $P_{\underline{p}}$ is the probability density function of the quantity of interest.

Collecting the elements described so far, the term of *virtualisation* can be introduced [35]. A virtualisation V^C is defined within a context C as,

$$V^C = (M^C_{\epsilon_1}, M^{EC}_{u,\epsilon_2}) \tag{2.5}$$

where $M^C_{\epsilon_1}$ is a generative model, which is considered an ϵ-mirror with tolerance ϵ_1 and M^{EC}_{u,ϵ_2} is the generative model used in order to model the uncontrolled parameters of the environment and is also considered an ϵ-mirror with tolerance ϵ_2. As described by equation (2.4), $M^C_{\epsilon_1}$ takes inputs generated by M^{EC}_{u,ϵ_2} in order to provide predictions for the quantity of interest. The variations that M^{EC}_{u,ϵ_2} describes, account for a part or all the variations in the predictions, which may be displacements, accelerations, natural frequencies etc.

A schematic illustration of the modelling strategy described so far is shown in Figure 2.3. As shown, a generative model M^{EC} is used in order to provide the best estimates $\underline{\hat{e}}^C_u$ of the uncontrolled environmental variables $\underline{e}^C_u$. At the same time, data $\mathcal{D}$ are acquired from the physical twin S. A part of them is considered to be the training data $\underline{S}^c(\mathcal{D}_{tr})$ and is used to calibrate the model M that will be used as a mirror. The mirror is itself a generative model and is informed by M^{EC} in order to make predictions regarding quantities of interest. The predictions are then compared to those of another part of the data, the testing data ($\underline{S}^c(\mathcal{D}_t)$) in order to define the parameter ϵ_1 and the function $P(\alpha)$, which characterise the ability of M to serve as an ϵ-mirror or α-mirror respectively. The two latter elements are the mirrors identity, as far as its predictive capabilities within the context C are concerned. Finally, the model can be used to provide predictions $\underline{p}^C$.

2.3 Machine learning

The discipline of machine learning has been an emerging field, and has already dominated the modelling approaches in many fields. The convenience of the various methods of the field lies in the automation of the definition of models, i.e. without the need of understanding the exact underlying mechanisms of the problem. Modelling is based exclusively on data and on finding relationships between some input and some output variables, a procedure called *learning* (as discussed later, this physics-blind approach is not always the

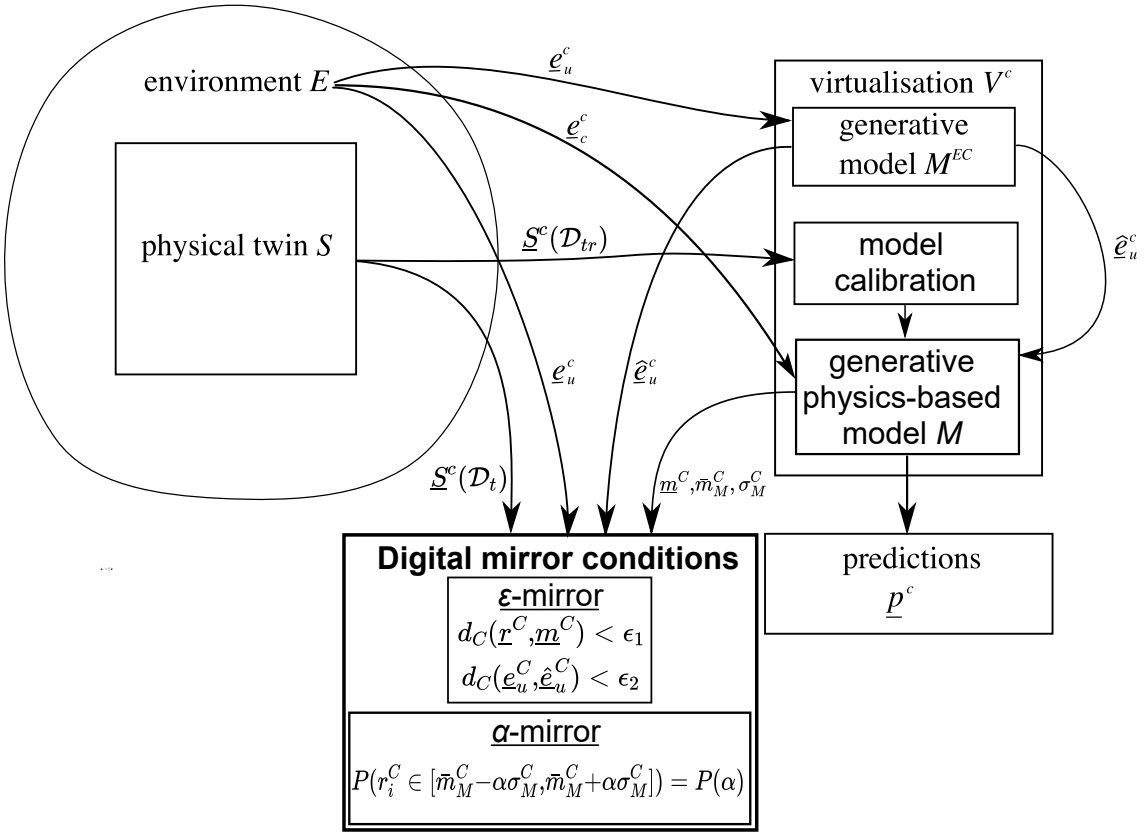


Figure 2.3: Schematic representation of the proposed framework for a digital mirror.

case, and approaches that take into account the nature of the data are more appropriate and perform better). A definition of learning is provided in [36]. A simplified version of the definition is considered here.

Let $\mathcal{X}$ and $\mathcal{Y}$ be two measurable spaces and $\mathcal{M}(\mathcal{X}, \mathcal{Y})$ be the set of all measurable functions (models) from $\mathcal{X}$ to $\mathcal{Y}$. Consider a *loss function* $\mathcal{L} : \mathcal{M}(\mathcal{X}, \mathcal{Y}) \to \mathbb{R}$ and pairs of data $s_t^i = (x^i, y^i),\ x^i \in \mathcal{X},\ y^i \in \mathcal{Y},\ i = 1, 2...n$. A *training algorithm* is a mapping $\mathcal{A}$ such that,

$$\mathcal{A} : \bigcup_{i=1}^{n} s_t^i = (x^i, y^i) \to \mathcal{M}(\mathcal{X}, \mathcal{Y}) \tag{2.6}$$

which is built in order to find a model $f_{s_t} = \mathcal{A}(s_t)$ that performs well on the *training data* s_t according to the loss function $\mathcal{L}$ and is able to perform well on unseen pairs of data s_u, such that $s_u \cap s_t = \emptyset$, according to the same loss function $\mathcal{L}$ or even some different loss function $\mathcal{L}' : \mathcal{M}(\mathcal{X}, \mathcal{Y}) \to \mathbb{R}$.

The above definition of learning is definitely not global and, strictly speaking, may not include some machine learning algorithms (for example *unsupervised learning* [37] or *reinforcement learning* [38]). In any case, it mainly describes the tasks that are performed in the current thesis and a basic modelling scheme which is, in parallel to classic schemes, followed by physics-based methods.

References to machine learning tend to be about complicated methods, and are often about algorithms that are considered *artificial intelligence* algorithms. Artificial intelligence is a wider field than machine learning. Even software based on simple *if* statements, that covers a wide range of potential inputs and performs well on some defined tasks, can be considered artificial intelligence. Software used in automated customer service is often of this type. A machine that passes the Turing test is essentially a machine whose performance is indistinguishable from a human [39], and is considered to be artificial intelligence.

Machine learning algorithms have a wide range and are sometimes far from what is considered to be artificial intelligence. A very simple but useful case of machine learning algorithms is that of *linear regression*. The specific algorithm is a very trivial type of model definition according to observations of a phenomenon. In practice, it is a simple linear function, $y = ax + b$ (in the one-dimensional case), fitted on data, which also has a closed form deterministic solution regarding the trainable parameters of the function. Although it is a very simple algorithm, it has been very successfully applied in many cases, such as in structural statics and dynamics, when one wants to define the Young's modulus of some material by compression or tension tests in a laboratory. An example of a linear regression problem, which could be considered as the most basic machine learning application, is shown in Figure 2.4. In the figure, the data are the black points and the model, which is trained according to them, provides the predictions of y for different values of x is the blue line. The training algorithm of the method explicitly minimises the sum of the distances of the predictions from the observations, i.e. the length of the red line segments [40].

When defining a model in such a data-driven manner, it is expected that errors will be present. This is clear from the error line segments in Figure 2.4, which cannot be zero in this case for any straight linear model. Different types of models could achieve even lower

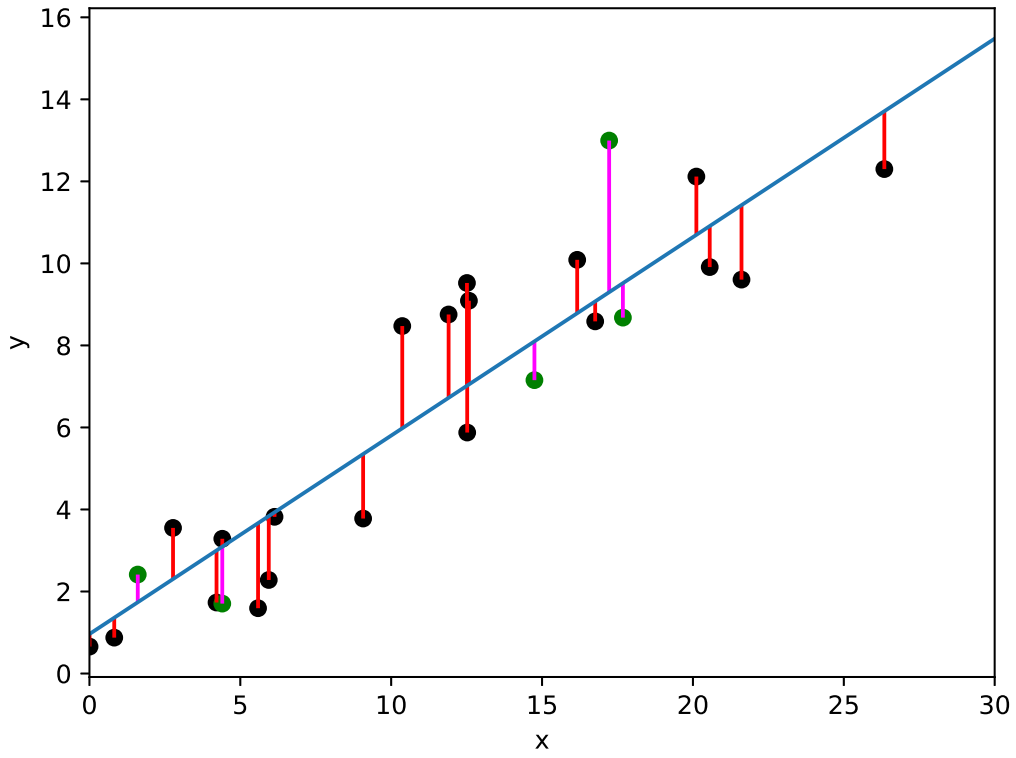


Figure 2.4: Example of linear regression model (blue line) trained according to observations (black points); testing data samples (green points), training errors (red line segments) and testing errors (magenta line segments).

errors.

An obvious model choice, different and more complicated than linear regression, would be *polynomial regression*. In a similar way to linear regression, where a linear binomial is fitted, a polynomial of maximum order n is fitted on the data. An example of such a machine learning algorithm is shown in Figure 2.5. In this figure, the observations are once again the black points and the fitted polynomial regression model is the blue line. Unseen data are shown as green points in the plot and their corresponding errors with magenta line segments. It is clear that the errors of the model for the training data and the unseen/testing data are of the same order of magnitude.

Polynomial regression offers a nice way of demonstrating the concept of *overfitting*. If one tries to reduce the error of the training data in the specific example shown in Figure 2.5, higher-order of polynomials could be considered as the model to be fitted on the data. By

increasing the order of the model, higher accuracy can be achieved (in fact if the order is one less than the number of points, the model should exactly fit on the training data). However, this may lead to situations like the one shown in Figure 2.6, where the curve predicts exactly the training data, but has quite an unstable behaviour and may have large errors for unseen data. This result is called *overfitting* and avoiding it has been a major objective of research in the machine learning community. A common way to avoid it is to use as simple models as possible (a lower-order polynomial in this case), an approach also referred to as *Occam's razor*.

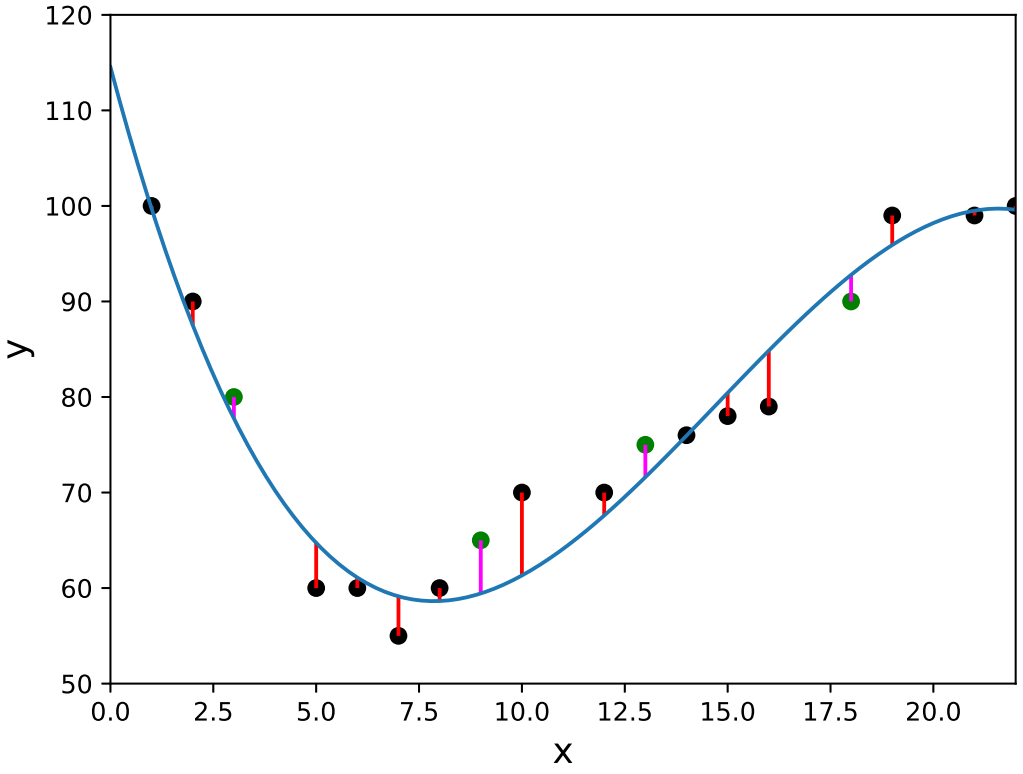


Figure 2.5: Example of a polynomial regression model (blue line) trained according to observations (black points); the errors of the training data are also shown (red line segments) as well as some unseen data (green points) and their corresponding errors (magenta line segments).

The two machine learning algorithms discussed so far belong to a wider category of machine learning algorithms called *supervised learning*. This category refers to algorithms that seek relationships, having instances of both input and output data. Some supervised machine learning models that are often used are *support vector machines* (SVM) [41], *decision*

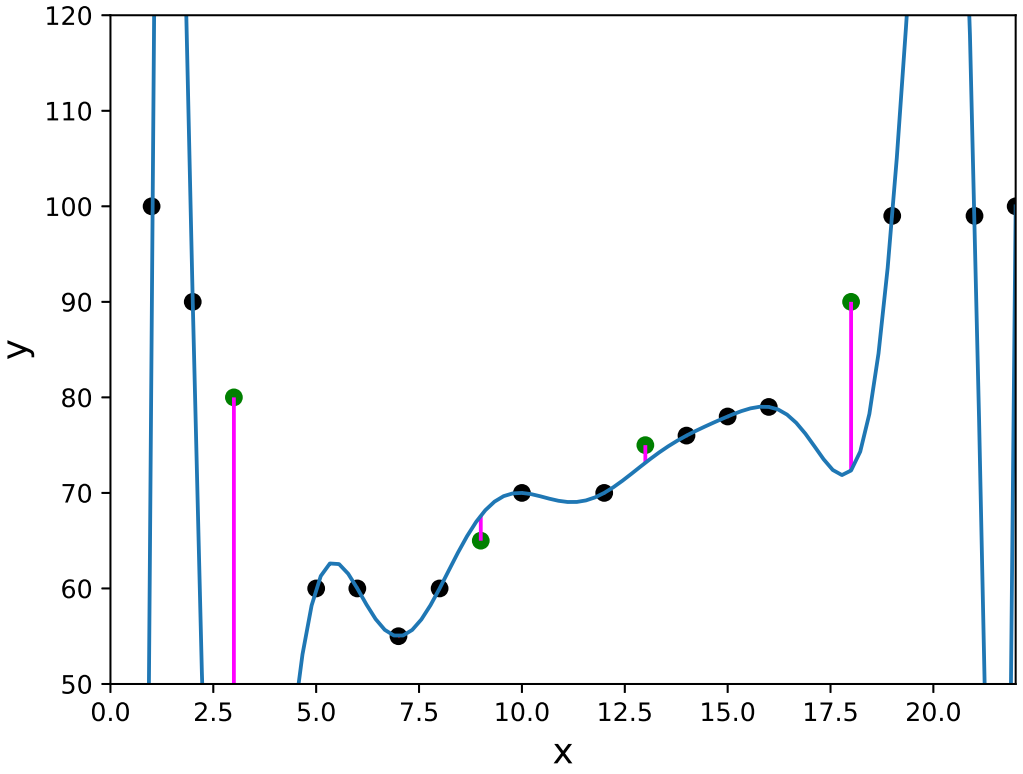


Figure 2.6: Example of polynomial regression model of higher order (blue line) trained according to observations (black points) which appears to be overfitted; the errors of the training data are also shown (red line segments) as well as some unseen data (green points) and their corresponding errors (magenta line segments).

trees [42], *Gaussian processes* [43] and, of course, *neural networks* [8, 10]. The supervised learning approach to defining models is the most relevant one for calibrating a physics-based model according to data. Models can be defined both for *regression* as well as *classification* tasks.

Supervised learning requires proper collection of data. As already mentioned, input and output data should be dictated by some causal relationship in order for the algorithm to perform proper inference. However, acquiring such data may be expensive. Since the input and the output should both be measured, more than one sensor is usually needed to define a proper dataset. Furthermore, when such an algorithm is used in order to imitate human behaviour, humans should spend time defining the inputs and the outputs of the data instances, which is a time and energy-consuming procedure. Nevertheless, supervised

models are, arguably, the most common and useful type of machine learning algorithms for structural dynamics.

A second type of learning often followed by machine learning algorithms is that of *unsupervised learning*. According to this scheme, data instances do not have predefined target outputs. Algorithms are called to perform inference (often classification) discovering patterns without any guidance from the user. Some such algorithms are *k-means clustering* [44], probability density function fitting [45], etc. The algorithms of the current category offer solutions to problems of big data, since they can automatically, or semi-automatically detect patterns within large datasets and therefore assist humans in managing them and using them.

A combination of the two aforementioned approaches is a third type of learning, that of *semi-supervised learning*. This type of learning often refers to having some labelled and some unlabelled data, and looking for a way to extract as much information as possible from both categories. Examples of such applications are presented in [46–48]. Semi-supervised learning has been mainly focussed on exploiting unlabelled data with a view to avoiding large labelling costs. In contrast to unsupervised learning, semi-supervised regression schemes are more often encountered.

Similar to semi-supervised learning, *active learning* [23] is an attempt to deal with situations where few labelled data are available for model training. Active learning is the discipline of selecting the unlabelled data samples whose labelling and inclusion in the training process would yield the best results, i.e. provide better information during model training. This training scheme is motivated by the large costs of acquiring and labelling data, and also from the lack of data needed to properly train some model. In structural dynamics -and more specifically structural health monitoring- active learning has been effectively applied to address the common issue of lack of data in the discipline [49].

Another type of learning worth mentioning is *reinforcement learning* [38]. Reinforcement learning is based on the idea of rewarding a trainee when he/she achieves good results (and maybe penalising them when they achieve the opposite). The method is convenient because an agent can be trained according to a simulated procedure and a score function which reflects how well it is performing the tasks it is needed to perform. Via the simulation

and a reward and/or penalty system, training of the agent is performed. Such training schemes are often followed when one wants to train an agent to play some game efficiently [50], because games tend to have scores of their own, whose maximisation is sought and so the reward/penalty scheme is already defined for the training procedure.

A type of learning, which is particularly interesting and useful for structural dynamics, is *dimensionality reduction*. The algorithms included in this type of learning could also be considered to be unsupervised learning algorithms. The goal of these approaches is to find a way of reducing the number of features for data samples of a dataset, losing as little information as possible. A basic dimensionality reduction algorithm is that of *principal component analysis* (PCA) [51]. That specific algorithm attempts to define new features of the data as a linear combination of the existing features, so that the correlation between the new features is minimised. A nonlinear version of the algorithm, is based on using an *autoassociative neural network* (an *autoencoder*) [52], to define a space with reduced dimensionality than the original data space, which preserves as much information as possible. For many machine learning disciplines, these algorithms are extremely useful, as they offer a way of reducing the input and output dimensionality of other machine learning algorithms and to achieve faster results and more interpretable models. For structural dynamics they are also very important, since there are parallels between them and *modal analysis*, which is one of the most powerful and important tools of the discipline of dynamics, and will be further discussed in Chapter 5. Dimensionality reduction algorithms also provide convenient ways of visualising data [53].

The types of learning that have been described so far are strategies that can be followed to fit different types of models. A specific type of model that has dominated the machine learning applications in recent years and can be trained using all of the aforementioned schemes is that of the *neural network* [8–10]. The algorithm has proven quite powerful in performing various tasks, and yields results that are close or even outperform human performance. In the current thesis, the machine learning algorithms studied are variations of neural networks and therefore, some discussion should be made about the background theory of the algorithm.

2.3.1 Neural networks

Neural networks, as their name indicates, are motivated by the functionality of neurons or nerve cells. A schematic representation of such a cell is shown in Figure 2.7. Neurons are responsible for transferring electrical signals within the brain resulting in the various functionalities of the human body. A common way that neurons function is that they are considered as gates which allow or block electrical signals according to their power. This latter functionality inspired the creation of a *single neuron* model.

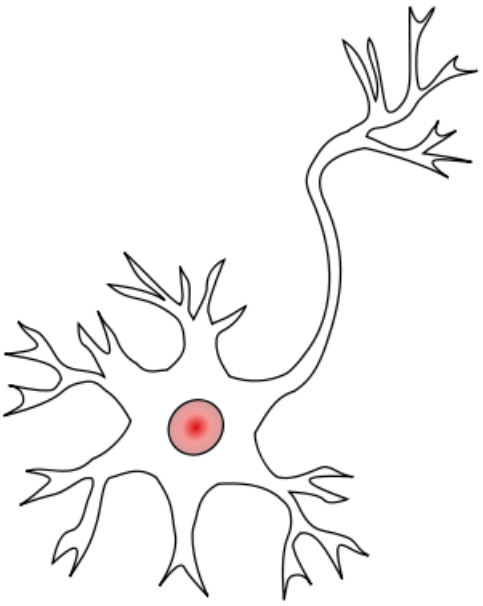


Figure 2.7: Schematic representation of a human neuron, source `https://en.wikipedia.org/wiki/Neuron`.

A single neuron function can be used in order to classify samples according to some threshold value t. The function would then be the Heaviside step function given by,

$$H(x) = \begin{cases} 0 & x \leq t \\ 1 & x > t \end{cases} \tag{2.7}$$

This function imitates the reaction of a neuron to an incoming electrical signal. The signal is allowed to pass and be transmitted to the next neuron if its power is greater than some threshold value. Equally, the function classifies inputs of $x \leq t$ to the class 0 and for every

other x to the class 1. Such models are quite simple; however, decision trees essentially use a collection of such functions in order to perform inference. Such a function is called an *activation function* of the neuron, because it defines whether the neuron is activated and lets the signal pass through it, or if it blocks the signal.

Although the Heaviside step function is convenient when a threshold is needed, it is not a smooth and differentiable function. The need for the activation function to be differentiable will become more clear when the training of the neural networks will be described. In order to use a function that is differentiable, and at the same time, offer a step-like behaviour like the Heaviside function, sigmoid functions are commonly used as activation functions of neurons. Two of the most common functions used are the *logistic function* (or just sigmoid function) $f(x) = \frac{1}{1+e^{-x}}$ and the *hyperbolic tangent* function $f(x) = \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. The aforementioned functions do not have any trainable parameters in the form shown. Introducing a coefficient that multiplies x before passing it through the activation function, the 'slope' of the sigmoid varies. The hyperbolic tangent function $\tanh cx$ is shown in Figure 2.8. It is clear that for higher values of c, the function becomes more steep and gets closer to the Heaviside step function.

For classification, a quite steep sigmoid function is an appropriate function to use since it is close to the Heaviside step function. For regression, neural networks are, in fact, proven to be global approximators [54], i.e. they are able to approximate any function with arbitrary accuracy. An intuitive way to illustrate how they achieve such results is now described. A collection of noisy observations of some underlying function are shown in Figure 2.9. A single sigmoid function cannot be used in order to approximate such a function as the one shown with the blue line. However, if the range of input values is divided in segments, as done by the vertical red lines, the separate curve segments of the underlying function can be approximated with better accuracy by a sigmoid function. Now, the ability of the sigmoid functions to act as a threshold, becomes even more valuable. The neural network can be considered to be a collection of neurons, each one fitted on a different curve segment and deactivated for every other interval of the input space. Together with some added quantity b to the sigmoid functions, which is called a *bias* and is also trainable similar to the coefficient c, the neural network, considered as a collection of neurons, can be used to approximate such curves. If the intervals are considered smaller and therefore,

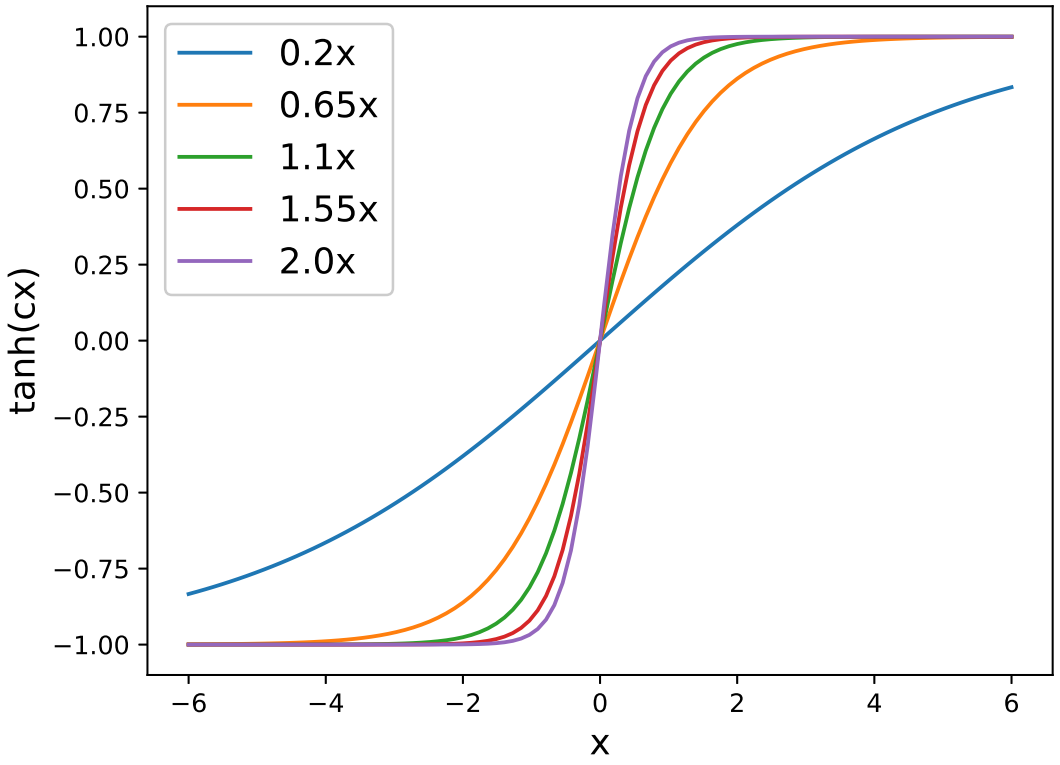


Figure 2.8: Hyperbolic tangent function $\tanh cx$ for different values of the parameter c.

more neurons are included in the neural network, the approximation capabilities increase even further, since the curve segments can be closer to straight lines, which can easily be approximated; as shown in Figure 2.8, for higher values of the coefficient c, the sigmoid function is closer to a straight line than a step function.

Neural networks can of course, approximate functions and relationships that include more than one input and more than one output; their layout is often defined as in Figure 2.10. This specific type of neural network is called a *feed-forward* neural network, since the information flows exclusively from the input nodes to the output nodes. The input nodes are called the input *layer* of the network and the output nodes the output layer. The collection of nodes in the middle are the *hidden layers*. In Figure 2.10, only one hidden layer is used. Such a neural network is considered to be a *shallow* neural network, while neural networks with more than one hidden layers are considered *deep neural networks* and are the main object of *deep learning*. Deep learning has interesting properties compared

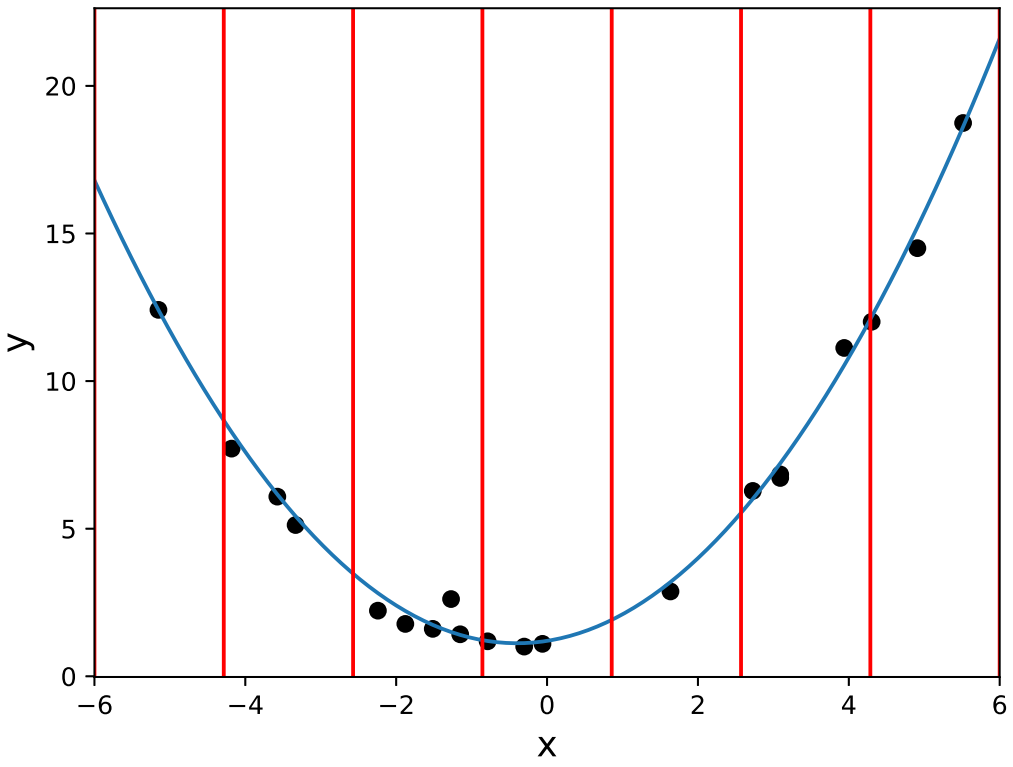


Figure 2.9: Noisy samples (black points) taken from some underlying function (blue line).

to using shallow networks, but since in the current application, mainly shallow networks are used, the interested reader is encouraged to refer to [10, 36] for further details.

A line connecting two nodes in Figure 2.10 is called a *connection* or a *synapse*. Every synapse has a weight parameter value assigned to it and it defines how the values of variables are transferred between nodes. The values of the variables of the input nodes are multiplied by the corresponding weight value of the synapse and transferred to the nodes of the hidden layer. Afterwards, every variable, multiplied by the corresponding synapse weight, that ends to a node of the hidden layer, is summed and passed through the activation function of the corresponding node. The same applies to define the values of the variables in the output nodes of the network. The output $\mathbf{z}^{(k)}$ of the k^{th} neural network layer is defined by,

$$\mathbf{z}^{(k)}(\mathbf{z}^{(k-1)}) = f(W^{(k)}\mathbf{z}^{(k-1)} + \mathbf{b}^{(k)}) \tag{2.8}$$

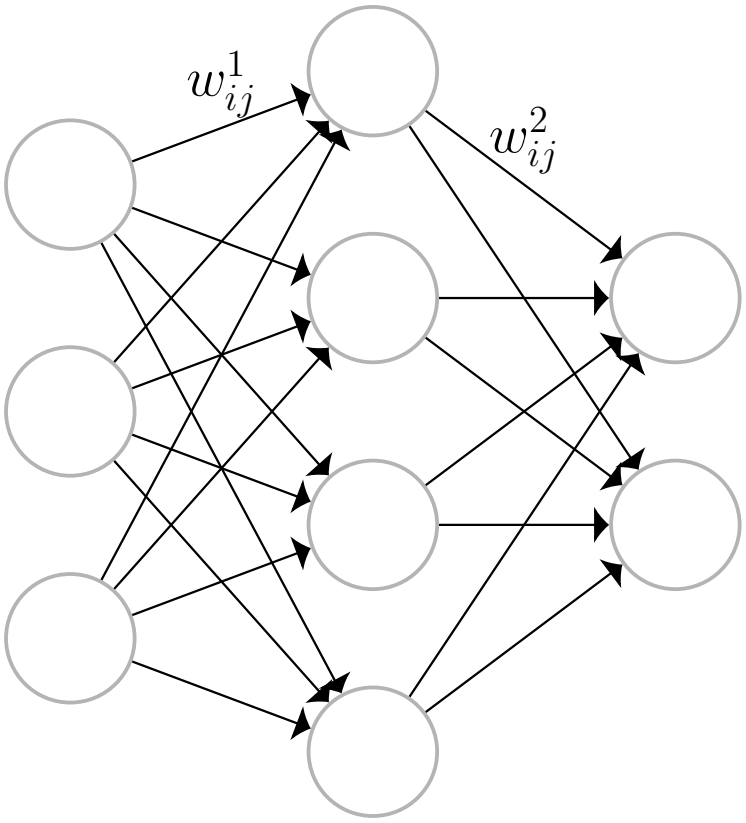


Figure 2.10: Example of a neural network layout.

where $W^{(k)}$ is the *weight matrix* of the k^{th} layer, whose elements w_{ij} are the weights of the synapses that connect the i^{th} synapse of the previous layer with the jth synapse of the current layer, $\mathbf{z}^{(k-1)}$ is the output of the previous layer, $\mathbf{b}^{(k+1)}$ is the bias vector of the k^{th} layer and f is the activation function of the nodes applied element-wise on the output vector. The above equation describes the calculations needed to calculate the output of the k^{th} layer, given the output of the $(k-1)^{th}$ layer. The calculation of the whole l-layered neural network output, also called a *forward propagation* is given by,

$$\mathbf{z}^{(l)} = \mathbf{z}^{(l-1)}(\mathbf{z}^{(l-2)}(...(\mathbf{z}^{(1)}(\mathbf{x})))) \tag{2.9}$$

Each hidden layer has different number of nodes, which is called the size of the hidden layer. Using more nodes, the predictive capabilities of the neural network are increased, as well as its trainable parameters. The weights of the synapses and the biases are the parameters that need training. The neural network is essentially an overparametrised function, organised in subsequent vector transformations via nonlinear functions, with

trainable parameters that are trained according to some observations. Training of neural networks is performed using *backpropagation*

Backpropagation is an optimisation algorithm, similar to gradient descent or steepest descent [55]. The algorithm is used to minimise (or maximise if convenient) the value of the loss function $\mathcal{L}$ of the neural network. The loss function most commonly used for regression is the mean squared error, given by,

$$\mathcal{L} = \frac{1}{n} \left\| \sum (\hat{\mathbf{y}}^i(\mathbf{x}^i) - \mathbf{y}^i(\mathbf{x}^i)) \right\|^2 \tag{2.10}$$

where $\mathbf{x}^i$ is the ith input vector, $\hat{\mathbf{y}}^i(\mathbf{x}^i)$ is the prediction of the network for the ith input and $\mathbf{y}^i(\mathbf{x}^i)$ is the target value of the prediction.

The procedure of backpropagation is performed in steps. At every step, the gradient of the loss function for every trainable parameter is calculated and then the value of each trainable parameter is updated according to,

$$\theta_i' = \theta_i - \alpha \frac{\partial \mathcal{L}}{\partial \theta_i} \tag{2.11}$$

where θ_i is the current value of the trainable parameter that is being updated, θ_i' is the updated value and α is the *learning rate*, which controls how fast or slow the value of the trainable parameter changes according to the derivative of the loss function. The above updating formula is applied iteratively in many steps, which are called *training epochs*. After every update, the output of the neural network gets closer to the target values of the dataset.

Equation (2.10) is used in regression tasks. For classification tasks, the corresponding loss function is given by,

$$\mathcal{L} = \sum_{c=1}^{n_c} y^i(\mathbf{x}^i) \log \hat{y}^i(\mathbf{x}^i) \tag{2.12}$$

where n_c is the number of classes of the classification task, $\hat{y}^i(\mathbf{x}^i)$ is the output of the neural network for the i^{th}input-output sample and $y^i(\mathbf{x}^i)$ is the output value of the i^{th} sample of the dataset. Using the above loss function, the network classifies each sample to the i^{th} class with the higher output y_i [8, 9].

The most general framework for training a neural network, is by defining a dataset with input and corresponding output samples and following the backpropagation procedure to fit the neural network to the data. It has already become clear that there are parameters, such as the number of layers and the number of neurons in each hidden layer, which are not calculated by the training procedure. These parameters are called the *hyperparameters* and the user must decide their values before training. A common practice to define the best value of the hyperparameters is to follow a *cross-validation* training scheme to fine tune them. Such a scheme involves splitting the available dataset into three subsets - the training, validation and testing datasets. Subsequently, the training dataset is used to perform backpropagation for various values of the hyperparameters. Every network is evaluated on the validation dataset and the model that yields the best results in the validation dataset is also tested on the testing dataset.

The neural network architecture can also be considered a hyperparameter. The neural network shown in Figure 2.10, and similar networks with more hidden layers, are feedforward neural networks, since information flows only from layer $l-1$ to layer l. Moreover, the way that this information flows, following equation (2.9) is via densely-connected layers; this is the most general case of neural networks. More types of architectures exist and serve different functionalities according to the users needs.

A major example of a different architecture is the *convolutional neural network* [20]. Their main usage is in image processing. Such neural networks allow the input to the network to be a whole image, which is processed according to filters that take into account the locality of the pixels. A different and crude approach would be to consider every pixel an input node. However, by considering trainable filters that perform convolutions locally on the image to extract its features, the algorithm becomes more versatile and exploits, to some extent, the "physics" of the image. At the same time inference is made in a more natural way, since it follows a similar procedure that the human eye follows.

A second type of neural network architecture, is the *recurrent neural network* [56]. This type is often used for time-series, since it is able to define relationships and temporal characteristics of data. Recurrent neural networks allow exchange of information between different hidden layers and are able to model time-series of different lengths. They are also used in natural-language processing, since texts are sequences of words connected via

grammar and semantics.

In every application, the type and architecture of neural network used should be defined properly. As described in [36], considering a learning task, a hypothesis space $\mathcal{F}$ is defined where the potential solutions to the learning problem live, also a search space $\mathcal{F}_\delta \subseteq \mathcal{F}$ where one shall search for the approximation function, a hypothesis $f \in \mathcal{F}_\delta$ is considered, and the error of the learning task $\mathcal{R}(f)$ is bounded according to,

$$\mathcal{R}(f) \leq \inf_{f \in \mathcal{F}} \mathcal{R}(f) + \epsilon_{opt} + \epsilon_{stat} + \epsilon_{appr} \tag{2.13}$$

where ϵ_{opt} is the optimisation error, ϵ_{stat} is the statistical error and ϵ_{appr} is the approximation error. The term $\inf_{f \in \mathcal{F}} \mathcal{R}(f)$ defines the approximation capabilities of the hypothesis space that has been selected; for neural networks -which are proven to be universal approximators- this should be equal to zero. The optimisation error refers to the algorithm used to search for the proper hypothesis. The approximation error is defined by the best possible solution $f^* \in \mathcal{F}_\delta$, and how well it explains the relationship to be learnt. Finally, the statistical error relates to the error induced by picking a function from the hypothesis set according to the available data (since values of the function for every point in the desired space are in general not available). Proper selection of models essentially means proper selection of the search space $\mathcal{F}_\delta$.

In [57], the idea of symmetry for the hypothesis space is introduced. By creating machine learning algorithms that respect symmetries in the data, the search space is largely reduced. If the assumption that the underlying relationship respects the aforementioned symmetries is correct, the statistical error is reduced, and at the same time, the approximation error is not increased; because the real underlying relationship of the data indeed respects the induced symmetries. This whole idea coincides with the idea of *inductive biases* [58], and is a smart and efficient way of driving the learning procedure faster and more efficiently to the proper solutions. In [57], the symmetry of existing neural network architectures is studied (such as convolutional neural networks and graph neural networks). Symmetry properties may potentially explain why such models are able to perform so well, even if they have a very high number of trainable parameters.

Many more types of neural networks exist and, according to the application, they have

certain advantages and disadvantages. Specific types of networks, such as the autoencoders and the graph neural networks are particularly useful in structural health monitoring. A new type of neural networks, emerging in recent years, is that of *generative adversarial networks* (GANs). The novelty of the algorithm lies mainly in the training procedure, rather than in the architecture of the networks involved.

2.3.2 Generative adversarial networks

A novel approach to training neural networks is via *adversarial training* [12]. Traditional schemes of training include using data in a supervised manner, to force the neural network to learn the underlying relationships between the inputs and the outputs. According to adversarial training, tuning of the parameters is achieved via a *competition* of *two* neural networks, which have clashing objectives and, as training progresses, they become more efficient at their tasks.

The two neural networks which are used in order to train a GAN are the *generator* and the *discriminator*. Their original purpose was to create images that resemble reality. The objective of the generator G is to create samples that look real according to some dataset. The discriminator D, on the other hand, has the goal of distinguishing between artificial samples and real samples, which are picked from the available dataset. The network which at the end performs the desired task, is the generator, but both D and G play an important role in achieving the task.

Training is performed in two stages. During the first stage, samples $\mathbf{x}$ are sampled from the available dataset and are fed into the discriminator network in order to define the probability of them being real; i.e. $P_{\boldsymbol{x}\sim p_{data}} = D(\boldsymbol{x})$. The discriminator therefore is a binary classification network, which is trained to predict zero if a sample is fake and unity if the sample is real. Ttraining is performed in a supervised manner, by having zero as target value, when artificial samples are used as inputs, and unity otherwise. During the second stage of training, the two networks are connected together as shown in Figure 2.11. This time the discriminator is not trained; its trainable parameters are considered constant. Random noise vectors $\mathbf{z}$ are sampled from a pre-defined probability distribution $p_z(\boldsymbol{z})$, and the generator transforms them into samples, which are subsequently passed

through the discriminator. This forward pass transforms the latent noise vector $\boldsymbol{z}$ into samples of the real space and, at the same time, the discriminator is performing inference about whether the sample is real or not. If trained with a target value of one, i.e. a "real" label, the trainable parameters of the generator will gradually learn to fool the discriminator into believing that its samples are real.

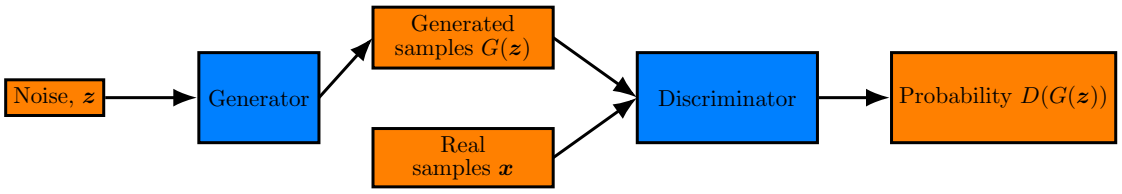


Figure 2.11: Layout of a generative adversarial network.

The optimisation problem for the competition between the two neural networks is given by,

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.14}$$

where $\mathbb{E}[\ ]$ is the mean value of the quantity in the brackets. The optimisation procedure should reach a *Nash equilibrium point* [59]. Neither of the two networks can be dominant in the competition, because this will lead to instabilities and phenomena such as mode collapse, which means that the generator generates a single sample for every value of the input noise vector. The performance of both networks should gradually and slowly increase, so that both can reach a certain point, at which, in terms of the generator, samples/images, which look real to the human eye, are generated.

In the original work [12], the goal of training a GAN was simply to generate artificial data that look real. The experiments were initially performed on the MNIST dataset -a dataset of images of handwritten digits between zero and nine. The algorithm was able to efficiently generate images of handwritten digits that looked real and were not possible to be distinguished by humans (an example of generated samples next to real ones is shown in Figure 2.12), whether a sample was picked from the real dataset or it was artificial. The results are impressive and, in later works, the algorithm and its variations are applied to figures of human faces [60], animated faces [61], three-dimensional object generation [62], and generic high-fidelity images [63], with equally and even more impressive results.

Figure 2.12: Artificially generated samples of handwritten digits [12].

The power of GANs to generate such realistic figures has even triggered discussions about cyber-security, because of the potential creation of artificial human faces and images in general.

Apart from the impressive results in generating images that look real, GANs are able to learn underlying distributions and manifolds of data. What the discriminator is doing, is defining a boundary around the manifold of real data in the multidimensional space to which they belong and indicating samples within the boundary as real, and outside it as fake. At the same time, because the real samples belong to some probability distribution $p(\mathbf{x})$, it also learns the probability distribution of the data. In fact, in the original work [12], it is proven that the equilibrium point of training is achieved if the generator generates samples from the same distribution as the real data. Of course, this functionality is not new to data-driven modelling. Defining a probability distribution of data is a common practice in modelling data; for example when novelty detection is desired. Common approaches to modelling the probability distribution of data include simply defining a Gaussian distribution with mean and standard deviation calculated from the data. This

approach obviously lacks versatility, since Gaussian distributions refer to unimodal distributions of data. Another approach would be to fit a kernel density function [64], which can explain distributions with many modes, but is computationally inefficient in high dimensions.

In [65], an attempt is made to define a variation of the GAN algorithm, which is able to disentangle underlying features of the available dataset. The algorithm is able to generate samples conditioned on categorical and continuous variables. The generated samples belong to a predefined class, in the case of the categorical variables, or have specific intensity of some feature, in the case of continuous variables. The important aspect of the algorithm is that the procedure of defining the underlying variables is done in an almost unsupervised manner; the only prior knowledge needed is the number of continuous and underlying variables. Given the number of the underlying variables, the algorithm searches for equal numbers of features or classes of the data, which are statistically independent. An application of the specific algorithm on a dynamics problem is presented in [66]. The *infoGAN* model is applied on a structural health monitoring dataset and is able to generate samples according to four different damage cases of a simulated structure. The model also proves efficient in encapsulating damage intensity via its continuous variable and is able to generate data with different damage levels by varying the specific input variable.

GANs prove quite efficient for high-dimensional data, even when they are combined with convolutional layers in order to extract features from images. A rather interesting characteristic of the algorithm, is that it maps noise into the real space, an operation usually performed in the reverse direction from decomposition algorithms (data are often mapped to some latent space [52]). This operation is interesting, since the algorithm, in an unsupervised manner, looks for a way to map latent variables to features of the data. A properly-trained GAN can be exploited in order to manipulate such features of the data, by varying the latent noise vector, to generate new data. Such examples are presented in [67, 68], where photos of human faces are being processed in the aforementioned way. Another use of GANs is for translating images and paintings into different styles, from realistic to famous painters' styles etc. The specific variation of the GAN algorithm is called a *cycle-consistent generative adversarial network* [69] and an application of it in nonlinear modal analysis is presented here in Chapter 5.

Other useful applications of GANs include filling gaps in images, which for some reason are damaged [70], and predicting video sequences [71]. Moreover, attempts have been made to learn underlying characteristics of the data in [65]. For structural dynamics, a straightforward use of GANs is to generate artificial data in order to better train some data-driven model. GANs can be used to generate as many samples as an algorithm may need to perform satisfactorily in some task. Lack of data is a common problem in structural dynamics, and GANs may be a way to deal with it. The rule of thumb for neural networks is that they need about ten data samples for every trainable parameter [72]. Moreover, GANs can be used to make use of unlabelled data [48, 73]. In a classification problem, a dual training scheme can be followed. The first is the adversarial training, which forces the generator to generate data within the manifold of the real data, and the second is a classic classification training with target values only for the labelled data. In this way, the generator is also a predictor of the classes of the data, is trained on both labelled and unlabelled data and is expected to perform better than being trained only on the labelled samples, since information from the unlabelled samples is also exploited.

In the current work, variations of GANs are used for different modelling approaches to structures. In Chapter 5, GANs are used to perform nonlinear modal analysis. An attempt to generate data for unseen environmental conditions and a generative scheme for defining a digital twin of a structure are presented in Chapter 7. Finally, in Chapter 8, an application of GANs for time series is presented, with a view to performing damage prognosis and remaining useful life estimation for structures with evolving damage. From such applications, it becomes clear that generative adversarial networks are a versatile algorithm, based on a new way of training such models, which may unlock new potential uses of data-driven algorithms. They have many applications in various disciplines and are definitely going to be objects of research and applications for many years to come.

Chapter 3

Structural health monitoring

3.1 Structural health monitoring (SHM)

Structures are an essential part of modern life. Almost every human activity includes using some structure. From living within a house and driving every day from home to work, to using special equipment, such as medical equipment, structures are a fundamental part of human society and civilisation. Just like humans and other living organisms, structures live within a changing environment and are subject to alterations, ageing and accidents. The length of their life is not infinite, and sometimes it is not even the length intended by the designers. Unexpected events can happen and cause damage to structures that will reduce their life.

A major way of avoiding such damage is by designing structures that can withstand the worst-case scenarios of external loading; this has been the practice quite often with civil structures. Engineers have to define the loading cases with the worst effects on the structure and design it to cope with most of these events. This approach naturally works, and has offered high levels of safety to humans for a long time, but there are definitely some drawbacks.

First and foremost, designing in such a way can lead to wasting resources in materials, space, human working hours etc. Structures may be able to withstand the worst-case

scenarios, even with weaker members or with proper evaluation and repair techniques throughout their lifetime. However, such design strategies may not allow engineers to try new techniques in construction, which could lead to new ways of creating structures and maybe even safer than before. A characteristic example of the above is *rocking seismic isolation* of buildings [74], which is often not allowed in foundation design codes, but could allow more slender buildings, saving space and economic resources from large and wasteful foundations.

The second drawback is that, in order to perform such a worst-case-scenario analysis, one has to have sufficient knowledge about the physics of the structure and its surrounding environment. Having knowledge about the materials and the structures in general, may have been the case for structures made of concrete (although even in this case the uncertainties have been huge), but for new materials -for example, composite materials- such knowledge may not be available. Moreover, from the stage of design to the stage of construction, errors and defects will always be present increasing even further the uncertainty about the design of the structure. This is aleatory uncertainty and does not allow proper modelling of structures to define the unfavorable scenarios and design in order for the structure to survive. If a probabilistic framework is used and the uncertainty is high, then design might lead to over-dimensionalising the structure, which is costly and also has a high-energy footprint on the planet.

A third problem is that, even if proper design is achieved, unwanted situations and accidents still happen. There is no way to guarantee that the structure will be subjected only to the loads which have been taken into account during the design phase. The environment around the structure constantly changes, and an unexpected situation might cause problems that might cost human lives. Modelling the environment of the structure is even more uncertain than the structure itself, because most of the environmental factors are completely out of the control of humans. Materials are also subjected to degradation and fatigue because of the environment, procedures which are largely stochastic phenomena and might lead to sudden collapse of the structure. Such an example, probably in combination with engineers not completely understanding the physics of the structure, is the Tacoma Narrow bridge (Figure 3.1), which arguably collapsed because of wind vortexes which caused the bridge to resonate and, eventually, collapse.

Figure 3.1: Tacoma Narrows bridge after having collapsed; source `https://www.britannica.com/technology/bridge-engineering/Tacoma-Narrows`.

Although such problems exist, structures still have to be created, exploited and have to be safe. Thousands of people every day use airplanes or cars, drive over bridges and use equipment for their work. Therefore, structural condition has to be maintained to such a level that they are safe for people to use them. Because of this need and the problems that arise in the design process, the field of *structural health monitoring* (SHM) has emerged [11].

SHM is based on detecting damage in structures, but has evolved into an engineering discipline with many more objectives than simply this. A first step to discussing what SHM is, what damage is should be defined. In [11], damage is defined as intentional or unintentional changes in a system, which adversely affect the current or future performance of the system. As will be discussed later, 'adversely' is a key-word of the definition. Systems change behaviour, but what is important for SHM, is to detect changes that might cause problems. In general, the various methodologies and procedures that are included in the SHM discipline can be categorised according to Rytter [75], and extended in [11], in the hierarchical structure:

1. Is there damage (*existence*)?

2. Where is the damage in the system (*location*)?
3. What kind of damage is present (*type/classification*)?
4. How severe is the damage (*extent/severity*)?
5. How much useful (safe) life remains (*prognosis*)?

As one might expect, SHM is indissolubly connected to data. In order to perform any of the tasks of SHM, the most important part is to be able to acquire data from the structure which needs monitoring. The next step is to process the data in order to extract informative features. For centuries, engineers would try to understand whether a structure is damaged. The main difference between these procedures and SHM, is that the former has been performed for many years via maintenance and offline inspections. SHM is an attempt to perform the whole task online and by minimising the human intervention needed. Similarly to how the human neural system works, a structure with an integrated SHM system would ideally be constantly monitored and only if something is wrong would a maintenance or repair procedure be initiated. Consequently, the final step is to create models based on the data. These models can be physics-based and calibrated according to data, but it is arguably more convenient to use data-driven methods in order to perform inference about the damage state of a structure. Structures have been monitored for many years by engineers.

The above hierarchy, apart from indicating five different types of tasks that one might encounter in an SHM campaign, also gives a context about the complexity of the tasks. The first step, that of detecting the existence of damage, can be considered the most basic. Models, which perform such detection, are often based on the idea of novelty detection [76, 77]. In order to perform novelty detection, a baseline healthy condition of the structure has to be defined and data need to be acquired from it. Afterwards, the procedure is as simple as trying to identify whether new data, which are acquired from the structure during monitoring, are similar to the data of the healthy condition. Methods that perform this objective might be as simple conceptually as defining a probability density function according to the healthy data, and calculating the probability density of the data during the monitoring period. More complicated methods include decomposition algorithms that describe the manifold of the healthy data using autoassociative neural networks [78].

Figure 3.2: Photo of the Z24 bridge.

In most cases, novelty detection is performed in a completely data-based manner. What is important for such methods is the availability of data that represent the healthy state of some structure and a way or a metric to define how different newly-recorded data are. A major problem with such approaches, is that the structure may be behaving differently because of benign changes. Such behaviour is often detected because of nonlinearity or changes in the loads or the environmental conditions. False alarms caused by benign changes can initiate maintenance procedures and lead to unnecessary costs - false alarms. A characteristic example of such behaviour in the discipline of SHM is the Z24 bridge [79, 80]. In the case of the Z24 bridge (Figure 3.2), which is considered a benchmark for many methods, environmental conditions affected the acquired data from the structure more than damage did. To avoid such situations, it might be useful to use physical knowledge about the behaviour of the structure to prevent unnecessary procedures. Again though, because of epistemic uncertainties and deviations between the construction and the design of the structure, most novelty-detection tasks are performed in a data-driven manner.

The second level of the hierarchy is about localisation of damage. In some cases, it is very important to know which part of the structure has been damaged. Specific members of a structure might be of higher importance regarding safety, and being able to localise damage to them could be crucial to avoide collapse. Moreover, localisation reduces maintenance

and repair costs even more, since by knowing before starting the procedures, where the damage is, targeted operations are performed and the procedure costs less and is faster. As far as the methods used for this task, again data-driven methods are dominant. Using local characteristics, such as *transmissibilities*, from a structure, can help identify the area in which damage is present. Understanding the physics of the structure may assist, and provide some prior knowledge about where damage might be present.

Completely data-driven ways of localising damage are also available and quite efficient. An approach that has been successfully followed to localise damage is by training a classifier model with data of a structure damaged in different locations [81]. One structure that has been studied is a Gnat aircraft wing with several sensors placed on it (Figure 3.3). The approach, is completely data-driven and requires no previous knowledge about how damage in different locations of some structure will affect the quantities that are being monitored. An interesting fact from the work presented in [81], is that, although local features (transmissibilities between sensors shown in Figure 3.3) are used to locate the damage, for localisation to be performed efficiently, the combined features have to be taken into account. This may largely be because of the nonlinearity of the materials of the structure and how damage affects the data that are monitored. Such a failure also indicates that data-driven methods might need more information than the amount that is available, and in some cases sensor placement might cost a lot or might even be impossible to be placed at some points of the structure.

The next step of Rytter's hierarchy is about defining the type of damage that has been detected. Different types of damage require different treatment. Some structural member might have been damaged because of failure of the material, corrosion, displacement of a component and many other reasons. Each type of damage affects the performance of the structure, and the required action to repair it, in a different way. In some cases, a simple maintenance procedure might be required, or it might be that the member needs complete replacement, which might be costly in terms of time and resources. SHM methods must be able to inform the people responsible for the structure, about the reason and the result of the damage that the structure has been subjected to.

The data-driven approach to deal with such problems is quite similar to the one followed for localising damage. Different types of damage affect the structure in different ways and, if it

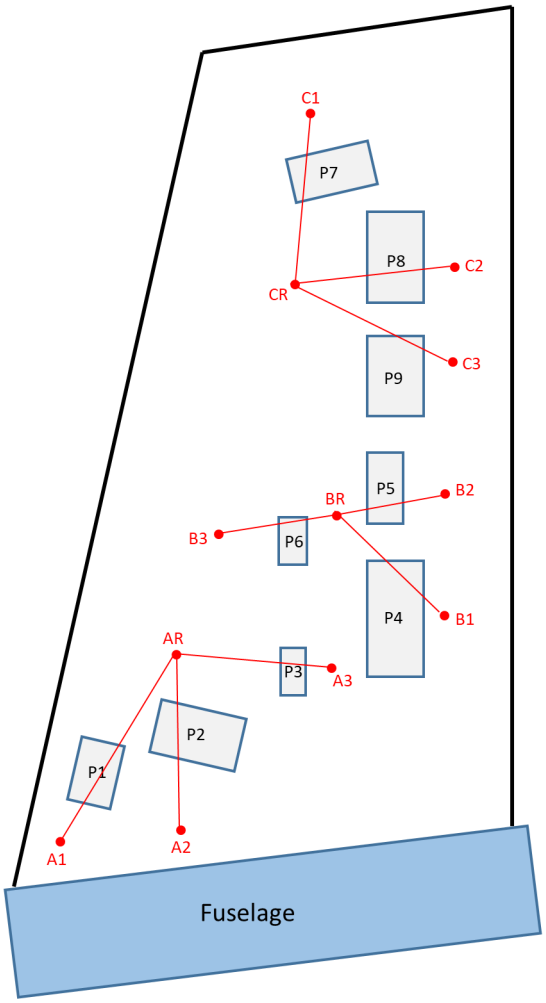


Figure 3.3: Configuration of sensors placed on the GNAT wing [81].

is properly monitored, they also affect the data features that are acquired differently. Using data, classification algorithms about different types of damage can be trained and provide quite accurate predictions about the type of damage that has infected the structure. Again, proper monitoring of the structure is essential for this procedure, since if some part of the structure (or some quantity that might crucial for some type of damage), is not being monitored, some classes of damage may be undetectable by data-driven techniques. Again physics knowledge might increase one's confidence about the type of damage, if one's knowledge about the damage mechanism is sufficient.

The final two stages of Rytter's hierarchy are the most difficult tasks to address. Step Four -defining the damage extent- requires knowledge about the mechanism and the evolution of

specific types of damage. Different types of damage have different mechanical or chemical rules of evolution through time and, in order to be able to define the current status of damage on a structure, detailed knowledge about these mechanisms is required. Data-driven approaches can be followed, but they would have many disadvantages. As in every case, in order to build data-driven models, data are required from different stages of the damage evolution. For a single structure, this may not be possible since, once damage is located, it usually gets treated immediately. Moreover, once repaired, the structure might behave differently in terms of the data that are recorded, and exactly the same behaviour is not expected to be observed [82]. Attempts to follow data-driven methods in this case are considered within the population-based SHM domain, which will be discussed later.

The most difficult stage of the hierarchy is that of *damage prognosis*. Damage prognosis is naturally the next step after defining the current level of damage of a structure. It is arguably the most difficult task of SHM and arguably the most uncertain procedure. Damage evolution and the *remaining useful life* of a structure are the result of many different factors, including environmental conditions, loading conditions, exact material properties etc. Some models that sufficiently predict the evolution of specific damage cases exist; one example is the Paris-Erdogan law for the propagation of cracks [83]. That specific model is based on observations and empirical definition of a formula that describes the propagation. Damage mechanisms are often quite complicated and cannot be trivially modelled in a similar manner.

For many SHM applications, the use of physics-based methods is not rare [84]. A common example is in the monitoring of bearings, which may develop cracks on their outer or inner ring. Such cracks have quite specific effects on the monitored quantities of the bearing. These effects have been studied and formulae which define the expected observations exist, based on the understanding of engineers of the phenomenon. As a result, locating cracks in such members is a standardised procedure and can be globally applied to almost any bearing in almost any structure. This is a quite important example, which emphasises the importance of including physics in the procedure of monitoring. The use of physical understanding makes models robust, and in some cases, universally applicable to specific types of structures.

In order to use a data-driven approach, data should be available for various stages of the

damage evolution, which, as discussed previously, does not happen very often. Autoregressive models [85], can be used to predict future states of the structure according to recorded states. However, there is no guarantee that damage will continue evolving in the same way until failure, and that such models will be efficient in predicting future states. Population-based structural health monitoring may be able to solve such problems. Such an application of prediction of potential future damage states of a structure according to data coming from a population of structures is presented in Chapter 8. Transferring knowledge from similar structures, which have had damage evolving in them, may be the only way of building trustworthy models of damage evolution for some structure which has never been damaged before, and some specific types of damage which have been observed and studied in similar structures. Otherwise, the task seems as if one is trying to predict largely stochastic quantities, without any knowledge about them, which is intractable.

3.2 Population-based structural health monitoring (PBSHM)

It is clear that, as acquiring data is fundamental for SHM, data-driven methods are a straightforward way of performing the various tasks described in the previous section. However a problem, that quite often appears to discourage the use of data-driven methods or SHM techniques in general, is a lack of data. In order to skip the step of understanding and conceptualising the problem into a mathematical model, sufficient data acquired from a phenomenon must be available. Data-driven methods are methods that perform well when interpolating, but when one tries to predict the outcome of some phenomenon without observing any similar phenomenon before, such models might fail. However, the approaches remain powerful and their application is desired by many.

Motivated by how medicine is practiced on humans (and animals), a novel branch of SHM, *population-based structural health monitoring* (PBSHM), has emerged. Medicine is largely based on statistical facts from large populations of humans. A major example is how the "normal" condition of humans is defined. *Homeostasis* is the function that maintains some elements of the human organisms in certain levels, such as the temperature and blood pressure. Variations exist, but the healthy intervals are overwhelmingly similar for healthy humans. Based on this observation, novelty detection on humans is often

performed by monitoring such values. It would have been impossible to try and define the healthy values for every human separately and by defining some intervals in which every human's features should be, has proven quite successful and has triggered appropriate investigations to people whose values did not fall into these intervals.

For medical diagnoses, approaches based on data are also preferred. Certain diseases affect different health indices in humans and different parts of the body. Studies are performed in order to define such relationships, in order to facilitate diagnosis of certain health problems. Many of these relationships may have been defined in a physiology-based way of thinking, but others are defined solely on data from patients and observations of medical tests. Oftentimes, the observation of such phenomena leads to investigation and explanation of the relationships between problems and how they affect blood test etc. It is certain that, if one is able to understand why some health problems cause specific indications in medical tests, the relationship between the two is more robust for diagnostic purposes, thus doctors can trust such approaches more. Similarly, diagnosis of problems and damage in structures can be both physics-based and data-driven. The data-driven approach is more simple and easily applied, but the physics-based one is more robust, since it is based on the mechanism of the damage.

Finally, the more macabre field of estimating remaining life of patients, is almost exclusively based on data from other deceased patients with similar issues. The mechanisms that cause death of patients with terminal diseases may be extremely complicated and random, and largely based on the life of patients so far, which has not been monitored by doctors. To be able to provide some estimate of remaining life, doctors need to match each case with similar cases in the literature or some database and maybe make slight adjustments according to what seems to differ from these cases. These adjustments may be made according to characteristics of patients that clearly affect the progress of the disease. The doctor's understanding of the disease mechanisms naturally guide them towards better estimations but the most part of the estimates is often based on data.

One might argue that such approaches may not be appropriate, since data-driven methods sometimes do not take into account different characteristics of humans and may have bad results. However, no one can argue that these approaches do not work. Following such methods, human life expectancy has been steadily increasing [86]. Further understanding

of diseases is constantly an object of research, and it is expected that better understanding will lead to better diagnosis and treatment; but, until that is achieved, data-driven methods offer an excellent and efficient alternative.

Parallels between structural health monitoring and medical diagnosis are abundantly clear. Although both procedures include detection, localisation, diagnosis and remaining life calculation, SHM can be considered to be a more difficult task, because structures are much more diverse than humans (as far as their organism is concerned). A global equivalent to homeostasis for structures may not exist. Every structure has its own normal condition characteristics which may vary significantly according to environmental and loading conditions. In some cases, they may vary even *more* because of the environment than because of damage (Z24 bridge). Even structures of similar characteristics may have quite different normal-condition characteristics, such as first natural frequency. Geometry of structures varies a lot, something that does not happen that much with human bodies. However, trying to be optimistic, one can think that mechanical mechanisms of structures are not as complicated as those of the human body, structures can be cut, repaired, damaged on purpose without any ethical or humanitarian dilemmas. Studying them is a matter of human safety, but leaving economic issues aside, one can almost always provide relatively-safe solutions for avoiding loss of human life.

Having spotted the connection between the application of medicine based on human populations, an initial attempt to define methods for population based SHM was performed [47, 87–89]. The applications ranged from homogeneous to heterogeneous populations of structures. Different methods were developed with a view to exploiting knowledge about some of the structures of the population to build SHM systems for structures for which knowledge may not be available.

In its most basic case, PBSHM for *homogeneous* populations is performed by defining a *form* for the population. In [90], a 'form' is defined as a rule that the whole population should follow. This rule may refer to the first natural frequency of the structures or some frequency response function, which should be similar for all structures within the population, since the structures are quite alike. It is common that the rule will have some margins, within which the structures are allowed to move, similar to the margins of human medicine for blood test results etc. The approach performs quite well, and is

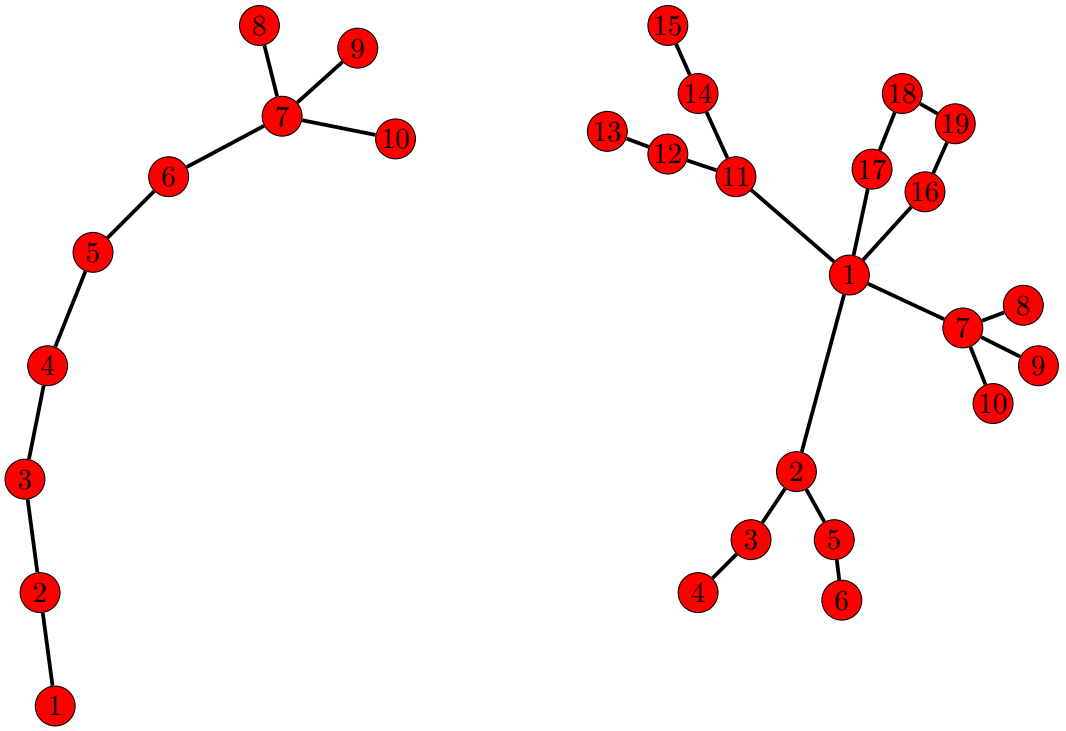


Figure 3.4: Graph representations of two structures following [91]: a wind turbine (left), and an airplane (right).

useful especially in cases of populations of structural members. Such members might be rolling bearings that are produced and used in many different structures; their monitoring can be performed following such a framework, which defines their normal condition characteristics.

For more diverse structures, different approaches are followed [91, 92]. Structures are converted into graphs, which represent the connectivity of their members. The graphs are created by considering every node to be an *irreducible element* (IE), and the edges to be the connections between the IEs. As described in [91], the philosophy of decomposing the structure into IEs is by considering the small parts of the structure, which have well-defined dynamics and might be common within different structures; for example, beams, plates etc. The connections between the IEs of a structure define the edges of the graph and describe the connectivity of the structure. A schematic example of transforming a wind turbine and an airplaine into graphs, presented in [91], is shown in Figure 3.4.

By performing such a transformation of the structures into graphs, one can detect common subgraphs or similarities between the structures. The similarities allow a clustering of the

structures with a view to defining subsets of the population, in between which knowledge transfer is more applicable, since structures within the subset are closer, according to some graph-distance metric. Moreover, one can detect common subgraphs within the population and try to focus on transferring knowledge only between the subgraph areas of the structures. For example, in Figure 3.4, nodes 7, 8, 9, 10 in both graphs form a similar subgraph, unlocking potentially local transfer of knowledge capabilities.

For a more detailed description of the structure, *attributed graphs* were used. It would be a very unlikely coincidence that every fundamental part or IE of every structure within the population is identical. The IEs are most probably made of different materials and have different dimensions. The same applies for the connections. IEs are connected by different types of connections, and with connections that have different attributes, e.g. bolted connections, welding connections etc. To make the graphs more informative, the different materials or their structural properties can be encoded as attribute vectors of the nodes and the edges of the graph; such an example is presented in the current thesis in Chapter 6. The properties allow better inferences when one tries to use the graphs within an algorithm. Furthermore, attributes allow a more detailed calculation of some distance metric between graphs. Structures with similar materials and similar connection types are more likely to be closer when their nodes and edges are attributed.

In [92], an application of knowledge transfer for heterogeneous populations is presented. The methods presented are motivated by transfer learning [93–96], and aim at developing classification models for some target domain $\mathcal{D}_t$ and task $\mathcal{T}_t$ that are based on models trained for a source domain $\mathcal{D}_s$ and task $\mathcal{T}_s$. The work in [92] exploits *domain adaptation* techniques [97–99]. The results of the applications reveal that the methods are able to efficiently transfer the distributions of the data and the labels from the source domain to the target domain, allowing to develop models with very good accuracy. Such techniques solve great issues with data-driven SHM and data-driven methods in general, since they provide data in cases where one has no available, or insufficient, data from a structure.

Similar methods have been applied to solve the problem of repairing a structure. By repairing a structure, the structural parameters change, thus making models which were used before the repair, inaccurate. In [82], similar transfer learning methods were used to overcome such a problem. The repaired structure is considered a different structure, and

transfer learning is employed in order to make existing models able to predict accurately after the repair procedure.

Population-based structural health monitoring is introduced as a way of addressing the problem of lack of data when one needs to monitor the behaviour of structures within a population. Essentially, it is motivated by the way that modern medicine is performed and more specifically *syndromic surveillance*, which is the discipline of epidemiology, of monitoring common health trends in populations. By identifying such trends of damage within structures, the proposed methods can be used to anticipate and prevent similar problems in new and undamaged structures. The framework is also able to exploit databases and knowledge bases, like the one that will be described in Chapter 4. PBHSM can be considered as a natural step forwards for the discipline of SHM but also for data-driven structural modelling in general, since it deals with the major issue of lack of data.

Chapter 4

Developing an SHM ontology

4.1 Definition of an ontology

There are many definitions for ontologies, but the one that best describes the funtionality of an ontology, as defined here, is given in [100]: "An ontology is a specification of a conceptualisation". The definition implies that the ontology is a way to make more clear the definition of a concept. The specification is achieved by definition of different elements/classes of the concept and introduction of connections between these classes that define the semantics of their interaction.

Ontologies are often developed in programming languages such as OWL [101]. However, it is easier to develop an ontology using software with a user interface, such as Protege [102] and GATE [103]. Using the software, one can easily define different elements of the ontology and also their interactions. Moreover, objects that belong to classes can be created and even be automatically clustered into classes according to their characteristics and axioms/rules defined by the user. A snapshot of the Protege software is shown in Figure 4.1.

An ontology is essentially created by defining its various components. The basic types of components that make an ontology are:

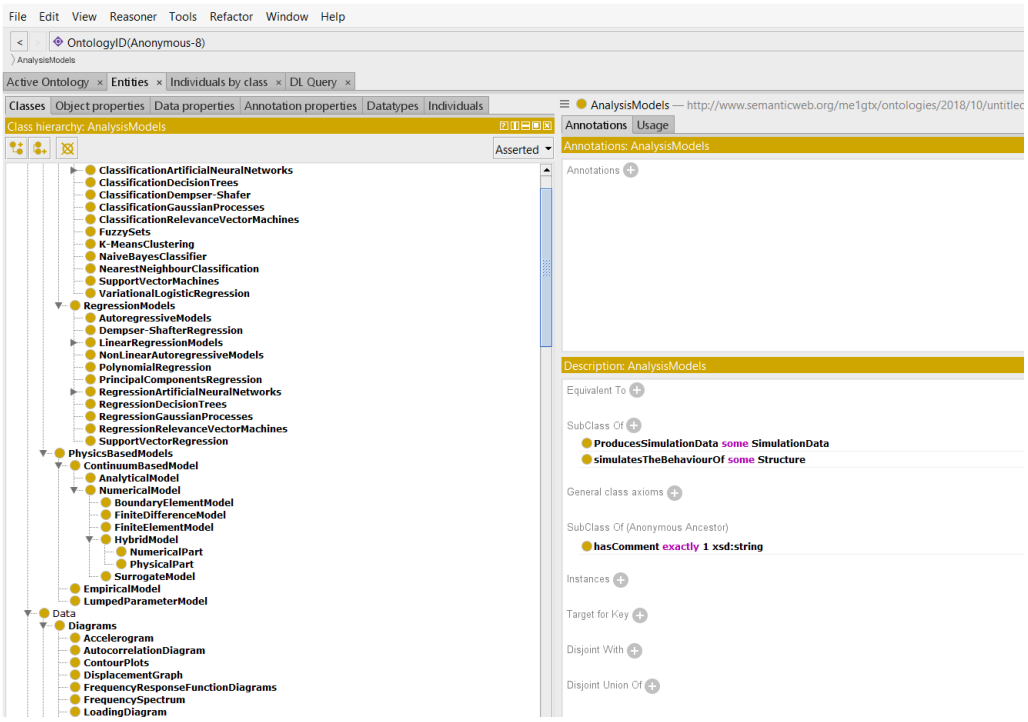


Figure 4.1: Protege ontology editor.

1. **Individuals**: the members of the ontology. Individuals are objects that belong to the classes. They inherit characteristics from the classes that they belong to and they constitute the population of objects that the ontology explains.

2. **Classes**: clusters of individuals which share some characteristics. Classes are created by collecting individuals with similar attributes and properties. Subclasses can also be created. Individuals of the subclasses inherit characteristics from the superclass as well.

3. **Connections**: are relationships between classes. The connections define how classes interact with each other. The define the exact semantics of the link of two classes. The connection relationship between classes can potentially be inherited to individuals of the classes.

4. **Attributes**: are the features of the individuals. They are qualitative or quantitative characteristics that an individual may have. Often they refer to values of some feature that varies across the total population of individuals. Specific values of these characteristics may be the way to define a class of individuals, e.g. some class A

might be defined by collecting all individuals whose value of the attribute a is equal to n.

5. **Properties**: are the connections which define the relationships between individuals and their attributes. A property is similar to a connection, but instead of the semantics of the link between classes, it defines the semantics of the link between individuals and their attributes.

6. **Annotations**: are pieces of text that give further information about objects of the ontology. Essentially, annotations are notes, including definitions and further instructions on how parts of the ontology work.

The elements above are the main objects used in order to define an ontology about SHM here. The ontology is focussed on organising the knowledge that might be needed in order to implement an SHM project. Its construction begins by simply defining a hierarchy of classes, resulting in a *taxonomy* or a tree graph of them. Subsequently, more connections between the classes will be defined, resulting in a more complex construct, but at the same time more informative about the functionality of the objects of the ontology.

4.2 Core SHM ontology classes

The creation of the ontology essentially begins with the reasons of beginning an SHM project. The reasons are often based on the the maximisation of the economic benefit of a structure, but might also be the preservation of a structure of historical significance. Another important aspect that needs to be taken into account before creating such an ontology, is the restrictions that are set by the environment. Quite often one is not able to attach any sensor on a structure, because of the environmental conditions. An example of restrictions might be that the structure is in space and therefore special sensors have to be placed on it in order to perform in extreme conditions or that the structure is a structure of great historical significance and people visit it quite often, so sensors placement on the structure might be restricted. Therefore, before the creation of the ontology, the goal of the SHM project and the restrictions have to be defined.

Having defined the reason of the SHM project and identified the restrictions, the first step needed in order to define the ontology is to define its core classes. The ontology aims at explaining the relationships between elements that take part in an SHM project. Therefore, the superclasses, which are chosen as the core of the ontology should be able to include every element or class that is further used in SHM. These are,

1. analysis models;
2. data;
3. data processing;
4. physical parts;
5. SHM methods.

The first four classes include distinct elements, such as methods, and structures that are monitored. The fifth class is the one that combines all of the above in a way to define the different SHM methods that one might essentially use to perform inference about structures. This latter class essentially combines elements from other classes in order to define algorithms to perform SHM. The top classes and some of their subclasses are shown in Figure 4.2.

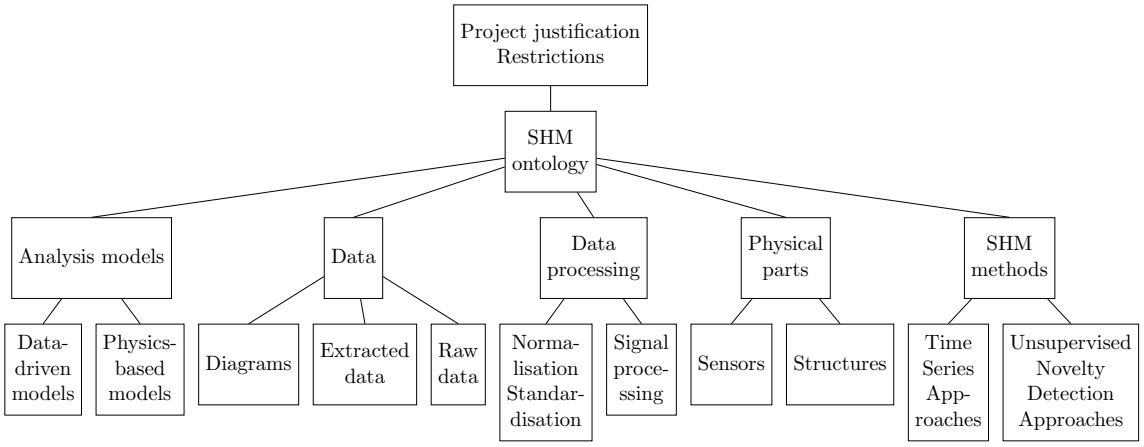


Figure 4.2: The five superclasses and some of their subclasses.

The first superclass contains different types of models that might be used in order to perform the desired monitoring. A major separation of such models is between physics-based models and data-driven models. These are also the two subclasses of the superclass

'analysis models'. The physics-based models include algorithms that are based on making predictions according to knowledge of the underlying phenomena that they model, e.g. the finite element method [7]. The second subclass, the data-driven models include models that rely exclusively on data. Machine learning [8] models are included in this class. Common types of such models are neural networks [9], K-means clustering [104], support vector machines [105], autoassociative neural networks [52] etc. Data-driven models are further split into classification and regression models. Both types are useful when one is monitoring a structure. Of course, combinations of the two methods exist - the hybrid models. Such models shall be individuals within the ontology that belong to both classes of 'data-driven models' and 'physics-based models'.

The second superclass, 'Data', contains any type of data that one might acquire from a structure. For SHM, data are really important, since they are the way to make contact with the structure and make inference about its condition. Data are required for both data-driven and physics-based models. For the first case, data are the way of defining the model and making inference. For the second case, data are needed in order to calibrate the models or to quantify uncertainty in the prediction, as well as to use them on acquired data to evaluate the condition of the structure. The class includes subclasses such as "raw data", which are the data in the form acquired from the structure, "simulation data", which may come from a model and "processed data", which are the product of the application of some signal processing method on raw data. A further subclustering could be to define the type of data such as "accelerograms", "contour plots", "mode shapes", "displacement simulation data", "displacement sensor data", etc.

The next superclass, "Data processing", contains methods that are traditionally used in order to process raw data coming from sensors. These methods include fundamental statistical analysis methods and transformations such as normalisation and standardisation schemes, envelope calculation, averaging, the Fourier transform [106], etc. More sophisticated methods such as the wavelet decomposition [107] are included as classes. Some examples, of subclasses within this superclass are "signal smoothing algorithms", "Hilbert transform", "principal component analysis", etc.

The fourth superclass includes tangible elements. Every monitoring scheme is focused on some structure, from which the data are acquired and whose condition is evaluated. The

sensors used are also included in this class in order to have a clear definition of the type of data that is acquired from each one, and also the models that rely on each sensor to make inference. Within population-based schemes [90–92], the different structures of the population should be included in the ontology in order to have a clear distinction of the data acquired from each one. At the same time, the different models that are available are connected to the structure that they refer to and provide a more clear image of the connectivity between models, data, sensors and structures.

The final superclass is the one containing all the SHM methods that are used in an SHM project. The various methods combine models and data in order to evaluate if a structure is damaged or not and what type of damage might exist. Almost every method refers to a structure or a part of it and uses data from the data class. Moreover they exploit the available models from the "model" class. Some types of SHM methods that are included, are "acoustic emission monitoring" [108], and "guided wave approaches" [109], whose data come from receivers trying to "hear" cracks in materials and ultrasound receivers correspondingly. Similar methods are "novelty detection methods", which are unsupervised damage detection methods [110]. The list of the developed ontology is not exhaustive and can of course be extended according to the needs of the users or some particular project.

4.3 Connections

The ontology, as defined so far is a simple taxonomy of classes that are connected only via the class-subclass connection. In order for the ontology to become more comprehensive, more connections should be defined between various classes and their individuals. Some of these connections are straightforward, such as that raw data are acquired from some sensor. Therefore, instances of raw data should be connected with instances of sensors via the 'is acquired from" type of connection.

By adding connections, the functionality of various components of the ontology are better described. Various models exploit data in order to make inferences and so have to be connected with the data using a connection called "uses data of type". The same model,

or an SHM method, might exploit a specific data processing method and as a consequence be connected to it via a connection called "processes data using".

Similar connections are defined between classes of the ontology, and the class whose elements create the most connections within the ontology is the "SHM methods" class. The class, as mentioned, includes various algorithms used in the SHM discipline. Such algorithms often combine many elements from different disciplines into one algorithm. An example of a novelty detection method is shown in Figure 4.3, where different connections are shown between various other classes of the ontology. The method is a simple autoencoder novelty detection method, like the one described in [111]. The class of the method is connected with various other classes, such as the class of the standardisation method that will be used for preprocessing of the data, the class of structure sensors that will provide the data in order to apply the method, the class of the AANN model that is used as part of the method, etc. Instances of the method are also connected to an attribute value, the novelty index that they yield. This whole connectivity explains the functionality of the class and its interaction with other classes of the ontology.

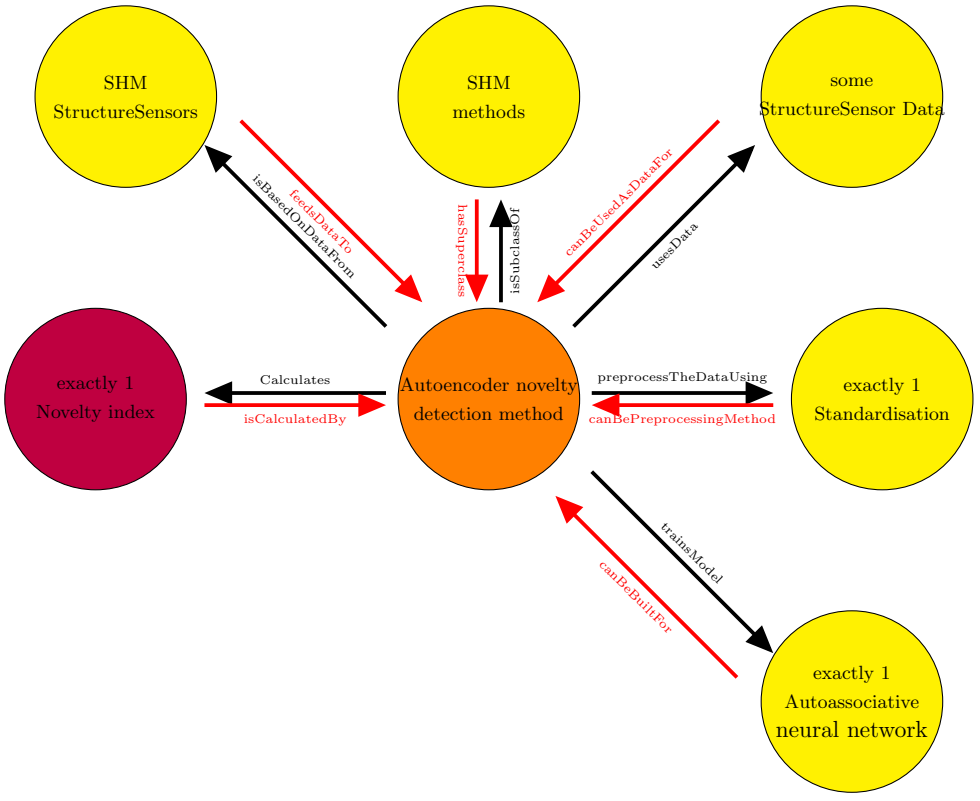


Figure 4.3: Example of a novelty detection method as it is represented in the ontology.

Equally important to defining connections, *inverse connections* have to be defined. Inverse connections fully define the interaction between a class and the rest of the classes of the ontology. A first important aspect of inverse connections, is the extensive explanation of the functionality of every instance of every class of the ontology. The existence of a class affects other classes, but is also affected by them, and inverse connections explain the latter effect. A second important property of these connections is that they define which classes are affected by the introduction of a new class or individual and how exactly are they affected. They can be thought of as a guide pointing towards what needs to be adapted when the ontology is extended by the addition of a new class. Therefore, they assist in keeping every class of the ontology up-to-date with the latest additions to it. In the example shown in Figure 4.3, the autoencoder novelty detection method uses some data from sensors, but the class of sensors should also be updated in order to be able to provide data to the method, as the inverse property dictates.

4.4 Aspects of the ontology

An ontology defined for SHM is a versatile diverse object and can be exploited in many different ways. The first and most straightforward way is to use it for the purpose that ontologies are originally built, to share knowledge. The various classes and methods that are included in the ontology describe how an SHM project might function, which quite often shall be a multidisciplinary project. Moreover, via the connections between the classes, one can further understand their interactions and their role in the project. The annotations and the descriptions of the classes provide further explanation of exactly how each part of the ontology takes part in the SHM campaign. It is expected that by using such a tool to share knowledge between people from different scientific backgrounds, the total time needed in order to coordinate such a diverse group, shall be much lower than if done in absence of the ontology.

A second aspect of the ontology might be the assistance in extending the existing knowledge and to define more efficient methods to perform SHM. The modularity of graphs such as the ontology can be exploited for this purpose. As shown in Figures 4.4 and 4.5, some SHM method might use algorithms that belong to some wider category of algorithms. By

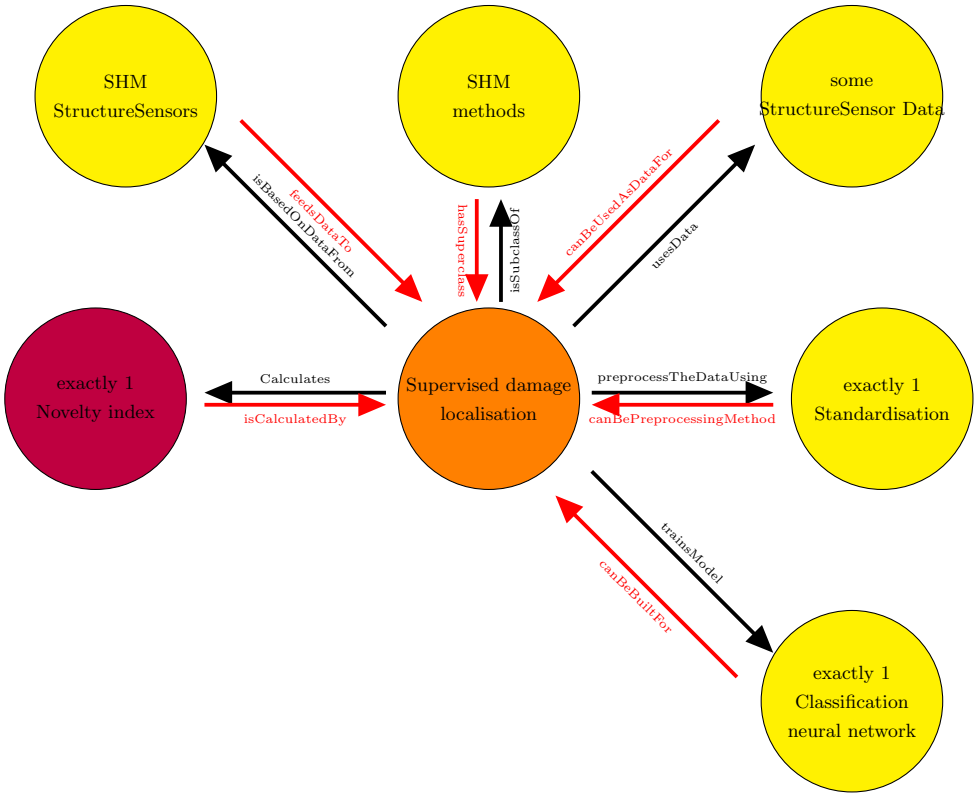


Figure 4.4: Initial method of supervised damage localisation.

exchanging one method (from a neural network to support vector machines in this case) for another from the same superset of algorithms might lead to more efficient SHM methods; this might be because of the nature of the data and the fact that some algorithms might be able to perform better in some datasets. Following such an approach might allow automation of the application of an SHM scheme, simply by trying different algorithms and considering the one that performs best. The same applies for the data or the sensors from which the data come from. One can exploit different sources of data or different datasets from a structure by shuffling the nodes of the ontology according to the rules of connectivity; i.e., exchange some sensor of a specific structure for a sensor of the same structure and not a different structure.

It also becomes clear that the concept of ontologies is quite close to the philosophy of *object-oriented programming*. The specific programming strategy has been a convenient way of structuring large software projects, because of the reusability and modularity of its elements. The ontology can provide a map of how to structure such a software project for SHM. An advantage of such an approach shall be that the various coded methods can

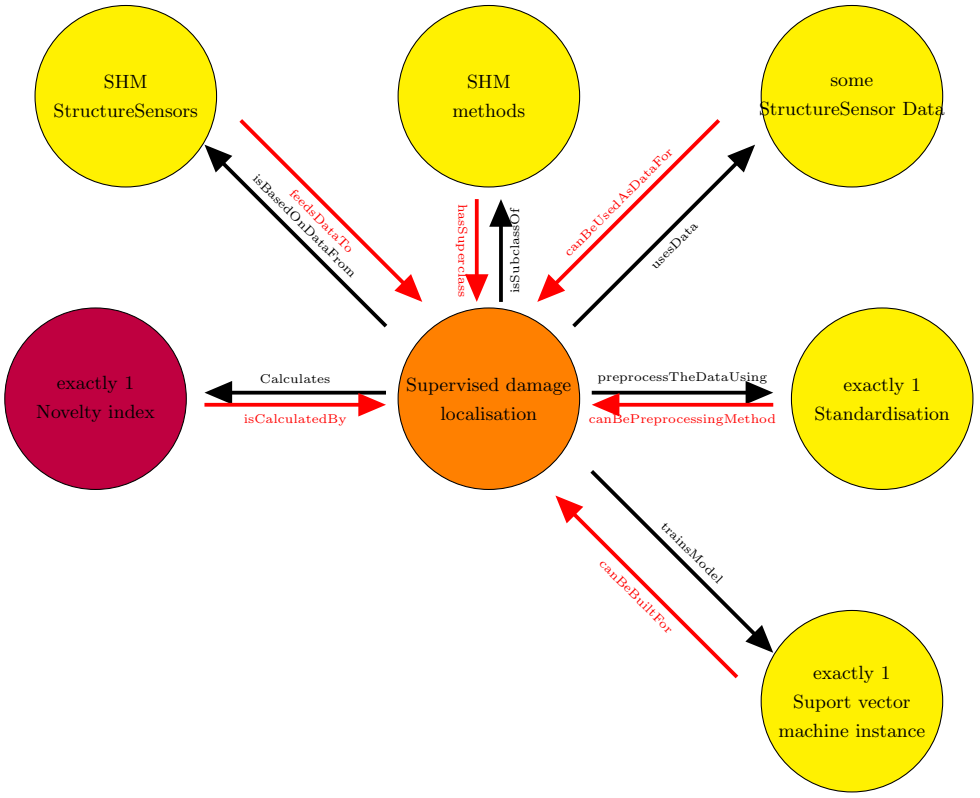


Figure 4.5: Alternative method of damage localisation created by switching inference algorithm.

be reused, as well as the self-explainability of the code. The ontology, apart from a guide in order to build such software, is also an explanation of how it works, and provides the software developers with the knowledge they may need to implement the various methods.

Finally, the ontology can serve as a database. The layout of the SHM ontology allows inclusion of various structures, parts of the structure, sensors and types of data. This, together with the characterisation of instances via attributes, might consist of a proper way to organise data for an SHM project. Especially by including different structures, the ontology can be very useful for a population-based SHM campaign. The attributes of the different sets of data can assist in navigating through the dataset and also through the various available models. For example, one might be interested specifically on finite element models that have a certain number of degrees of freedom and so, they can look for individuals of the ontology of the class 'Finite element models' that have attribute 'degrees of freedom' greater than a specific number.

The ontology seems an appropriate way of organising the data, the knowledge and the

models that are available for an SHM project. It can moreover be expanded in order to serve the same purpose for digital twins. Digital twins can be much more than a simple model. They might include many different models, lots of data and methods and even more than one structures. Having such a variety of objects that need coordinating and sharing, the ontology might prove quite useful by clearly defining interactions and sets of similar objects that can be used for similar purposes.

4.5 Summary

Concluding, the ontology appears to be a useful tool with which to organise a project of monitoring structures. The presented framework allows a comprehensive inclusion of data, models, and methods, which are often used in such a framework. A schematic representation, which describes the functionality of different entities that are used during the project, is also created via the use of the ontology. The various entities and their interactions are explained and knowledge can more easily be shared between members which participate in the project. The methods that are described in following sections can be included in the ontology; in this way, their role in the project and their interaction with physical objects (such as sensors and structures), as well as with existing methods and data is explained.

The framework could also include various structures and their corresponding data and models. This approach could assist in identifying similarities between structures and in applying population-based SHM. Organising knowledge and data in an ontological manner, could be a way to motivate and promote *smart structures*. The ontological approach facilitates knowledge exchange between projects, making it easier to apply similar methods across different projects. The ontology could as well facilitate generation of methods by some artificial intelligence agent, since the methods and the components that are used by the methods are presented as graphs, which can be processed by such agents.

Chapter 5

Generative adversarial networks for modal analysis

5.1 Modal analysis

As mentioned, the a project (and as a consequence an ontology referring to the specific project) might include different types of models. Models that perform predictions, models that monitor some structure, etc. A great part of modelling systems is often focused on *decompositions*. The process of decomposition is aimed at extracting components of the quantities of interest which have certain properties. Decomposition might be as important as predicting future values of quantities of interest. By decomposing data acquired from a system, one can study and understand further the behaviour of the system. A great deal of effort has been spent on developing decomposition methods, and some of them can be considered as important as the inference algorithms described before; e.g. principal component analysis (PCA) [51]. A digital twin, being a collection of data and models which describe the behaviour of a system, may also include such models. Enriching a digital twin with decomposition models offers the opportunity to the users to have a more comprehensive view of its behaviour, and the ability to make decisions according to an augmented set of features of the system.

A property that one often seeks when decomposition is applied is *dimensionality reduction* of the data. Multidimensional data often live in subspaces of the full multidimensional feature space. By decomposing the data one can describe them, with minimal loss of information, using less variables than the ones needed for the whole space. This reduction facilitates storing the data, studying them and applying machine learning methods, since data of large dimensions make model fitting harder and create the need for acquisition of large quantities of data (the curse of dimensionality). Two widely-used methods for dimensionality reduction are PCA [51], and the use of autoencoders [52]. The first method calculates components of the data so that each captures variance of the data in decreasing order. The decomposition is linear and its calculation has a closed form. The second decomposition method is based on using neural networks with a bottleneck layer in order to decompose the data and then reconstruct them, maintaining as much information as possible. This practice forces the algorithm to project the data on a space of smaller dimension than their original feature space. The advantage of the algorithm is that it can provide *nonlinear* projections, but the calculation of the decomposition is based on an iterative process, therefore does not have a closed form.

Another desired property of the decomposed data is the independence of the components. In the case of multidimensional data, a break down into components, which function separately would allow for studying them independently. Humans have difficulties studying data with many components which interact all together. The decomposition into independent components would make the solution of the equations more tractable and the understanding of the engineer more thorough.

For structural dynamics, the method that has dominated the analysis of the vibration of structures is *modal analysis* [19, 112, 113]. The specific type of analysis offers a decomposition of the movement of structures into *modes* - meaning ways of motion. The modes have several properties, which, in the case of linear structures are time-invariance and reciprocity and allow one to study them almost separately and to make inference about the structure according to them. In order to extract the modes for linear structures and following notation from [19], the equation of motion of an unforced linear system is given by,

$$[m]\{\ddot{y}\} + [k]\{y\} = 0 \tag{5.1}$$

where $\{y\}$ is the n-dimensional displacement vector of the structure, $\{\ddot{y}\}$ is the corresponding acceleration vector, $[m]$ is the $n \times n$ mass matrix and $[k]$ is the $n \times n$ stiffness matrix of the structure. Assuming the excitation to be harmonic, the solution to the differential equation above is,

$$\{y(t)\} = \{\psi\}e^{i\omega t} \tag{5.2}$$

where $\{\psi\}$ is a constant vector. Substituting equation (5.2) into equation (5.1) yields,

$$-\omega^2[m]\{\psi\} + [k]\{\psi\} = 0 \tag{5.3}$$

which is a standard linear eigenvalue problem and, assuming that the mass matrix is invertible (which usually it is), yields,

$$[m]^{-1}[k]\{\psi_i\} - \frac{1}{\omega_i^2}\{\psi_i\} = 0 \tag{5.4}$$

where ω_i and $\{\psi_i\}$ are the i^{th} solutions to the eigenvalue problem.

The eigenvalues ω_i of the problem of the equation above are actually the eigenfrequencies/natural frequencies of the structure which is studied. The corresponding vectors $\{\psi_i\}$ are the modeshapes which, as shown in [19], are orthogonal with respect to the mass and stiffness matrix; this means that,

$$\{\psi_i\}^T[m]\{\psi_j\} = 0 \quad i \neq j \tag{5.5}$$

$$\{\psi_i\}^T[k]\{\psi_j\} = 0 \quad i \neq j \tag{5.6}$$

This latter property of the modeshapes allows the decomposition of the system of equations of motion. Consider the system of equations of motion given by,

$$[m]\{\ddot{y}\}(t) + [c]\{\dot{y}\}(t) + [k]\{y\}(t) = \{F\}(t) \tag{5.7}$$

where $[m]$ is the mass matrix of the system, $[c]$ is the damping matrix, $[k]$ is the stiffness matrix and $\{F\}$ is the external load vector at time t. By substituting the displacement

vector $\{y\}$ by the calculated modal decomposition given by,

$$[\Psi]\{u\} = \{y\} \tag{5.8}$$

where $[\Psi] = [\{\psi_1\}, \{\psi_2\}...\{\psi_n\}]$ is the modal matrix and left-multiplying by $[\Psi]^T$, equation (5.7) becomes,

$$[\Psi]^T[m][\Psi]\{\ddot{u}\}(t) + [\Psi]^T[c][\Psi]\{\dot{u}\}(t) + [\Psi]^T[k][\Psi]\{u\}(t) = [\Psi]^T\{F\}(t) \tag{5.9}$$

Because of the orthogonality condition in equation (5.5), the matrices $[M] = [\Psi]^T[m][\Psi]$ and $[K] = [\Psi]^T[k][\Psi]$ are diagonal. The damping matrix is often assumed to be a linear combination of the mass and stiffness matrices (e.g. Rayleigh or proportional damping) and therefore, the matrix $[C] = [\Psi]^T[c][\Psi]$ is also diagonal. Taking into account the aforementioned orthogonality conditions, the equations of motion yield,

$$[M]\{\ddot{\psi}\}(t) + [C]\{\dot{\psi}\}(t) + [K]\{\psi\}(t) = \{f\}(t) \tag{5.10}$$

where $[M]$, $[C]$, $[K]$ are all diagonal and $\{f\}(t) = [\Psi]^T\{F\}(t)$. The fact that the matrices are diagonal means that the solution of the n-degree-of-freedom system of equation (5.7) is reduced to the solution of n differential equations given by,

$$M_{ii}\{\ddot{\psi}_i\}(t) + C_{ii}\{\dot{\psi}_i\}(t) + K_{ii}\{\psi_i\}(t) = f_i(t) \quad i = 1, 2..., n \tag{5.11}$$

where M_{ii}, C_{ii}, K_{ii} are the i^{th} diagonal element of the corresponding matrices and f_i is the i^{th} element of the vector $[\Psi]^T\{F\}$.

Solving the n equations (5.11), and combining the solutions according to equation (5.8), is obviously a much easier task than solving the global equation of motion (5.7). Moreover, the components $\{\psi_i\}$ of the total motion of the structure oscillate with corresponding frequencies ω_i; a realisation that allows determining which modes will be affected by some external forcing with specific frequency content.

Modal analysis, apart from a proper decomposition of the movement of linear systems, offers a way of reducing the number of degrees of freedom of a system. Modern finite

element [7] models have thousands or even millions of degrees of freedom and the solution of equation (5.7) is not computationally efficient. By calculating the modal matrix $[\Psi]$ for a number of modes, which explain a sufficient percentage of the structure's motion, one can reduce the system of equations of motion to independent equations equal to the number of modes considered. This strategy can also be followed when only a subset of the modes of a structure are of interest.

Another way of exploiting the decomposition, is by isolating the modes that are affected by some specific excitation. As mentioned, each mode refers to oscillations of some specific frequency. Therefore, excitations with specified power spectral densities would affect only specific modes. During the design phase of a structure, a common tactic would be to design the structure so that no major modes are affected by the frequencies of the forcing which contain the most energy.

It might seem that modal analysis is a perfect tool for analysis for structure, and it might be, but only for linear structures. For structures with members which exhibit nonlinear behaviour, modal analysis fails to provide the properties of orthogonality and independence. Even the calculation of the modes would not be so straightforward. The failure of modal analysis for nonlinear structures is also expected, since the decomposition of equation (5.8), is based on the linear superposition of ways of motion, which stands only for linear structures; i.e., the sum of the motions corresponding to two distinct external excitations is equal to the motion corresponding to the sum of the excitations.

In order to decompose the motion of nonlinear structures, in the current work, machine learning is employed. Since modal analysis is often applied on existing structures exploiting acquired data, machine learning methods seem like a natural way to process the data and extract decompositions that serve the purposes of modal analysis. In the following sections, the machine learning algorithm which was selected for the purposes of the current work, as well as the reasons and the goals of its application are described. Subsequently, results of its application using simulated and experimental data coming from a structure with harsh nonlinearity are presented. In a digital-twin framework, such a tool would be very useful, allowing a completely data-driven decomposition of the motion of a structure into independent components. These components can then be exploited by other parts of the digital twin in order to perform inference.

5.2 Cycle-consistent generative adversarial network (cycle-GAN)

5.2.1 Problems of GANs

Generative adversarial networks, as described, are used to map some random noise vector onto the manifold of the real data. However, this mapping is not subjected to any restrictions, and therefore there is no control over how the noise will be mapped onto samples of the target feature space. A common problem with this approach is the *entanglement of features*. Since there is no restriction on how the noise might be transformed into data samples, it is quite possible that noise variables will not encode any meaningful features of the data, and even that many different noise samples will correspond to the same generated sample.

In order to illustrate the problem of entanglement of features, an SHM example is presented. The problem refers to the simulated three-degree-of-freedom system shown in Figure 5.1. The system is subjected to a white noise excitation on its first mass. Moreover, damage is simulated via stiffness reduction of its springs. Two damage cases are considered. The first damage case refers to stiffness reduction of k_2 and the second of stiffness reduction of k_3. The stiffness reductions considered for each spring are within the interval $[0\%, 20\%]$ and according to the Cartesian product $[0\%, 20\%] \times [0\%, 20\%]$. Collecting samples of concatenated transmissibilities between the first and the second and the second and the third masses for every combination of stiffness reduction, a dataset is created.

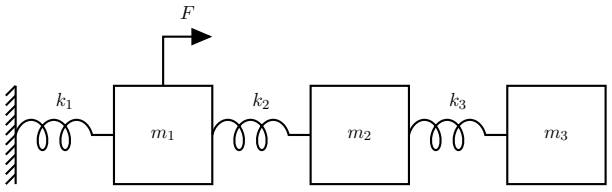


Figure 5.1: Three-degree-of-freedom mass spring system.

In order to visualise the dataset, *principal component analysis* (PCA) [51] was performed on the data. The resulting dataset can be seen as the blue points in Figure 5.2a. The

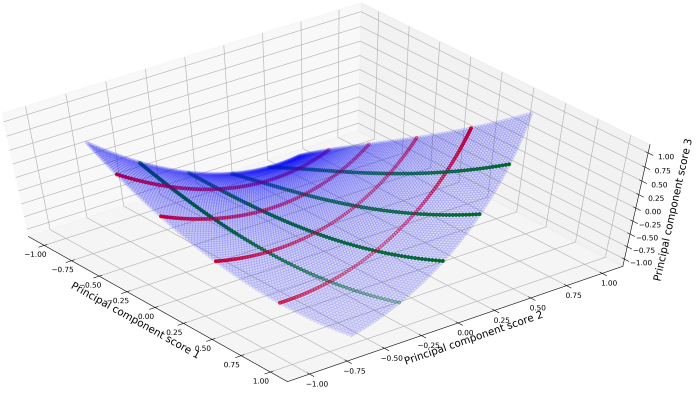
manifold shown in the aforementioned figure is parametrised exclusively by the two stiffness reduction parameters. The contour curves shown in red and green represent points that have a constant value for one of the two parameters, while the other varies.

In the specific example, the problem of entanglement becomes clear if one tries to fit a GAN on the created dataset. The underlying parameters of the dataset are the two stiffness-reduction percentages. Therefore, the noise vector of the GAN is selected to be a two-dimensional Gaussian noise vector. Ideally, after training, each noise variable should approximately correspond to one of the damage parameters. However, this does not happen. A potential result of training is shown in Figure 5.2b. The green and red curves in the latter figure represent points with one of the two noise variables being constant and the other one varying. The entanglement of the features is clear and expected, since there is no restriction targeting avoiding such situations.

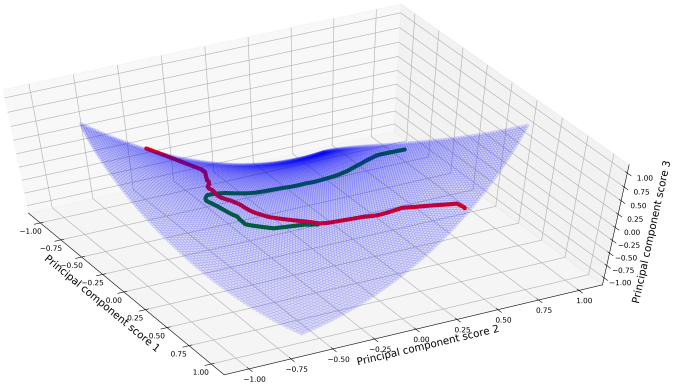
5.2.2 The property of invertibility

In the current work, the property of invertibility is studied as a way to assist avoiding entanglement situations like the one shown before. Apart from noise variables not corresponding to explicit underlying features of the data, the entanglement also refers to different noise samples generating the same artificial sample. In order to solve this issue and move a step closer to the desired disentanglement, the invertibility of the mapping, which GANs provide, is proposed.

Given a bijection $\phi : M \to N$, where M and N are manifolds, it is straightforward to show that if c_1, c_2 are curves in N, then $\phi^{-1} \circ c_1$ and $\phi^{-1} \circ c_2$ are curves in M with the same number of points of intersection. Considering $\phi : M \to N$ and two curves $c_1, c_2 \in N$ then $\phi^{-1} \circ c_1 = c_1'$ and $\phi^{-1} \circ c_2 = c_2'$ are curves in M. For every point of intersection t_i of the two curves c_1, c_2, it stands that $c_1(t_i) = c_2(t_i)$, which leads to $c_1'(t_i) = c_2'(t_i)$ because ϕ is a bijection. Therefore, c_1 and c_2 have the same points of intersection as c_1' and c_2'. Consequently, by introducing the inductive bias of invertibility in a GAN, helps to avoid mappings like the one shown in Figure 5.2b, where two orthonormal axes from the latent variable space are mapped onto the entangled green and red lines. Enforcing invertibility



(a)



(b)

Figure 5.2: First three principal component scores of the dataset (blue). On the top, points corresponding to constant damage for spring 1 and varying damage percentage for spring 2 (red), and points corresponding to constant damage for spring 2 and varying damage percentage for spring 1 (green). In the bottom, GAN-generated samples (red and green) by locking one latent variable to 0 and varying the other in the interval $[-1, 1]$; regular GAN used.

might force the GAN to more efficiently achieve a disentanglement of the features of the data.

In order to illustrate the effect of imposing invertibility to the mapping, a cycle-consistent generative adversarial network (cycleGAN), which will be described later, is trained using the same dataset. The results are shown in Figure 5.3. It can be seen that indeed, the entanglement which is observed in Figure 5.2b, is not present and that the mapping is

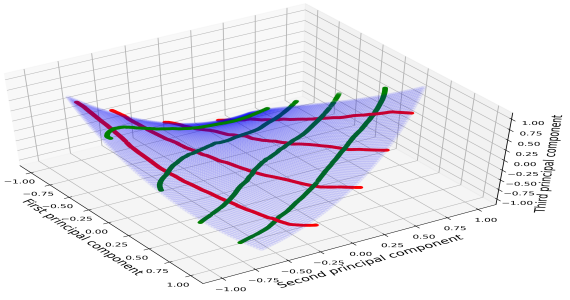


Figure 5.3: First three principal component scores of the dataset (blue) and cycleGAN-generated samples (red and green) by locking one latent variable to various constant values and varying the other on the interval $[-1, 1]$.

closer to the desired mapping according to the underlying parameters, shown in Figure 5.2a.

For the purposes of the current work, invertibility has a second and more important meaning. It is a fundamental part of modal analysis. In modal analysis, it is required to have both a forward mapping from the physical space to the modal space and inverse mapping. The inverse mapping ensures that the forward mapping is meaningful, since, if the inverse mapping does not exist, then the modal space could eventually be an arbitrary space that satisfies the orthogonality restrictions but has no physical meaning.

5.2.3 The cycleGAN algorithm

The algorithm of *cycle-consistent generative adversarial networks* (cycleGANs) [69], is used herein as a way to impose the desired invertibility property to the GAN. The algorithm was originally developed in order to translate figures to different styles. For example, to transform photographs into paintings of famous artists. In order to achieve such a task, the algorithm learns a mapping and its inverse between two domains in an unsupervised manner.

The architecture of the cycleGAN is schematically shown in Figures 5.4 & 5.5. The training procedure is similar to training a GAN but with some additions. The first addition is that the procedure is followed in order to learn two transformations, one from some domain X to some domain Y, and the other its inverse transformation. Therefore there are

two generators ($G_{X\to Y}$ and $G_{Y\to X}$) and two discriminators (D_X and D_Y). Training is performed in two stages. The first stage is similar to training a GAN for a mapping from domain X to domain Y, and the second is the inverse problem. As a consequence, the generators learn the sought after mappings between the two domains. The discriminators, as in the case of GANs, act as auxiliary networks. The adversarial loss $\mathcal{L}_1$, in similar terms as in the GAN, is given by,

$$\mathcal{L}_1(G_{X\to Y}, D_Y, X, Y) = \mathbb{E}_{\boldsymbol{y}\sim p_y(\boldsymbol{y})}[\log D_Y(y)] + \mathbb{E}_{\boldsymbol{x}\sim p_x(\boldsymbol{x})}[\log(1 - D_Y(G_{X\to Y}))] \quad (5.12)$$

where p_y is the probability density function of the samples in domain Y and p_x the corresponding probability density function of samples in domain X.

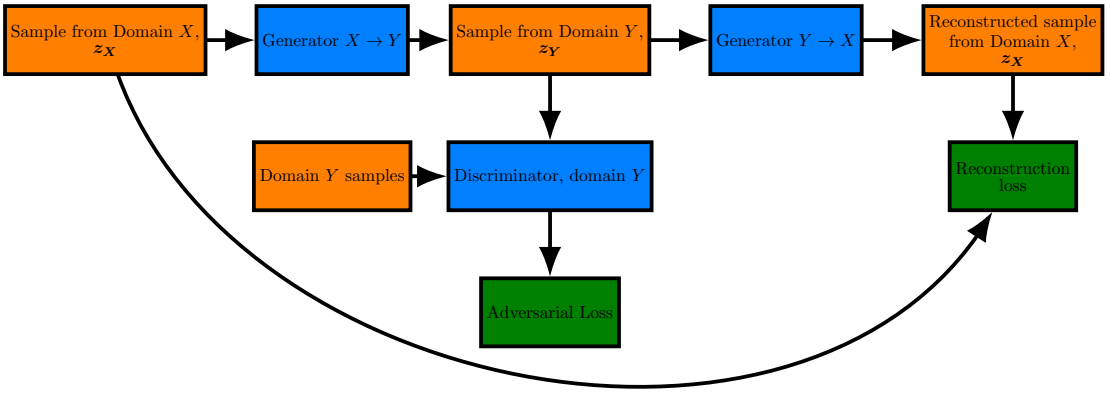


Figure 5.4: Cycle GAN layout assembled in order to learn the mapping from domain X to domain Y and back. Samples are converted from X to Y by the generator $G_{X\to Y}$. The generated samples are used for adversarial training of the discriminator D_Y and the generator $G_{X\to Y}$. Subsequently, the samples are inverse-mapped back to domain X via generator $G_{Y\to X}$ and both generators are trained using the reconstruction-loss error.

The second, and most important difference compared to training a GAN, is the addition of the *reconstruction loss*, which enforces the desired invertibility. Every sample mapped from one domain to another is also mapped back to its original domain by the corresponding generator and the error between the original and the reconstructed samples is calculated.

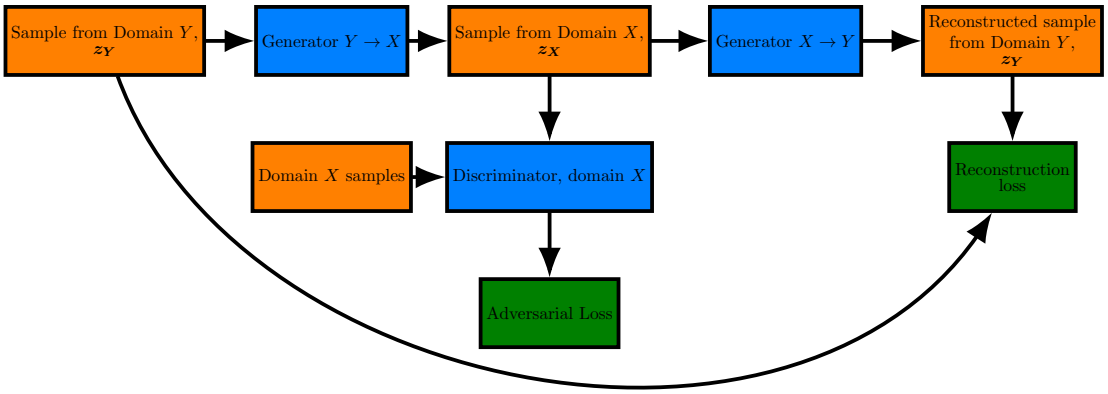


Figure 5.5: Cycle GAN layout assembled in order to learn the mapping from domain Y to domain X and backwards. Anti-symmetrical to the procedure shown in Figure 5.4.

The reconstruction loss (or *cycle loss* in the original work) $\mathcal{L}_2$ is given by,

$$\begin{aligned}\mathcal{L}_2(G_{X\to Y}, G_{Y\to X}) = \mathbb{E}_{\boldsymbol{x}\sim p_x(\boldsymbol{x})}[\|G_{Y\to X}(G_{X\to Y}(\boldsymbol{x})) - x\|_n] + \\ \mathbb{E}_{\boldsymbol{y}\sim p_y(\boldsymbol{y})}[\|G_{X\to Y}(G_{Y\to X}(\boldsymbol{y})) - y\|_n]\end{aligned} \tag{5.13}$$

where $\|\ \|_n$ is the n-norm. The norm used here is a second-order norm.

The total loss of the cycleGAN is calculated via,

$$\begin{aligned}\mathcal{L}(G_{X\to Y}, G_{Y\to X}, D_X, D_Y) = \mathcal{L}_1(G_{X\to Y}, D_Y, X, Y) + \\ \mathcal{L}_1(G_{Y\to X}, D_X, Y, X) + \\ \lambda\mathcal{L}_2(G_{X\to Y}, G_{Y\to X})\end{aligned} \tag{5.14}$$

where λ is a parameter that controls the magnitude of the reconstruction loss, compared to the adversarial losses. The suggested value in the original paper is $\lambda = 10$. Finally, the optimisation problem to-be-solved is,

$$G^*_{X\to Y}, G^*_{Y\to X} = \min_{G_{X\to Y}, G_{Y\to X}} \max_{D_X, D_Y} \mathcal{L}(G_{X\to Y}, G_{Y\to X}, D_X, D_Y) \tag{5.15}$$

Following the training scheme of the cycleGAN, and considering the physical space to be X and the modal space to be Y, a mapping and its inverse between the two is defined.

For consistency with [114], the modal space from now on will be referred to as U and the physical space as Y. In order to enforce even more characteristics of modal analysis into the training procedure, in the next paragraph, an orthogonality assembly of neural networks is proposed to enforce the orthogonality of the mode shapes.

5.2.4 Orthogonality inductive bias

Although orthogonality has been imposed in previous works [115, 116], during GAN training, a different approach is needed in order to perform modal analysis using GANs. Modal analysis includes different forms of orthogonality; statistical orthogonality is one of them, in the sense that modes should be statistically independent and their value should not be predictable from the values of the others. A second type of orthogonality is that of the mode shapes. According to this latter type, physical coordinates that correspond to different modes should give orthogonal vectors. This type of orthogonality is enforced as an inductive bias in the current training scheme.

The mappings learnt, are established to be $1-1$ functions. However, in order to be considered a modal decomposition, a mapping should also be able to enforce some of the orthogonalities mentioned previously. Statistical independence, as will be discussed later, is enforced by predefining the modal coordinates as Gaussian white-noise vectors. As far as the orthogonality of the mode shapes is concerned, it is enforced by training the algorithm so that samples in the physical space that correspond to separate modal variables, are orthogonal in terms of their inner product.

In order to enforce such an orthogonality, an assembly of neural networks using one of the two generators of the cycleGAN is defined, as shown in Figure 5.6. The architecture shown is the calculation, in a numerical manner, of the angle between two vectors tangent to the manifold of data and each one parallel to a different axis of the feature space.

The procedure followed starts by sampling a random point $\boldsymbol{u_1}$ in domain U. Afterwards, four points are defined in the neighbourhood of $\boldsymbol{u_1}$. Two by adding and subtracting a small quantity ϵ to one of the coordinates of the point ($\boldsymbol{u_{1a}^{+}} = \boldsymbol{u_1} + \{0, 0, ..., \epsilon, 0..., 0\}$ and $\boldsymbol{u_{1a}^{-}} = \boldsymbol{u_1} - \{0, 0, ..., \epsilon, 0..., 0\}$) and another two defined similarly for a different coordinate ($\boldsymbol{u_{1b}^{+}} = \boldsymbol{u_1} + \{0, 0, ..., 0, \epsilon, 0..., 0\}$ and $\boldsymbol{u_{1b}^{-}} = \boldsymbol{u_1} - \{0, 0, ..., 0, \epsilon, 0..., 0\}$). The two pair of

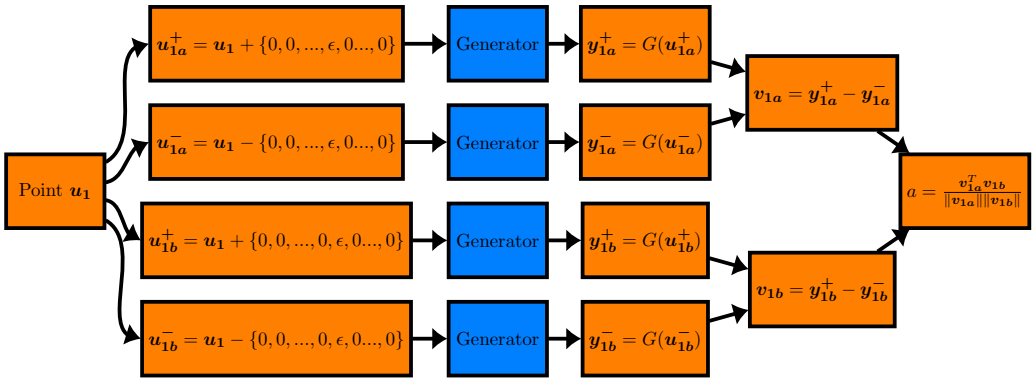


Figure 5.6: Orthogonality enforcement assembly using the generator that maps samples from the source Cartesian space to the target manifold.

points are subtracted and two vectors which are orthogonal are defined, which are the gradients along the two axes (in geometrical terms $\frac{\partial}{\partial u_i}|_{\boldsymbol{u_1}}$)).

The four points are subsequently transformed by the generator onto domain Y, resulting in points $\boldsymbol{y_{1a}^+} = G(\boldsymbol{u_{1a}^+})$, $\boldsymbol{y_{1a}^-} = G(\boldsymbol{u_{1a}^-})$, $\boldsymbol{y_{1b}^+} = G(\boldsymbol{u_{1b}^+})$ and $\boldsymbol{y_{1b}^-} = G(\boldsymbol{u_{1b}^-})$. From the four newly-defined points, the vectors $\boldsymbol{v_{1a}}$ and $\boldsymbol{v_{1b}}$ are defined as $\boldsymbol{v_{1a}} = \boldsymbol{y_{1a}^+} - \boldsymbol{y_{1a}^-}$ and $\boldsymbol{v_{1b}} = \boldsymbol{y_{1b}^+} - \boldsymbol{y_{1b}^-}$. Finally, the inner product of the two vectors is calculated and divided by the product of their norms, yielding the cosine $a = \frac{\boldsymbol{v_{1a}^T v_{1b}}}{\|\boldsymbol{v_{1a}}\|\|\boldsymbol{v_{1b}}\|}$, of the angle of the two vectors.

Considering domain U to be the modal space and domain Y to be the physical space, the procedure calculates the angle between two vectors (each one is parallel to a different axis in the modal space), after transforming them into the physical space. By setting the target value of the calculated cosine equal to zero and training using back-propagation, the mapping that the generator provides is forced to locally maintain the orthogonality. Such a preservation is described in Figure 5.7. Mappings such as the one sought here, which maintain the orthogonality locally are called *conformal* or *angle-preserving*.

5.3 Application of cycleGAN in nonlinear modal analysis

Following a similar machine learning framework as in [114], the cycleGAN algorithm, together with enforcement of the orthogonality described in the previous section is used to perform modal analysis for nonlinear structures. The statistical independence in [114] was

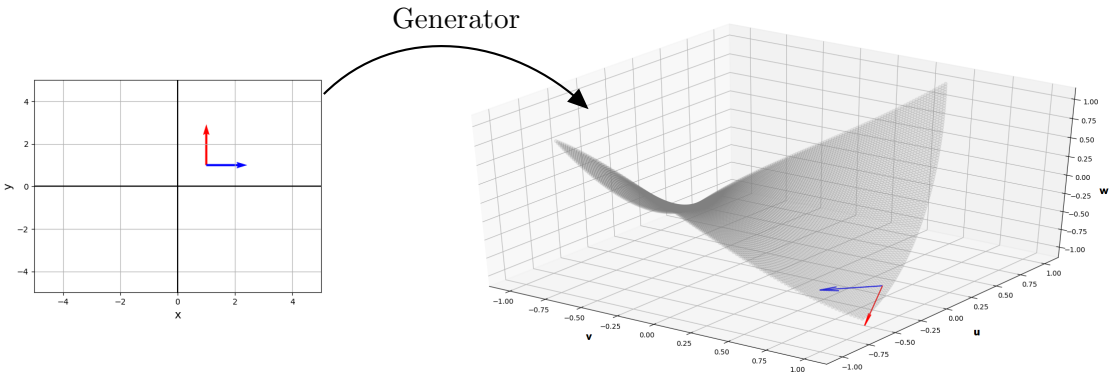


Figure 5.7: Preservation of orthogonality of vectors by the mapping of the generator from the source Cartesian space (x, y coordinate system) to the target manifold (u, v, w coordinate system).

enforced using a genetic algorithm [117], whose cost function included correlation terms in order to maximise statistical independence between modal coordinates. More specifically, the decomposition proposed for a three-degree of freedom system was given by,

$$\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix} + \begin{bmatrix} b_{11} & \dots & b_{19} \\ b_{21} & \dots & b_{29} \\ b_{31} & \dots & b_{39} \end{bmatrix} \begin{Bmatrix} y_1^3 \\ y_1^2 y_2 \\ y_1^2 y_3 \\ y_2^3 \\ y_2^2 y_1 \\ y_2^2 y_3 \\ y_3^2 \\ y_3^2 y_1 \\ y_3^2 y_2 \end{Bmatrix} \tag{5.16}$$

where $\{y_1, y_2, y_3\}^T$ are the physical coordinates, $\{u_1, u_2, u_3\}^T$ are the modal coordinates and $a_{ij}, i, j \in \{1, 2, 3\}$ and $b_{ij}, i \in \{1, 2, 3\}, j \in \{1, 2..., 9\}$ are trainable parameters in order to provide the decomposition. The objective function whose minimisation was attempted

was,

$$\begin{aligned} J = &|\{A_1\}\{A_2\}| + |\{A_1\}\{A_3\}| + |\{A_2\}\{A_3\}| \\ &+ \mathrm{Cor}(u_1, u_2) + \mathrm{Cor}(u_1, u_3) + \mathrm{Cor}(u_2, u_3) \\ &+ \mathrm{Cor}(u_1^3, u_2) + \mathrm{Cor}(u_1^3, u_3) + \mathrm{Cor}(u_2^3, u_1) \\ &+ \mathrm{Cor}(u_2^3, u_3) + \mathrm{Cor}(u_3^3, u_1) + \mathrm{Cor}(u_3^3, u_2) \end{aligned} \tag{5.17}$$

where $A_i = \{a_{1i}, a_{2i}, a_{3i}\}$. One can see that the proposed decomposition is a truncated polynomial, restricting its potential to provide proper decoupling into independent modes as the number of degrees of freedom increased.

As already mentioned, the objective function of equation (5.17) aims at defining a decomposition into modal coordinates whose correlation is as close to zero as possible. Using the cycleGAN algorithm proposed here, the statistical independence of the modal coordinates is attempted by predefining the modal coordinates to be Gaussian white noise vectors, i.e., defining the U domain to contain random vectors with mean value equal to zero and a correlation matrix equal to I_n, which is an n-dimensional identity matrix. Be defining the modal coordinates in such a way, the correlation terms of equation (5.17) are expected to be zero. Moreover, to further enforce statistical independence, PCA is applied to the physical coordinates before training; this way, existing initial correlations are minimised.

Further to minimising any initial linear correlations, PCA is expected to assist regarding a scaling problem that might occur. Different modes (ways) of motion participate at a different level to the whole movement of the structure. Different modal coordinates/axes, correspond to axes of the manifold of the collected displacement samples in the physical space. Since some modal coordinates have smaller magnitude, their corresponding axes on the aforementioned manifold have smaller length than others. The difference in the length of these axes may lead to difficulties of the algorithm to match the modal axes to the ones of the physical space manifold. PCA, in combination with a scaling into the interval $[-1, 1]$, which the neural network training requires, can solve this problem.

Finally, the orthogonality assembly of Figure 5.6, with a target cosine value equal to zero, is used in training. The generator that performs mapping from the modal space to the physical space ($G_{U\to Y}$) is subjected to such a training scheme, forcing physical space displacement vectors that correspond to distinct modal coordinates to be orthogonal

in terms of their inner product. The major advantage of the cycleGAN compared to previous approaches, is that the decomposition has no restrictions, since neural networks can approximate any function [54]. In the next sections, examples of the application of the algorithm to simulated and experimental data are presented.

5.3.1 Simulated case studies

The first case study chosen to illustrate the potential of the algorithm is a simulated two-degree-of-freedom system with a cubic nonlinearity, as shown in Figure 5.8. The parameters of the model were $m = 1.0$, $c = 0.1$, $k = 10$ and $k_3 = 1500$, which are similar to the ones used in [114]. The excitation was a white noise with zero mean and standard deviation equal to 5.0, which was low-pass filtered onto the frequency interval $[0, 50]$ Hz. In every simulated experiment, two datasets including 100000 points each were recorded. The first was used in order to train the cycleGAN and to select the best model over the whole training history and the second dataset was used in order to test the algorithm and presents the results. The quality of the decomposition is tested in terms of mode decoupling in the *power spectral density* (PSD), functions of the modal coordinates. The PSDs are calculated using Welch's method [118].

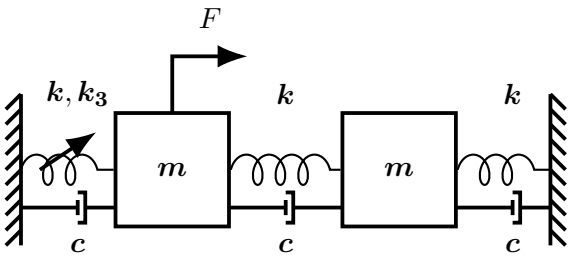


Figure 5.8: Two degree-of-freedom mass-spring system.

The architecture of the neural networks in every experiment was similar. They all had three layers and the size of the hidden layer was subjected to optimisation according to the performance of the model. The hidden layer sizes tested belonged to the set $\{50, 60, ...190, 200\}$ and for every size, 20 random initialisations and training of the neural networks were performed, since the algorithm is quite sensitive to the initial values of the trainable parameters. The activation functions were hyperbolic tangents except for the activation functions of the discriminators, which were sigmoid functions, since their

outputs should be in the interval $[0, 1]$, to represent the probability of a sample being real. The value of the parameter λ of equation (5.14) was set to 10, as suggested in [69], and the training algorithm used was the Adam algorithm [119].

Training followed a train-validate-test scheme, as in many machine learning applications. Training was performed as described. As far as validation is concerned, in order to pick the best model throughout the whole training history of the cycleGAN, a selection criterion is defined, which is an attempt to imitate the 'by eye' model selection performed in [114]. The criterion is described by,

$$\mathcal{L}_{\cos} = \sum_{i=1,j=i+1}^{n_{dof}} \frac{PSD_i \cdot PSD_j}{\|PSD_i\| \, \|PSD_j\|} \tag{5.18}$$

where PSD_i is the power spectral density of the i-th modal coordinate. By minimising $\mathcal{L}_{\cos}$, a decoupling of the modes is expected. Although the inner product may not go to zero, because of damping, which causes energy to be concentrated on frequencies around the natural frequencies, its minimisation should be able to provide a satisfactory decoupling of the modes.

5.3.1.1 Two-degree-of-freedom system

The algorithm was applied on data from the structure shown in Figure 5.8. The PSDs of the physical coordinates of the structure are shown in Figure 5.9. It is clear that the nonlinearity is affecting the PSD of the first degree of freedom, since a movement and a spread towards higher frequencies is observed, which is common in structures with such types of nonlinearities [19]. The natural frequencies of the underlying linear problem are equal to 0.5 Hz and 0.87 Hz, but because of the hardening behaviour of the nonlinear member, energy in the PSDs is observed in higher frequencies.

The procedure described before was followed and the model that yielded the best results in terms of the inner-product criterion of equation (5.18) had 100 units in its hidden layer. For comparison between the PSDs that the cycleGAN algorithm yielded and the ones that a common PCA decomposition yields, Figure 5.10 is shown. Linear PCA (top row)

is naturally unable to decouple the modes, since the simulated structure has a nonlinear member. In contrast the proposed algorithm provides a clear decoupling of the modes.

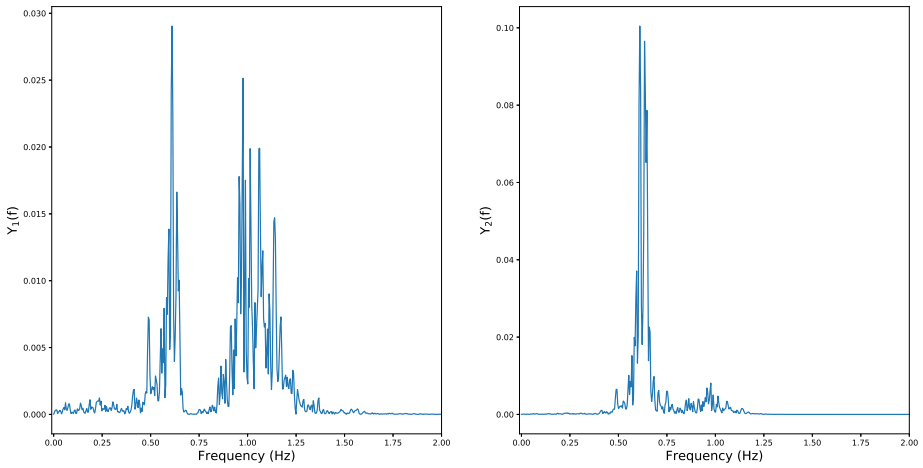


Figure 5.9: PSDs of two-degree-of-freedom structure; physical coordinates.

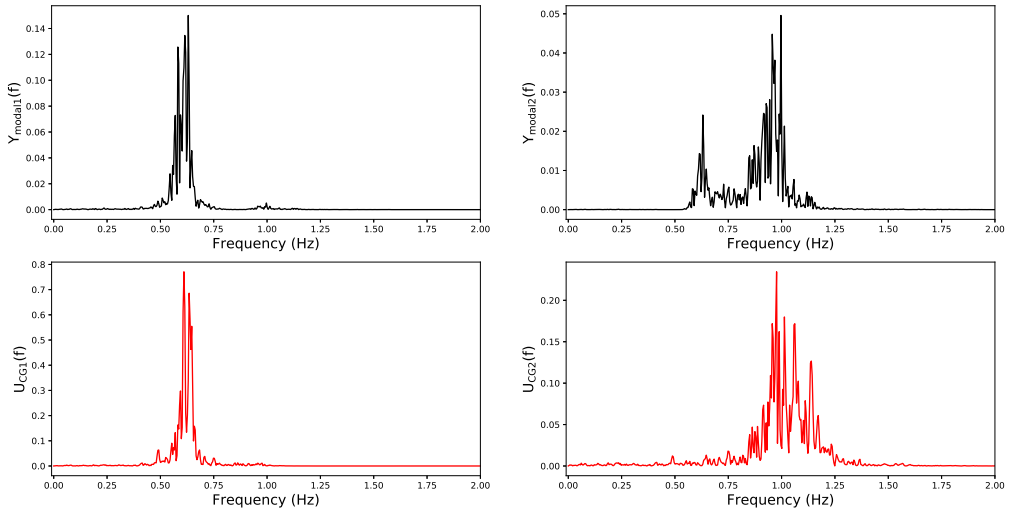


Figure 5.10: PSDs of two-degree-of-freedom structure, linear modal decomposition (top) and cycle-GAN decomposition selected via the inner-product criterion (bottom).

The ability of the algorithm to provide an inverse mapping for the modal decomposition is also tested. At every step of training, the generator that maps the modal coordinates back to the physical ones, is available by definition of the cycleGAN algorithm. To test its accuracy, the modal coordinates of the testing dataset are calculated and then they

are mapped back to the estimates of their original values using the second generator, i.e. $\hat{y}_i = G_{U \to Y}(G(Y \to U))$, where $\hat{y}_i$ is the estimate of the reconstruction.

The accuracy of the inverse mapping (or *superposition of the modes*, as it is often referred to), is tested according to the *normalised mean-square error* (NMSE) between the real and estimated values of the physical coordinates. The NMSE error of the reconstruction is given by,

$$NMSE = \frac{100}{N\sigma_y^2} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{5.19}$$

where y_i is the real value of the displacement of the ith sample, N is the total number of samples and σ_y is the standard deviation of the samples dataset. The NMSE is a proper metric of accuracy for regression problems. It provides an unbiased way of evaluating the results. NMSE values close to 100% indicate a model that is always providing predictions close to the mean value of the data, values less than 5% indicate a well-fitted model and values lower than 1% indicate an excellent-fitted model.

A part of the reconstruction results for the two-degree-of-freedom system is shown in Figure 5.11. The results seem very good, and this is confirmed by the value of the NMSE, which was 0.46% for this case.

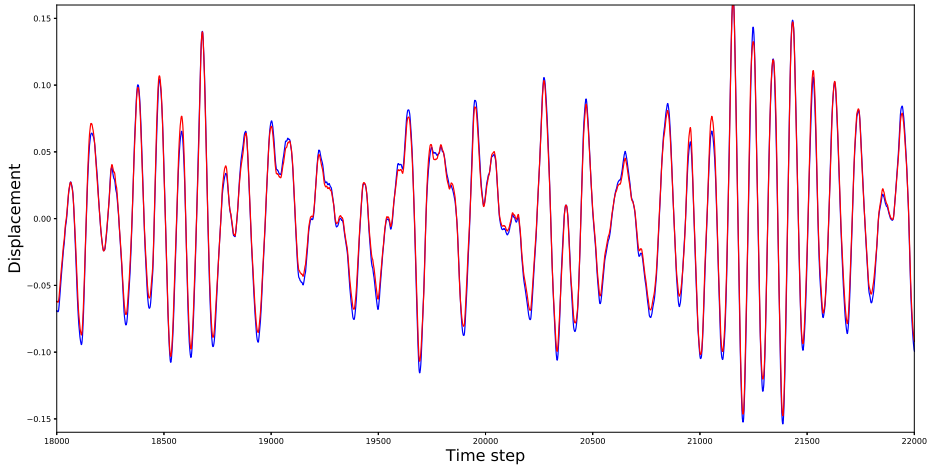


Figure 5.11: Superposition/inverse modal transformation for the two-degree-of-freedom system (red) and original displacements (blue).

5.3.1.2 Three-degree-of-freedom system

The algorithm was tested on systems with higher numbers of degrees-of-freedom. The same system parameters were used as in the two-degree-of-freedom system of Figure 5.8, but with an extra mass. The PSDs of the physical coordinates in this case, are shown in Figure 5.12. The spread because of the nonlinearity is also visible, since the natural frequencies of the underlying linear system are 0.39 Hz, 0.71 Hz and 0.93 Hz. The model that performed best in this case, had 110 neurons in its hidden layer and the resulting PSDs are shown in Figure 5.13, again in comparison with the PSDs from a PCA decomposition. The results are satisfactory, since, in every PSD of the 'modal' coordinates, only one peak is dominant and only small effects of other modes are present.

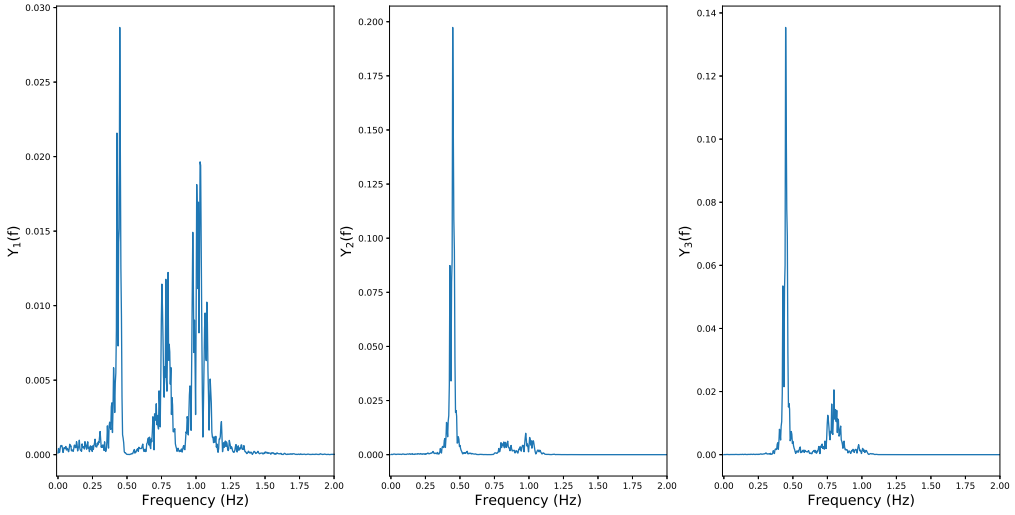


Figure 5.12: PSDs of three-degree-of-freedom structure; physical coordinates.

The superposition of the modes provided by the inverse mapping is also tested in terms of the NMSE error. Some results of the superposition are shown in Figure 5.14 and seem to be quite accurate. The accuracy is confirmed by the value of the NMSE which in this case was equal to 1.92%.

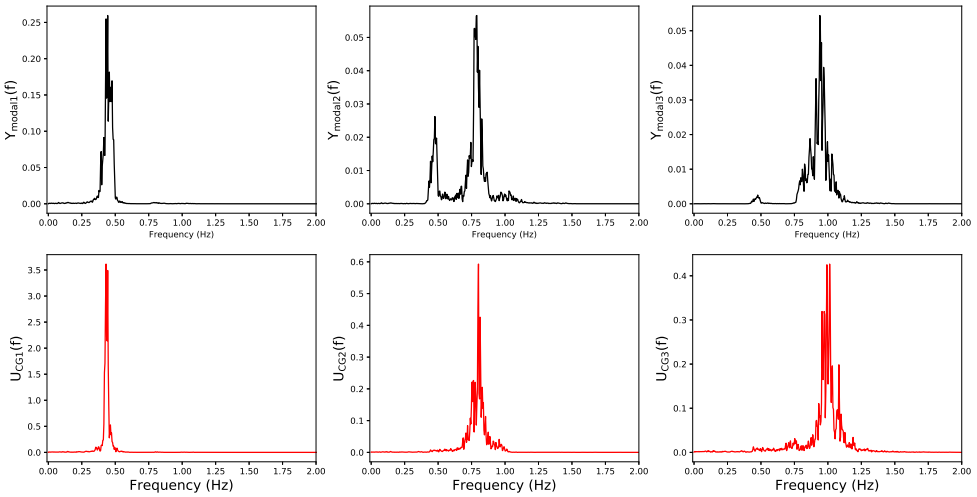


Figure 5.13: PSDs of three-degree-of-freedom structure, linear modal decomposition (top) and cycle-GAN decomposition selected via the inner-product criterion (bottom).

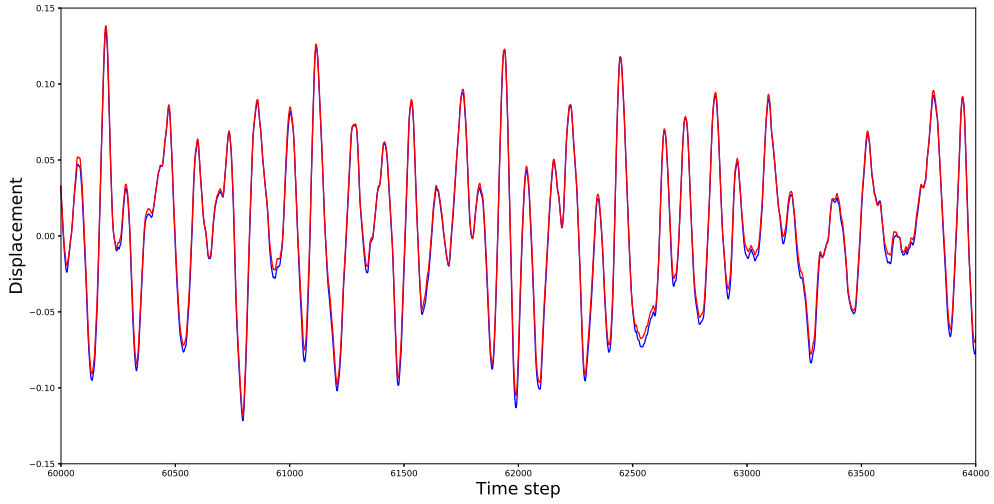


Figure 5.14: Superposition/inverse modal transformation for the three-degree-of-freedom system(red) and original displacements (blue).

5.3.1.3 Four-degree-of-freedom system

A four-degree-of-freedom system was also tested, a case study that was not studied in [114, 120], since the results on the three-degree-of-freedom system were not as good in that case as here. The new system has the same parameters as the two systems studied

before, except for its cubic nonlinearity parameter, which in this case was set to $k_3 = 3000$ so that strongly-nonlinear behaviour is observed. In this case the model that had the best performance was a neural network with 100 neurons in its hidden layer. The PSDs of the physical coordinates are shown in Figure 5.15. The shift of natural frequencies is visible as the natural frequencies of the underlying linear system are 0.31 Hz, 0.59 Hz, 0.81 Hz and 0.96 Hz. In Figure 5.16, the decomposition provided by linear PCA is shown, and in Figure 5.17, the decomposition provided by the cycleGAN algorithm is presented. The PCA decomposition does not yield PSDs with specific dominant peaks, as shown in the third and fourth PSD. However, the proposed algorithm seems to have only one dominant peak per modal coordinate PSD.

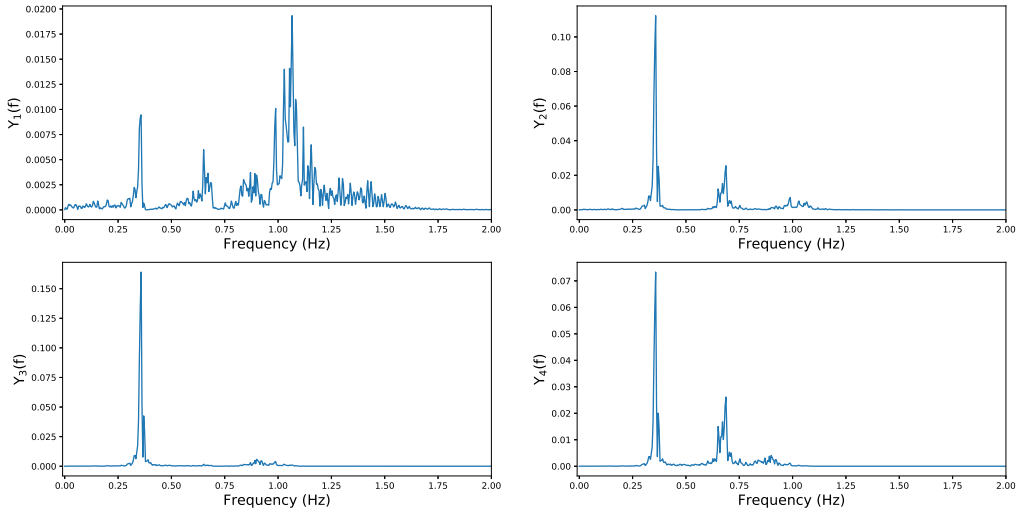


Figure 5.15: PSDs of four-degree-of-freedom structure; physical coordinates.

5.3.2 Experimental case studies

The algorithm was also tested on experimental data acquired from the set-up shown in Figure 5.18. The structure was excited by random noise applied to its base. The experiments were performed for 17 different states of the structure shown in Table 5.1. The states that are of interest in the current work are states $10 - 14$, where a nonlinearity is introduced to the structure. The nonlinear element is shown in Figure 5.18; it is a bumper and a column between the second and the third floor of the structure with a varying initial

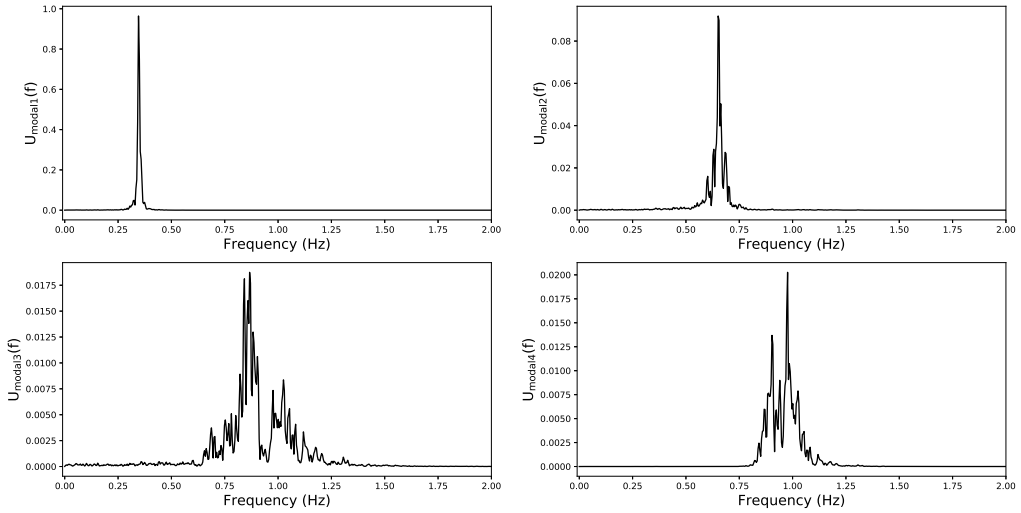


Figure 5.16: PSDs of four-degree-of-freedom structure, linear modal decomposition coordinates.

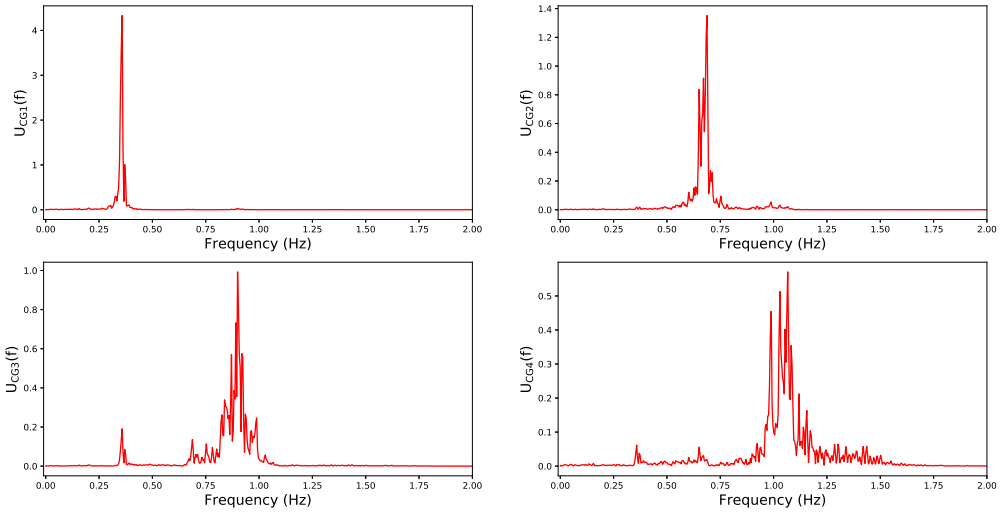


Figure 5.17: PSDs of four-degree-of-freedom structure selected via the inner-product criterion, cycle-GAN latent variables.

gap between them. The element is essentially a bilinear stiffness element which might also be introducing sudden energy dissipation to the structure. For the current analysis, two of the aforementioned states are considered. The first is State 12, which corresponds to an initial gap equal to 0.13mm and the second is State 14, which corresponds to a gap of 0.05mm. The second state is expected to have more intense nonlinearity and both cases

have harsh nonlinearities compared to the smooth nonlinearities presented in the simulated cases. In the experimental case studies, the acceleration of the base is considered to be the excitation of the three-floor building. Such an approach could be the case when one analyses such a structure for an earthquake case study.

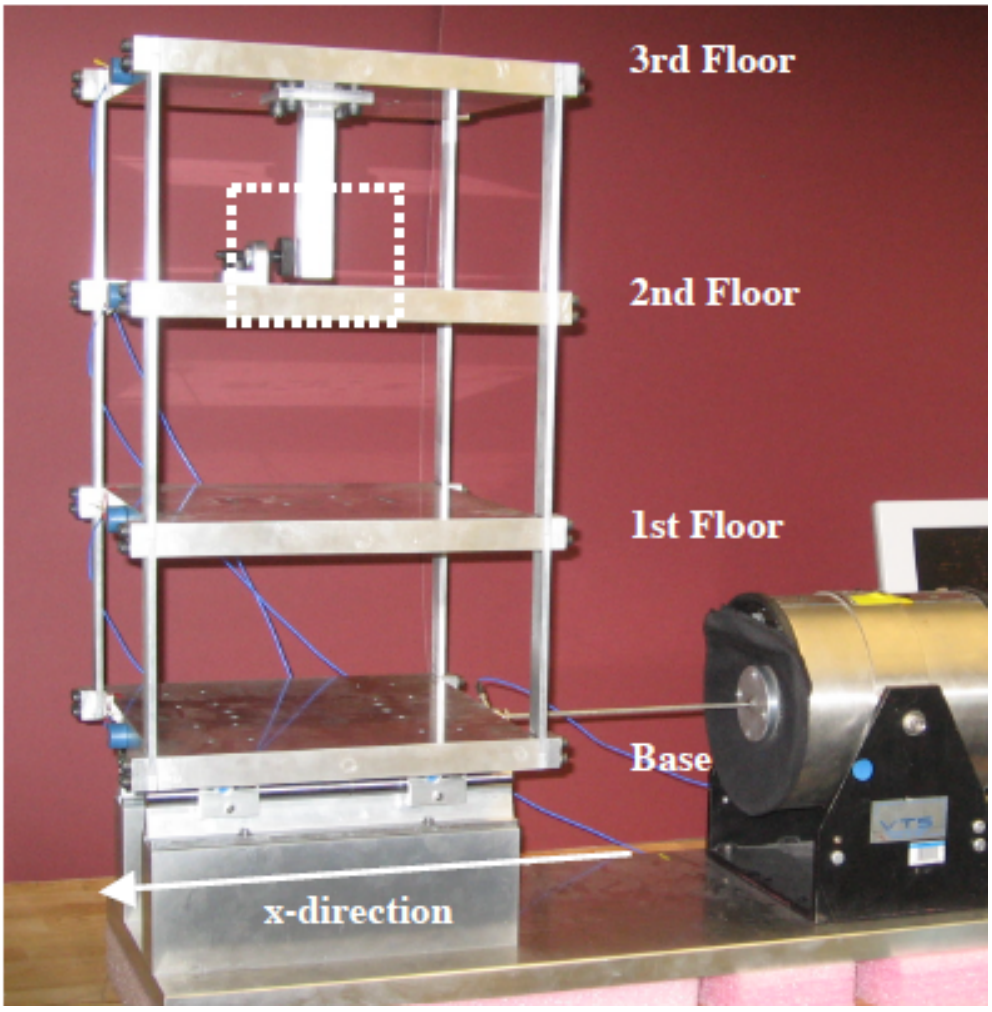


Figure 5.18: Experimental set-up of three-floor and the bumper nonlinearity between the second and third floor shown in the dashed box [121].

5.3.2.1 Experimental case study: State 12

The first experimental case study is that of State 12 of the experimental structure. The procedure followed is exactly as before, with the only exception that for every state, 50 experiments were performed. A sample PSD of one experiment for State 12 is shown in Figure 5.19.

Label	**State Condition**	**Description**
State #1	Undamaged	Baseline condition
State #2	Undamaged	Added mass (1.2 kg) at the base
State #3	Undamaged	Added mass (1.2 kg) on the 1^{st} floor
State #4	Undamaged	Stiffness reduction in base front column
State #5	Undamaged	Stiffness reduction in base front and rear column
State #6	Undamaged	Stiffness reduction in 1^{st} floor front column
State #7	Undamaged	Stiffness reduction in 1^{st} floor front and rear column
State #8	Undamaged	Stiffness reduction in 2^{nd} floor front column
State #9	Undamaged	Stiffness reduction in 2^{nd} floor front and rear column
State #10	Damaged	Gap (0.20 mm)
State #11	Damaged	Gap (0.15 mm)
State #12	Damaged	Gap (0.13 mm)
State #13	Damaged	Gap (0.10 mm)
State #14	Damaged	Gap (0.05 mm)
State #15	Damaged	Gap (0.20 mm) and mass (1.2 kg) at the base
State #16	Damaged	Gap (0.20 mm) and mass (1.2 kg) at the 1^{st} floor
State #17	Damaged	Gap (0.10 mm) and mass (1.2 kg) at the 1^{st} floor

Table 5.1: Description of different states of the experimental set-up [121].

The cycleGAN was trained using the accelerations from all 50 experiments. The size of the hidden layer that yielded the best results was 100 neurons. The achieved decomposition is shown in the bottom plots of Figure 5.20. Compared to the top plots of the same figure, which correspond to the decomposition provided by PCA, it is clear that the current algorithm performs better, since PCA does not achieve decoupling of the modes, while the cycleGAN has achieved single dominant peaks in every PSD.

As far as the (nonlinear) superposition of the modes is concerned, the inverse mapping was evaluated as before. Part of the reconstruction is shown in Figure 5.21. The quality of the reconstruction seems very good and this is confirmed by the NMSE value which is 1.23%.

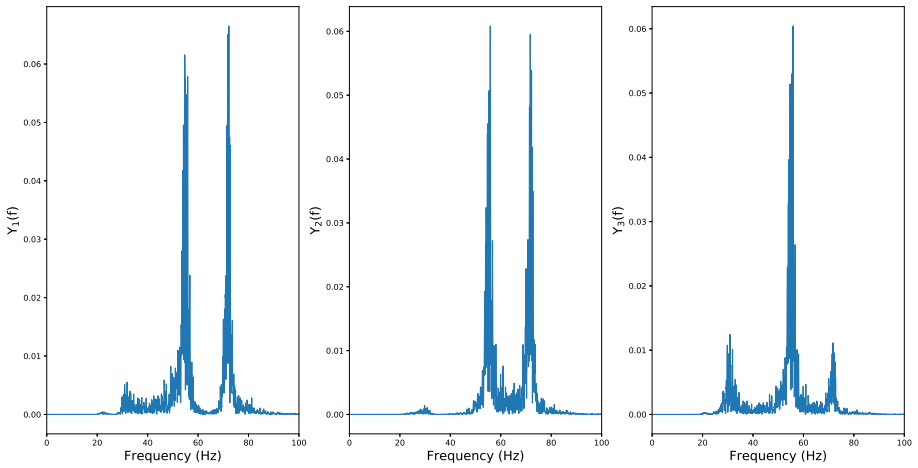


Figure 5.19: PSDs samples of three-floor experimental structure, physical coordinates of State 12.

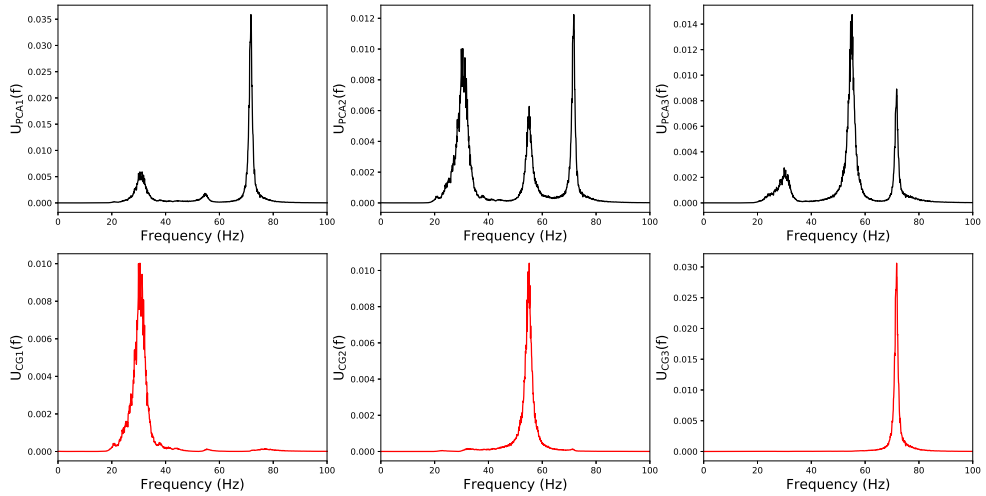


Figure 5.20: PSDs of three-floor experimental structure, PCA decomposition (top) and cycle GAN decomposition selected via the inner-product criterion (bottom); State 12.

5.3.2.2 Experimental case study: State 14

The same procedure was followed for State 14. The average PSDs of the physical coordinates of all 50 experiments are shown in Figure 5.22. The generator which yielded the best results had 100 nodes in its hidden layer. In Figure 5.23 the decomposition provided

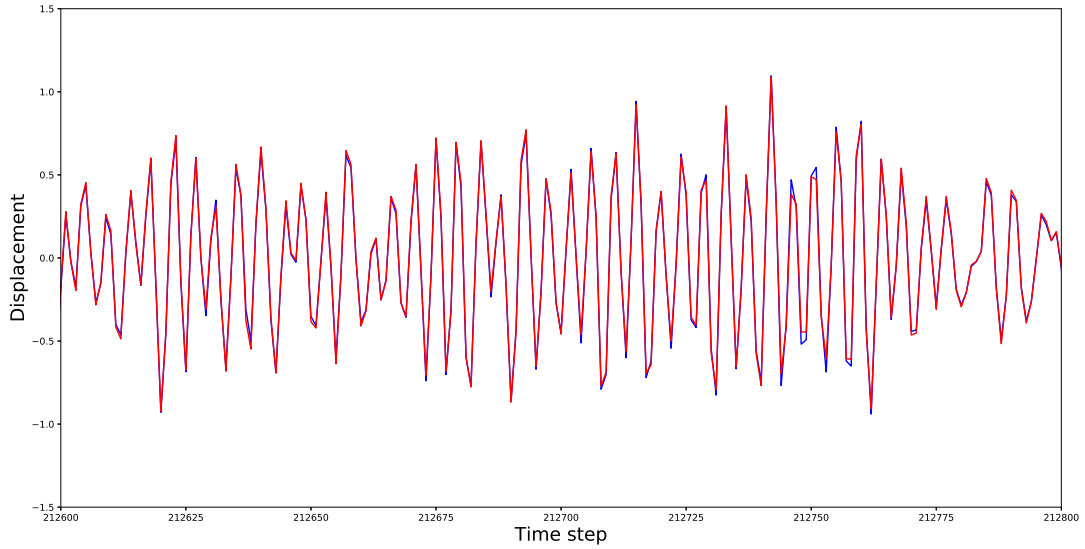


Figure 5.21: Superposition/inverse modal transformation for the experimental system(red) and original displacements (blue), State 12.

by PCA is shown in the top-row plots, and it is clear that it does not achieve any modal decoupling. The cycleGAN, on the other hand, as shown by the PSDs in the bottom-row plots of Figure 5.23 achieves better results, since each modal coordinate PSD has clearly one dominant peak.

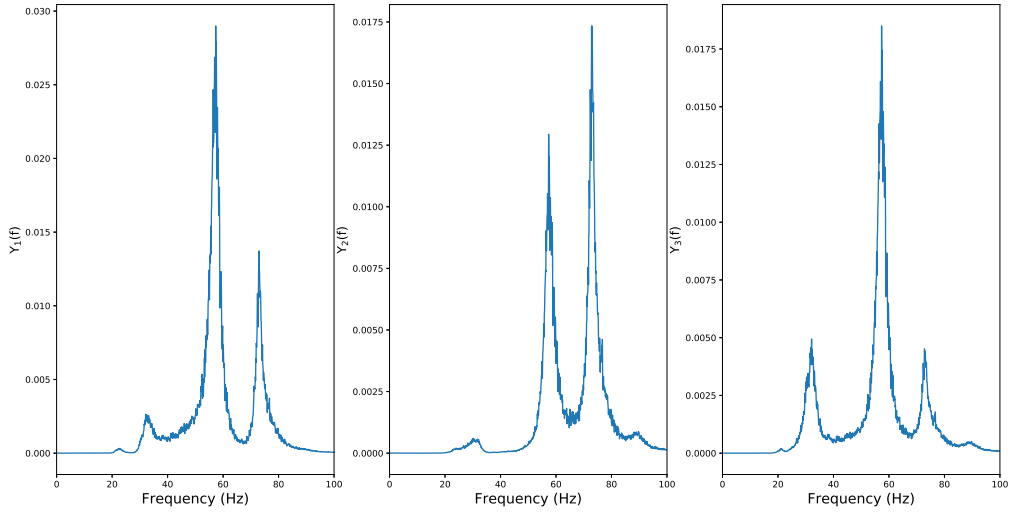


Figure 5.22: Average PSDs of three-floor experimental structure, physical coordinates of State 14.

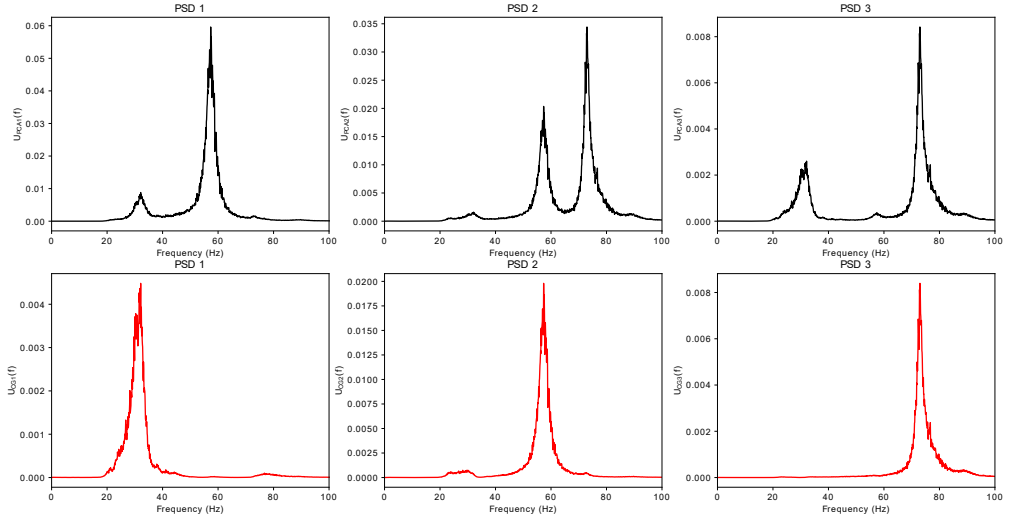


Figure 5.23: PSDs of three-floor experimental structure, PCA decomposition (top) and cycle GAN decomposition selected via the inner-product criterion (bottom); State 14.

The superposition mapping was tested also in this case. Part of the results are presented in Figure 5.24. The quality of the inverse mapping this time, seems to be lower and this is confirmed by the NMSE value which was 10.93%. Clearly the harsh nonlinearity has affected the quality of the inverse mapping.

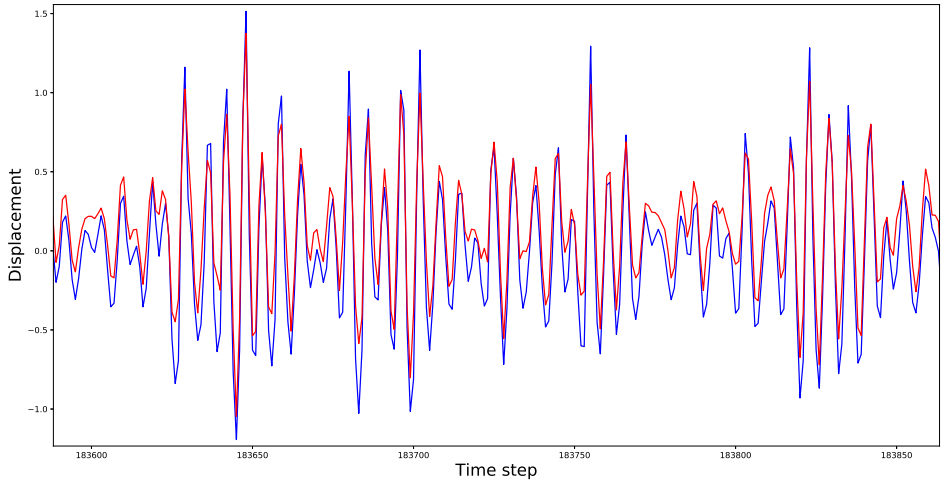


Figure 5.24: Superposition/inverse modal transformation for the experimental system(red) and original displacements (blue), State 14.

5.3.3 Modal correlation study

A major part of modal analysis is the statistical independence of the modal coordinates. Statistically-independent modal coordinates mean that the value of one coordinate cannot be estimated from the value of the others. The proposed algorithm attempts to achieve such independence by predefining the modal coordinates as $U \sim \mathcal{N}(\boldsymbol{\mu}, I_n)$. In the current paragraph, the performance of the algorithm in maintaining the independence of the modal coordinates is studied. The algorithm has to balance three types of losses (adversarial, reconstruction and orthogonality), which may lead to overlooking one of them, and if that one is the adversarial loss, then the modal coordinates may be associated.

The correlation of the modes is evaluated according to two correlation metrics, a linear and a nonlinear one. The linear metric is Pearson's correlation coefficient, which is widely used in statistical analysis. Pearson's correlation coefficient for a pair of random variables (X, Y) is given by,

$$\rho_{X,Y} = \frac{\mathbb{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \tag{5.20}$$

where μ_X, μ_Y are the mean values of X and Y respectively, σ_X, σ_Y the standard deviations of X and Y and $\mathbb{E}[\ ]$ is the expected value of the quantity in brackets. This correlation coefficient has values in the range $[-1, 1]$. Values of the metric close to zero mean that the variables X and Y are uncorrelated, while higher values indicate linear correlation of the two variables. In the current work, only the magnitude of the coefficient is of interest and therefore its absolute value is used.

The linear correlation is expected to be more easily minimised, since PCA is performed on the samples before applying the algorithm. Therefore, a second metric is also considered, which takes into account nonlinear correlations. The correlation metric is the *distance correlation* metric [122].

The distance correlation between two random variables (X, Y) is calculated by initially defining matrices A and B as follows; elements $\alpha_{j,k}$ and $\beta_{j,k}$ need to be defined by,

$$\begin{aligned} \alpha_{j,k} &= \|x_j - x_k\|, \qquad k, j = 1, 2, ..., n \\ \beta_{j,k} &= \|y_j - y_k\|, \qquad k, j = 1, 2, ..., n \end{aligned} \tag{5.21}$$

where n is the number of observations. Having defined the elements, A and B are given by,

$$\begin{aligned} A_{j,k} &= \alpha_{j,k} - \bar{\alpha}_{j\cdot} - \bar{\alpha}_{\cdot k} + \bar{\alpha}_{\cdot\cdot}, \qquad k,j = 1,2,...,n \\ B_{j,k} &= \beta_{j,k} - \bar{\beta}_{j\cdot} - \bar{\beta}_{\cdot k} + \bar{\beta}_{\cdot\cdot}, \qquad k,j = 1,2,...,n \end{aligned} \tag{5.22}$$

where $\bar{\alpha}_{\cdot k}$ is the mean value of the kth column, $\bar{\alpha}_{j\cdot}$ the mean of the jth row and $\bar{\alpha}_{\cdot\cdot}$ the mean value of all α elements. Finally, the *distance covariance* is calculated by,

$$dCov^2(X,Y) = \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{j,k}B_{j,k} \tag{5.23}$$

and the distance correlation $dCor$ between X and Y is given by,

$$dCor(X,Y) = \frac{dCov(X,Y)}{\sqrt{dVar(X)dVar(Y)}} \tag{5.24}$$

The value of the distance correlation is within the range $[0,1]$ and, as with Pearson's correlation, values closer to zero indicate lower correlation between the variables, while higher values indicate higher dependence. Next, the two correlation metrics are calculated and presented between the inferred modal coordinates of the previously presented case studies.

5.3.3.1 Two-degree-of-freedom simulated system

The correlation metrics between the modal coordinates for the two-degree-of-freedom simulated system are shown in Figure 5.25. The diagonal has values of 1 which is expected since the correlation of the modal variables with themselves is expected to be the maximum. The values of the coefficients between two modal coordinates are quite low and the value of the nonlinear correlation coefficient is higher, since it takes into account both linear and nonlinear dependencies.

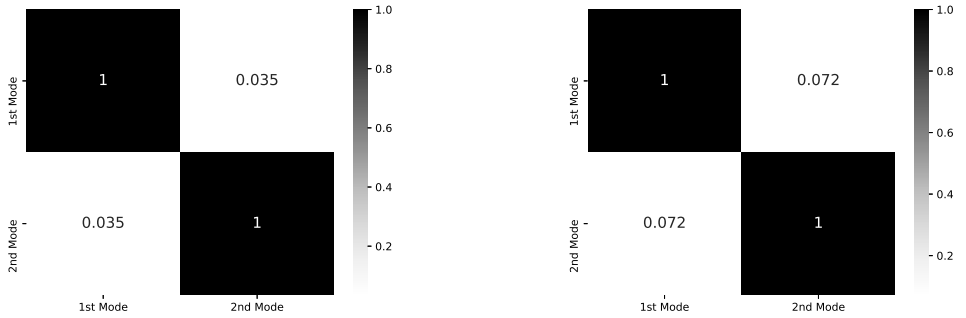


Figure 5.25: Pearson's linear correlation coefficient (left) and distance correlation coefficient (right) of the modal decomposition computed for the two-degree-of-freedom system.

5.3.3.2 Three-degree-of-freedom simulated system

Similarly, the coefficients of the modal coordinates of the three-degree-of-freedom system are shown in Figure 5.26. Again the correlation coefficients between the modal coordinates are quite low.

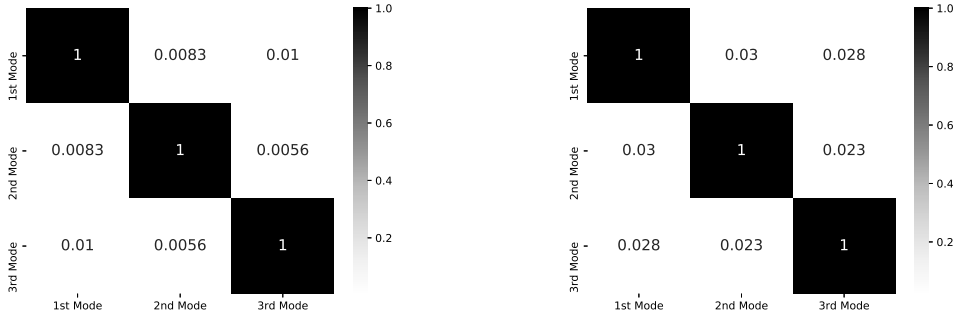


Figure 5.26: Pearson's linear correlation coefficient (left) and distance correlation coefficient (right) of the modal decomposition computed for the three-degree-of-freedom system.

5.3.3.3 Four-degree-of-freedom simulated system

The correlations for the four-degree-of-freedom system are shown in Figure 5.27. As before, the correlation coefficients are quite low.

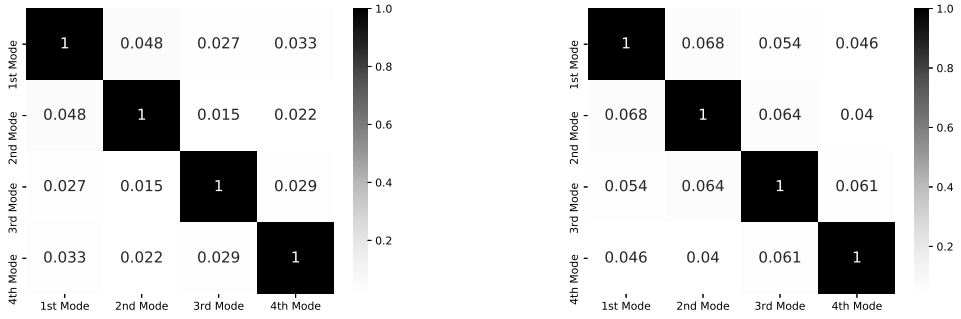


Figure 5.27: Pearson's linear correlation coefficient (left) and distance correlation coefficient (right) of the modal decomposition computed for the four-degree-of-freedom system.

5.3.3.4 Experimental system: State 12

For the case of the 12^{th} state of the experimental set-up, the correlation coefficients are shown in Figure 5.28. The values reveal that the modal coordinates seem quite uncorrelated.

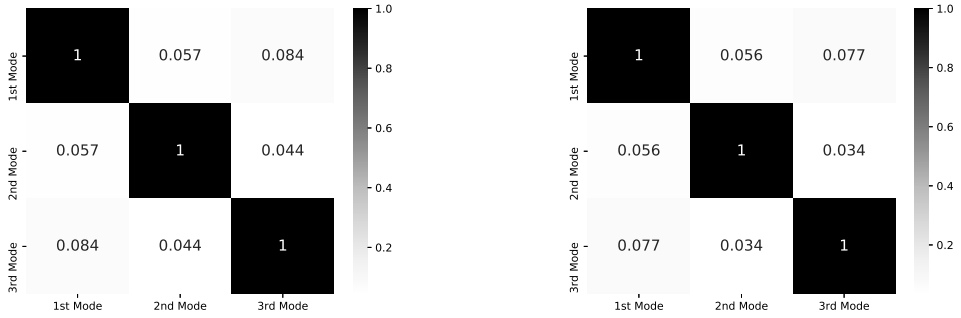


Figure 5.28: Pearson's linear correlation coefficient (left) and distance correlation coefficient (right) of the modal decomposition computed for the experimental system.

5.3.3.5 Experimental system: State 14

In the case of the 14^{th} state, the correlation coefficient values are shown in Figure 5.29. This time, the correlation coefficients between the first and the second modal coordinates seem to be higher than before.

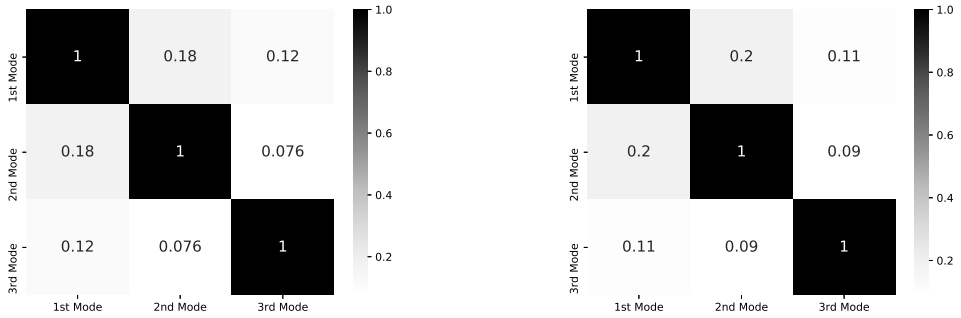


Figure 5.29: Pearson's linear correlation coefficient (left) and distance correlation coefficient (right) of the modal decomposition computed for the experimental system.

5.4 Summary

The results of this chapter reveal that the proposed algorithm is able to efficiently decompose the movement of the structures with nonlinearity. The decomposition, as far as the PSDs are concerned, offers modal coordinates whose PSDs have a single dominant mode. Moreover, the model, during training the forward transformation from the physical to the modal coordinates, is also trained to provide the inverse transformation, which in most cases was very accurate. Finally, the statistical independence of the modes is, in most cases, achieved. The very harsh nonlinearity in one case appears to create issues, both in the accuracy of the inverse transformation and in the achieved statistical independence of the modes.

As a further step to the method, the velocities of the system could be included in the process of the decomposition. Including the velocities could account for nonlinearities of the damping of the system. Such a modification to the method may be closer to the original Shaw-Pierre concept ansatz [123], according to which, NNMs were considered as manifolds in the phase space of the structure. According to [123], if the movement of the structure was initialised on one of these manifolds, it would remain on the same manifold, which is considered an NNM.

Chapter 6

Feature spaces of structures and potential state manifolds

6.1 Fibre bundles for PBSHM

As mentioned before, PBSHM is the subdiscipline of SHM that deals with performing inferences within a population of structures. Each structure might behave different according to its connectivity, dimensions etc. In order to connect the different behaviours in a compact way, a mathematical object is needed. In the current work, *fibre bundles* were selected to be the object used to connect the different behaviour of structures within a diverse population. Fibre bundles are mathematical objects based on the idea of *differential manifolds* [124]. Some terminology about manifolds and vector spaces is given in Appendix 9.2; they are tightly connected with the ideas of *gauge field theories* [125, 126]. Although they do not often appear in SHM literature, they are commonly encountered in nonlinear dynamics [127–130]. In the current work they shall be combined with machine learning techniques to perform inferences, which is also not something new, as algorithms called *manifold learning* [131, 132] have been studied before.

The basic elements that compose a fibre bundle are two, a *base manifold* M and a *total manifold* E. The two manifolds are connected via a mapping π, which is a projection

from E to M. Using such a mapping, movement on one manifold causes movement on the other. Since π is a mapping that projects points from E to M, it follows that the dimension of E is *higher or equal to the dimension* of M. The inverse mapping of π (π^{-1}) might thus be a multi-valued mapping defined on every point of M. The definition of π^{-1} leads to definition of the objects F_x defined by,

$$F_x = \pi^{-1}(x) \in E, \pi : E \to M \tag{6.1}$$

where F_x is called the *fibre above x*. A first requirement for fibre bundles is that all such fibres are homeomorphic to each other.

Given the description of a fibre bundle so far, the set of objects needed to define it are $\{M, E, \pi, F\}$. An example of such a fibre bundle is shown in Figure 6.1. The simplest fibre bundle one can define is by selecting a fibre F to be a specific manifold, define the base manifold M and then define the total space E as the Cartesian product $E = F \times M$. However, this may not serve any useful purpose. In order for the fibre bundle to be more versatile, E should have a more general structure. By working locally, one can allow the properties described before to stand for local domains U of M. Then it stands that $\pi^{-1}(U) = U \times F$ locally. This assumption means that the homemorphism of the fibres does not need to extend globally on all of M but only locally. Following such a scheme, the whole bundle is defined as the set of objects $\{M, \{U_i\}, E, \{\varphi_i\}, \pi, F\}$ where $\{U_i\}$ is a coordinate atlas on M and for each U_i on M, φ_i is a homeomorphism (Figure 6.2),

$$\varphi_i : \pi^{-1}(U_i) \longrightarrow U_i \times F \tag{6.2}$$

a property which is also called *local triviality*.

Fibre bundles can be used in order to formulate the procedure of performing inference within a population of structures. The first step in order to do so, is to define the base manifold to be the 'space' of structures S. Consequently, each structure s_i is assigned a fibre F_i which is selected to describe the *potential states* that the structure may be observed in. The states may describe the structure in terms of characteristics, such as natural frequencies, frequency response functions (FRFs), transmissibilities etc. In any

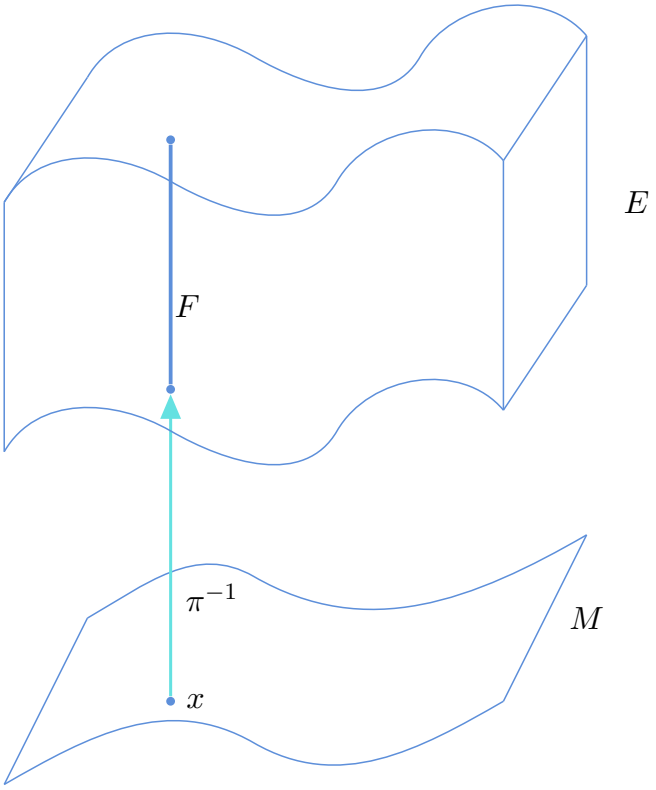


Figure 6.1: Schematic of basic objects in a fibre bundle.

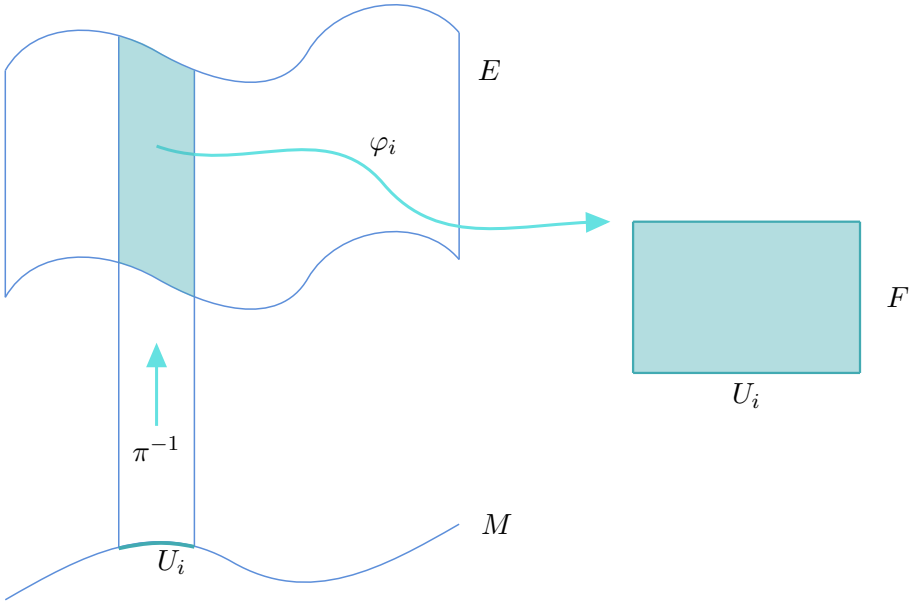


Figure 6.2: Local triviality property in a fibre bundle.

case, since the whole construct of the fibre manifold is going to be used for structural health monitoring, these attributes should be damage sensitive.

The fibres can be collected and glued together in order to define the total space together with the base manifold. The procedure of making inference within the population is reduced to moving from structure to structure in the base manifold and inferring the corresponding fibre for every point/structure. Essentially, the function that needs to be defined is the π^{-1} function which would map a structure to its potential states, i.e. $\pi^{-1}(s_i) = F_i$.

The fibres can describe the structure in its healthy as well as in its damaged conditions. Since data are acquired from existing structures from different states, data from their healthy conditions are quite often available, but damaged conditions could also be available. The fibres, as mentioned, describe the potential states of the structures; therefore, their damaged states can also be included in the fibres, resulting in a more holistic approach to inferring potential states of the structure of interest.

If only healthy states are included in the fibres, the mapping π^{-1} provides a way to infer the healthy characteristics of structures within the population from available data of different structures. This strategy is useful, especially when one wants to perform novelty detection in order to define whether damage is present in a structure [76, 78, 79, 133]. Novelty detection essentially learns a distribution or a manifold from the available data, and indicates anything that does not belong to the learnt element, as an outlier. If trained with healthy data, such methods indicate changes as damage. In any case, in order to train such a model, one should have available data from the healthy state of some structure, which is not always possible, since it might be difficult to acquire them, or the structure may be relatively new. Following a PBSHM strategy in order to infer the healthy state data of such a structure from other similar structures could be a way to deal with such issues.

A more holistic approach would be to include all available data in the fibres and infer the potential states for a range of damage cases. In this case, another useful element of the fibre bundles is defined - this is a *cross section*. The cross section of a bundle is defined by simply collecting a point or a set of points from every fibre. Then, by gluing these submanifolds together, a subset of the total space is created which is called a cross section. The points of the cross section may have something in common; for example, they may refer to the healthy states of the structure, thus defining the *normal condition cross*

section, a schematic representation of which is shown in Figure 6.3. The cross sections may also refer to different types of damage.

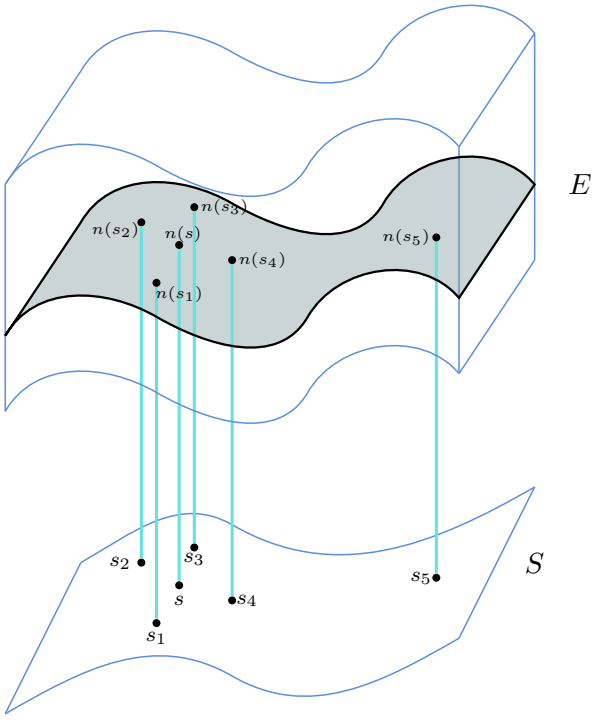


Figure 6.3: Normal section $n(s)$ of a feature bundle over a space of structures S. It is assumed that normal condition data are not known for the point s. Interpolation/transfer of the normal condition from any neighbouring points s_i should only be considered if they are sufficiently close to s in some metric on S. In this diagrammatic example, s_5 might be considered 'too far away'; for transfer.

If one wants to move further up Rytter's hierarchy [75], and perform damage classification [81], data from damaged states of the structure should be used. In that case, the fibres should include data from damaged states of the structure. Including such data in the potential states, one is transferring the knowledge of how damage mechanisms affect the structures within the population. Following such a framework, multiple mappings π_i^{-1} can be defined in order to map a structure to different damage states i. One can see that the fibre bundle formulation is quite versatile and allows one to decide which characteristics to infer within the population, and under which circumstances.

An example to illustrate the above strategies within a fibre bundle framework, could be to define such a bundle for a population of cantilevers. The cantilevers have a rectangular cross section and their geometric characteristics are their length l, width w and thickness t.

Moreover, they have some structural characteristics, which are their Young's modulus E, their Poisson's ration ρ and their density ρ. The base manifold S can be trivially defined to be equal to $\mathbb{R}^6_+$, since these are the potential values that these quantities might have.

A characteristic one might monitor in order to detect damage within such a population could be the set of the n first natural frequencies of the cantilevers. Consequently, each fibre includes sets of the n first natural frequencies of the cantilevers for different values of environmental conditions that might affect their values (temperature, humidity etc.). If one includes also damaged states within the formulation, then each fibre F_i would also include sets of natural frequencies recorded for damage-states of the cantilevers. The formulation of such a population-based SHM problem is schematically shown in Figure 6.4. By navigating in the space of cantilevers S, one can apply at any point the inverse transformation π^{-1}, and infer the potential states that a structure s can be observed in.

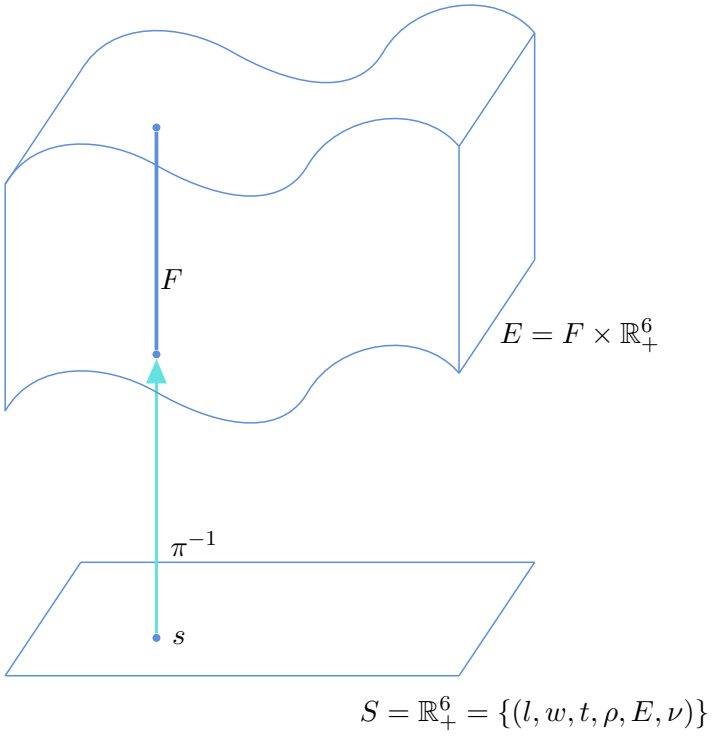


Figure 6.4: Collected feature spaces across a population of structures S considered as a vector bundle.

Even though the above formulation is quite appealing, its application has some obstacles that one has to overcome. A major problem is that structures within populations cannot be easily encoded into points in a multidimensional space, in order to allow the inference that

is to be performed. Structures are complicated objects and therefore a different approach should be followed. In the next sections, a way to avoid such problems is described, as is the algorithm used to perform the whole procedure in a totally data-driven manner.

6.2 Graph neural networks (GNNs)

Machine learning is focussed on defining relationships between quantities, via the use of data. The general scheme of a machine learning algorithm is to define sets of input and output quantities, and try to define the function that maps the former to the latter. Algorithms such as Gaussian processes [134], support vector machines (SVMs) [135] and neural networks [8, 9] search exclusively for a relationship between data. Lately, the need to define models whose functionality is more interpretable and controllable have led to the need for algorithms that are further informed by the structure of the data and the relationships that might exist between features of the data. This trend has been observed in social network modelling [136–138], biology [139], chemistry [140], medicine [141], etc. The new framework is focussed on improving existing methods by inducing some physics into training and inference, as well as using more diverse objects for inputs and outputs, other than simply vectors.

An algorithm that has been motivated by such problems is that of *graph neural networks* (GNN) [58]. A simple problem presented in the original work, with a view to exposing common problems of machine learning algorithms, is the calculation of the centre of mass of a planetary system. The inputs to the model are the coordinates of the n planets of the system; the output is the coordinates of the centre of mass. The problem seems trivial but a 'traditional' data-driven model, trained to perform the task, would fail in two cases. The first is if a permutation of the planets' masses was used as an input. If the indices of the planets are shuffled compared to the order they were given during training, the data-driven model would fail to predict the center of mass, since it cannot be informed somehow about the shuffle. The second way the model would fail, would be by considering a different number of planets. Traditional machine learning algorithms learn mappings from one space to another, where both spaces have fixed dimension.

Graph neural networks have been developed to solve such problems. The algorithm is based on using graphs as inputs and outputs to the inference procedure. The values of interest are assigned as features to the nodes and edges, and as global features to the graphs. As will be shown, the graphs used by the algorithm do not need to have the same size in order for the algorithm to be able to perform. Therefore, one of the two problems presented in the previous trivial example does not occur. Inference is based on information exchange between nodes, via the edges that connect them. The edges reveal relationships between the nodes and encode how these relationships affect the inference procedure.

The procedure of defining a graph of the data is equivalent to inducing existing knowledge about the physics of the underlying problem into the inference algorithm. Such a framework is often referred to as using *inductive biases* in the model. Such biases are common when one trains a machine learning algorithm which should favour specific types of solution to the problem, according to one's knowledge about the problem. A major type of inductive bias is the use of *regularisation* when one trains a neural network. Regularisation of the weights of the neural network penalises the loss function for higher values of the weights, favouring smaller values and therefore more smooth mappings, as an attempt to avoid overfitting. A second commonly-used inductive bias is in the use of convolutional neural networks [142, 143]. Convolutional neural networks use trainable filters which extract features from images, exploiting the fact that local substructures in images often define their characteristics, such as edges and shapes. The inductive bias induced in this case is that of the *locality* of the features.

The inductive bias in the case of graphs is inserted as connections between nodes of the graph. Similar models are the *probabilistic graphical models* (PGMs) [144] and *Markov processes*. In such models, the edges connect states of the system that affect each other. In the current work, the connections will not refer to states of the structures but will represent the connectivity between elements of the structure. Such connections are expected to encode the layout of the structure and allow the algorithm to learn how these connections affect the behaviour of the structures. Different patterns of connection between members encompass the knowledge that is needed to be transferred within the population of the structures.

As discussed earlier, the inputs and the outputs of the algorithm are graphs and more

specifically *attributed graphs*. An example of such a graph is shown in Figure 6.5. The graph has different elements, and each one has *attributes* in the form of vectors. As shown, the edges are directed, revealing the direction of the flow of information. In the most general case, all elements of the graph may have attributes, but, if it is convenient for the user, some class of elements may not have.

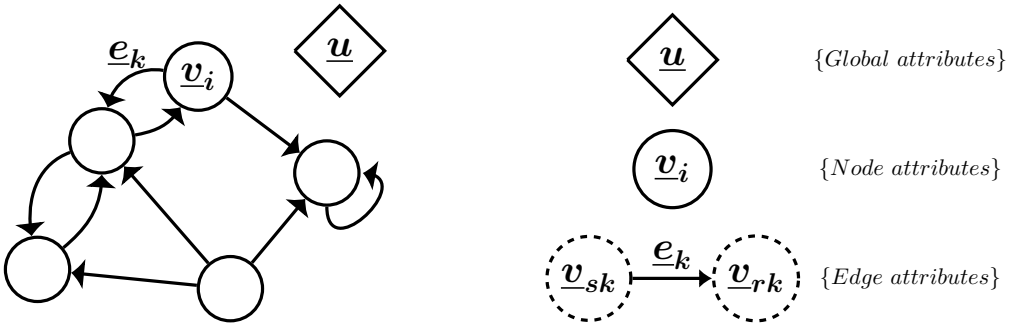


Figure 6.5: General graph architecture.

Graphs like the above form the inputs and the outputs of the algorithm. The procedure followed for the transition from one graph to another is called a *computational block* (CB). A CB comprises three operations:

1. the edge updates,
2. the node updates, and
3. the global attribute updates.

During each step, the values of the features of a different element of the graph are updated. A different number of CBs may be used sequentially in order to get the output graph with the required target values. The number of CBs is a hyperparameter of the model.

6.2.1 Edge update

The edge update is the first step of a CB. During this step, the values of the edge attributes of the graph are updated. A schematic representation of the updating is shown in Figure 6.6. The updated attributes $\underline{e}'_k$ of the kth edge are updated according to,

$$\underline{e}'_k = \phi^e(\underline{e}_k, \underline{v}_{s_k}, \underline{v}_{r_k}, \underline{u}) \tag{6.3}$$

where $\underline{e}_k$ are the values of the attributes of the kth edge before the updating, $\underline{v}_{s_k}$ are the attributes of the *sender node* on the tail of the kth edge, which is updated in the current step, $\underline{v}_{r_k}$ are the attributes of the receiver node and $\underline{u}$ are the global attributes of the graph. The function ϕ^e is a trainable function used in order to update the edge attributes.

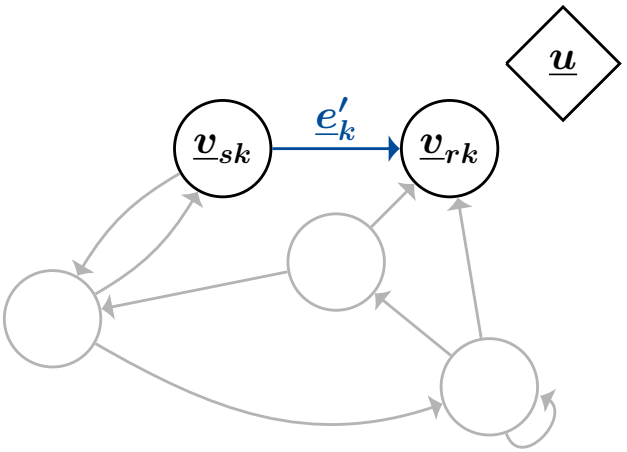


Figure 6.6: Edge update step.

It is clear that the procedure of updating the edge is local. This makes the algorithm invariant to permutations of the indices of the nodes and also ensures that only elements, which are connected to the element which is updated, affect the new values. The function ϕ^e plays a similar role to the convolutional kernel in a CNN.

6.2.2 Node update

The second step is to update the attributes of the nodes. The procedure is similar to before, and only elements in the neighbourhood of the node affect its updating procedure. The updating process is schematically shown in Figure 6.7. A major difference to the edge update is that the number of elements that affect the update of every node varies. A different number of edges points to every node and therefore, a way to aggregate the information is needed. To do so, an *aggregation function* $\rho^{e\to v}$ is defined, which is applied on the updated attributes of the set of edges E' pointing to the node, which is currently updated.

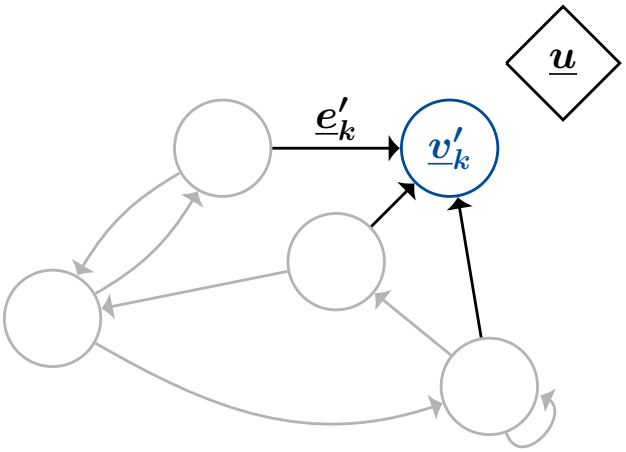


Figure 6.7: Node update.

Similarly to before, the updated values of the attributes of the kth node $\underline{v}'_k$ are given by,

$$\underline{v}'_k = \phi^v(\rho^{e\to v}(E'), \underline{v}_k, \underline{u}) \tag{6.4}$$

where ϕ^v is the updating function of the nodes, which is learnt by the data. The aggregation function $\rho^{e\to v}$ could be an averaging or summation function, and is selected by the user as a hyperparameter of the algorithm.

6.2.3 Global update

The final step of updating is to update the global attributes of the graph. This updating is illustrated in Figure 6.8. Every element of the graph participates in the updating, and two aggregative functions are needed $\rho^{e\to u}$ and $\rho^{v\to u}$; one for the updated values of the edges E' and one for the updated values of the nodes V'. The updated values of the global attributes $\underline{u}'$ are given by,

$$\underline{u}' = \phi^u(\rho^{e\to u}(E'), \rho^{v\to u}(V'), \underline{u}) \tag{6.5}$$

where the function ϕ^u, as in the case of ϕ^v and ϕ^e, is learnt from the data.

The updating steps concern only neighbourhoods of the elements that are updated. This is the reason that more than a single CBs might be needed to allow information to flow

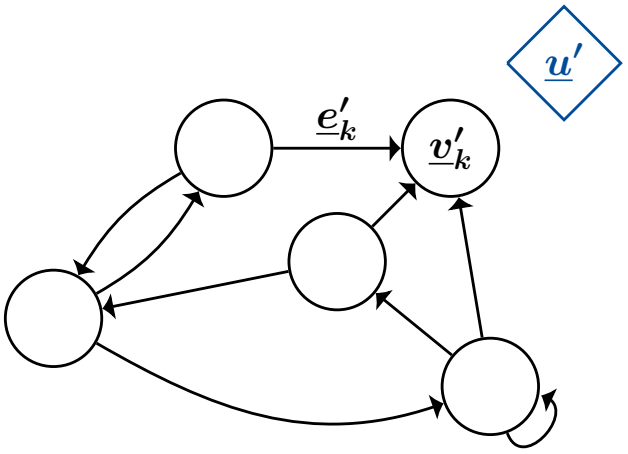


Figure 6.8: Global update.

between elements that are not close in the graph. Another way to allow such information flow would be to include nodes and edges in the wider neighbourhood of the under-update element to participate in the updating.

6.2.4 Training of graph neural networks

The three steps of the updating procedure presented are essentially the equivalent of a forward pass in a neural network. A complete computational block can be seen in Figure 6.9. In practice, all three ϕ functions are selected to be neural networks for them to be trainable. Considering that the whole procedure is differentiable, the error between some target values of the output attributes can be backpropagated to train the neural network functions, e.g. by using scaled conjugate gradients [8, 9] or the Adam training algorithm [119].

Although a full computational block is shown in Figure 6.9, the target values that one may be interested in may not be all the outputs (u', V', e'). If the quantity of interest is only one of them, then the computational block can be reduced, as shown in Figure 6.10, where only the node attributes are of interest. This way, the computational cost of the algorithm is reduced.

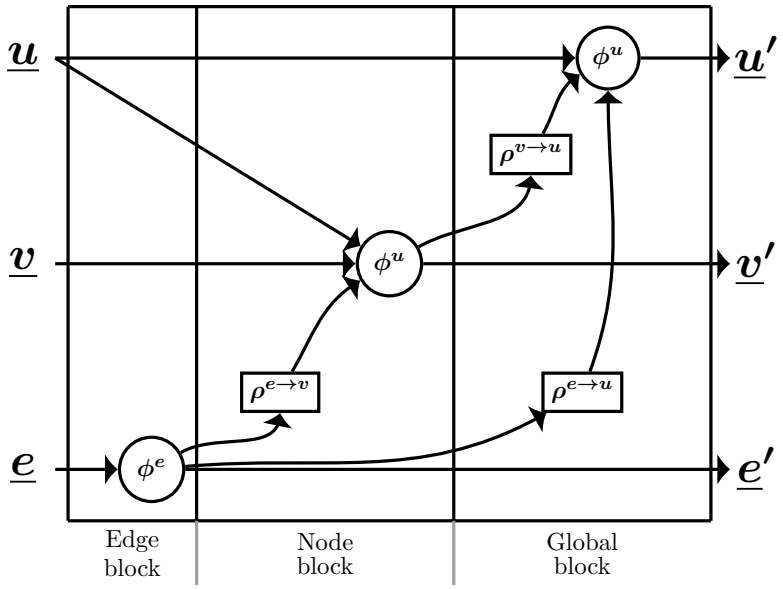


Figure 6.9: Full computation block (motivated by [58]).

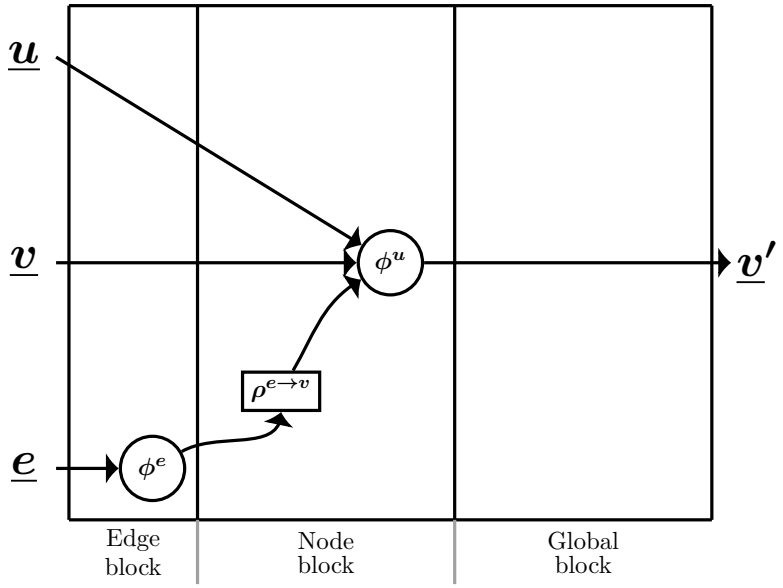


Figure 6.10: Reduced computation block.

6.3 Application of graph neural networks in PBSHM

6.3.1 Structures as graphs

The algorithm of GNNs seems to fit really nicely the inferences between a population of structures that are needed. As mentioned, structures are abstract objects that cannot

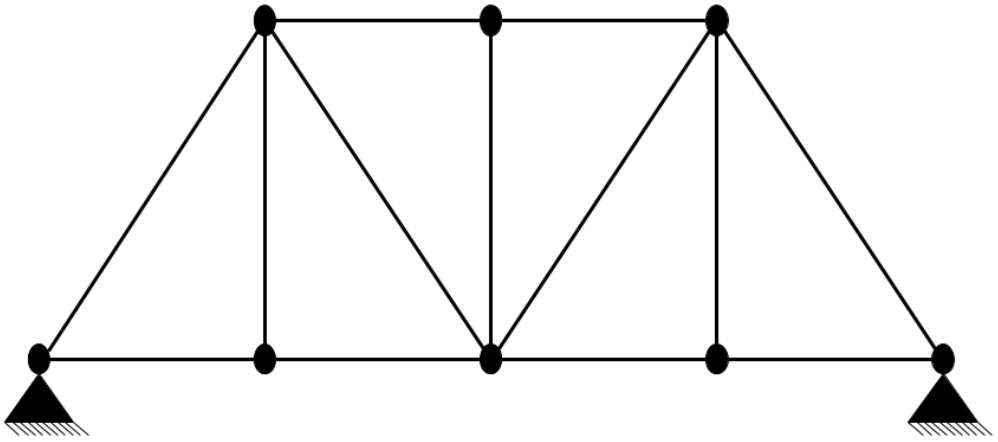
be straightforwardly transformed into feature vectors in order to use a machine learning algorithm to perform inference within a population. However, the GNN algorithm, being a non-Euclidean algorithm, offers an alternative to this approach. Combining the conversion of structures into graphs introduced in [91], with a GNN model, transferring knowledge within a population of structures becomes plausible. In order to evaluate such a hypothesis, a simulated population of trusses is considered here. Trusses were selected mainly because of their unambiguous transformation into attributed graphs.

Trusses are structures comprising rod elements, which are connected only at their edges, making them 'two force elements', i.e. loaded only in their axial direction. Even though they are quite simple structures, they are used in a wide range of constructions, such as roofs, antennae, cranes and bridges. A railway bridge and a simple planar truss model of it are shown in Figure 6.11 [1]. Trusses within the simulated population are created in a similar way to how the planar model was created.

The attributed graph models were formed in a reverse manner to that in [91], where the members of the structures are represented as nodes and the connections as edges of the graphs. The procedure here is the inverse, the members are the edges and the joints are the nodes. The attributes describe the structural characteristics of the truss. The node attributes are the x and y coordinates of the node, as well as binary variables encoding whether the node is fixed in the x or y direction. The attributes of the edges are selected to be binary variables that encode the different classes that each member belongs to. These different classes correspond to different Young's modulus E and different cross sectional area A. Moreover, they encode how different member classes are affected by environmental parameters, such as temperature. Moreover, it was found that, in order to facilitate training, the edge attributes should also include the length L of the member, which is needed to calculate its stiffness $K = \frac{EA}{L}$, as well as the sine and cosine of the member's angle with the horizontal axis. The categorical encoding of the members is a convenient way of avoiding defining their exact parameters, which is usually done in a laboratory test. The actual parameters change over time and via the exposure of the structure to the environment. Therefore, a categorical encoding rather than an exact definition of the material parameters is more feasible.

[1] The bridge image was taken from `https://en.wikipedia.org/wiki/Truss`: last accessed 11/11/20.

(a)

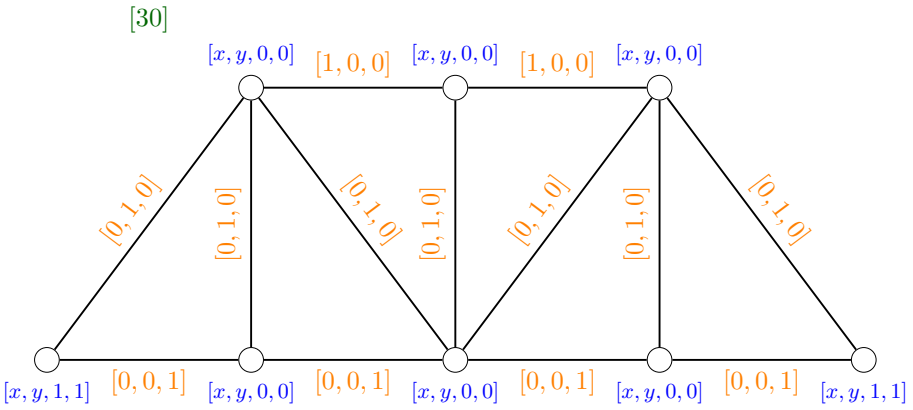


(b)

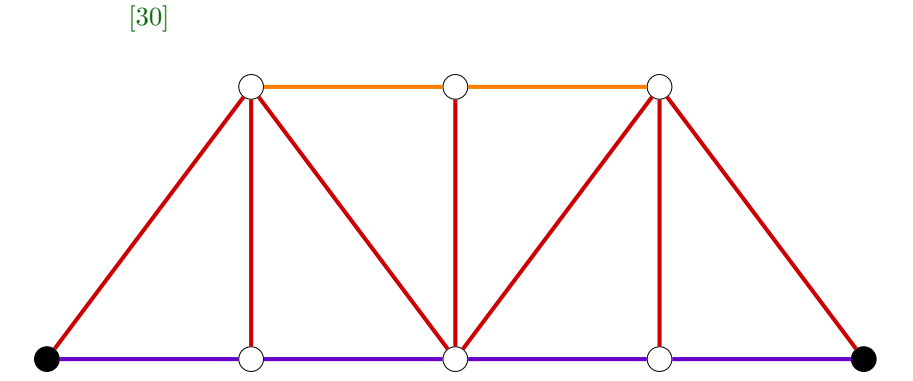
Figure 6.11: (a) Truss railway bridge, and (b) simple planar model.

Following the described transformation strategy of trusses into graphs, the truss from Figure 6.11 is encoded into the graphs shown in Figure 6.12. The members at the top of the structure are considered to belong to the first category, the members in the 'middle' in a second and the members at the bottom in a third, explaining their binary encoding at the top figure ($\{1, 0, 0\}$ for the first category, etc.). The nodes attributes include their x and y coordinates and binary variables representing their boundary conditions. All nodes are free to move except for the leftmost and rightmost nodes that are fixed in both directions. A global feature could be the temperature of the environment to which the

truss is exposed. In the example, the temperature is $30°C$



(a) Detailed attribute representation.



(b) Simplified 'categorical' representation.

Figure 6.12: Network representation of the bridge shown in Figure 6.11. (a) shows a detailed representation in terms or explicit node (blue) and edge (red) attributes; (b) shows a simplified representation indicating fixed (black) and pinned (white) nodes, with the three member types colour coded as 'top' (orange), 'middle' (red) and 'bottom' (blue).

6.3.2 Case studies

In order to test the method, a population of structures should be defined. The population here comprised two-dimensional trusses, by randomly sampling the number of nodes of each truss from the interval $[10, 40]$, and the coordinates x and y of every node from the interval $[0, 10]$; the trusses were created by applying Delaunay triangulation [145] on the nodes. The nodes were randomly fixed in no direction, x direction, y direction or both. In the cases that a temperature was used, the temperature was also randomly sampled from the interval $[20, 40]$. The value whose approximation is sought here, is the first natural frequency of the trusses (denoted ω) and, in order to create the datasets, the natural

frequencies were calculated by a simple eigenvalue decomposition of the stiffness matrices of the trusses. The three case studies considered are:

1. **Case Study One**: The population is created by trusses whose members all belong to the same member class (they all have the same EA quantity). In this case, the categorical encoding of the member types is not needed. The variation in stiffness of each members is exclusively due to variation in the length of the members. The population is quite homogeneous in terms of the structural members, but quite heterogeneous in terms of the topology of the structures.

2. **Case Study Two**: In this case study, temperature is introduced as an external parameter of the simulations. Temperature affects the stiffness of the members according to the relationship shown in Figure 6.13. In this case, as in the first case, a categorical encoding of the member classes is not needed.

3. **Case Study Three**: For the third case study, a second type of member is introduced in the population. Each truss member is randomly selected to belong to one of the two classes. A temperature variation is introduced as in the second case study and a second member type has a nonlinear relationship between its EA quantity and temperature, as shown in Figure 6.14 (the relationship is given by $EA = -13T^2 + 500T + 7200$, where T is the temperature).

6.3.2.1 Case study One

For each case study, three datasets were created, one for training the algorithm, one for validation/model selection and one for testing. All datasets included 16000 trusses; in Figure 6.15 some samples of the randomly created trusses are shown. In order to train the model, different GNN model architectures were tested and a cross-validation scheme was used to select the optimum. The experiments showed that three computational blocks yielded the best results

A major problem during the cross-validation procedure was the computational cost of the experiments. Training models with different numbers of hidden layers and nodes in them for a wide range of such parameters would be practically infeasible with conventional

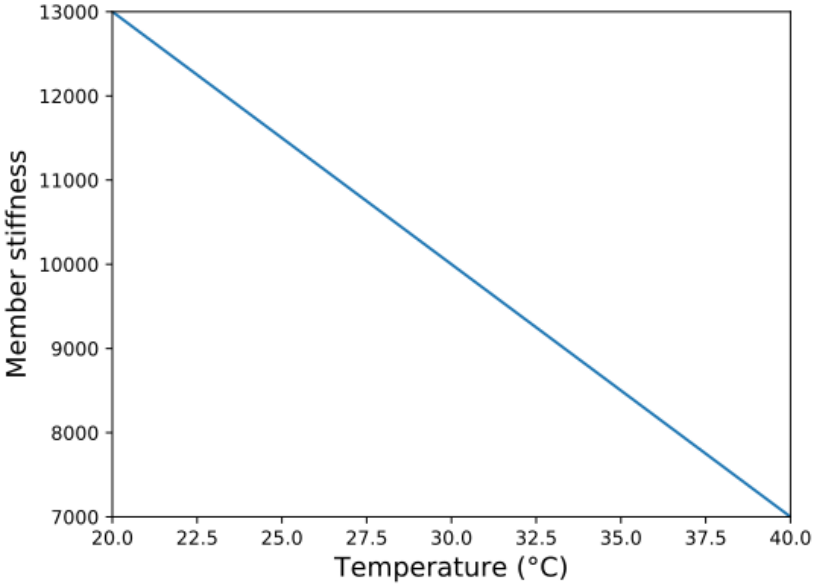


Figure 6.13: Linear relationship between temperature and EA of first type of members (Case Studies Two and Three). Note that it is clearly unrealistic to assume that the stiffness of a member would halve over a range of 20 degrees Celcius. However, the point of the exercise here is to demonstrate that the methodology works well, even in the presence of large confounding influences. For this reason, the influence of temperature is deliberately exaggerated.

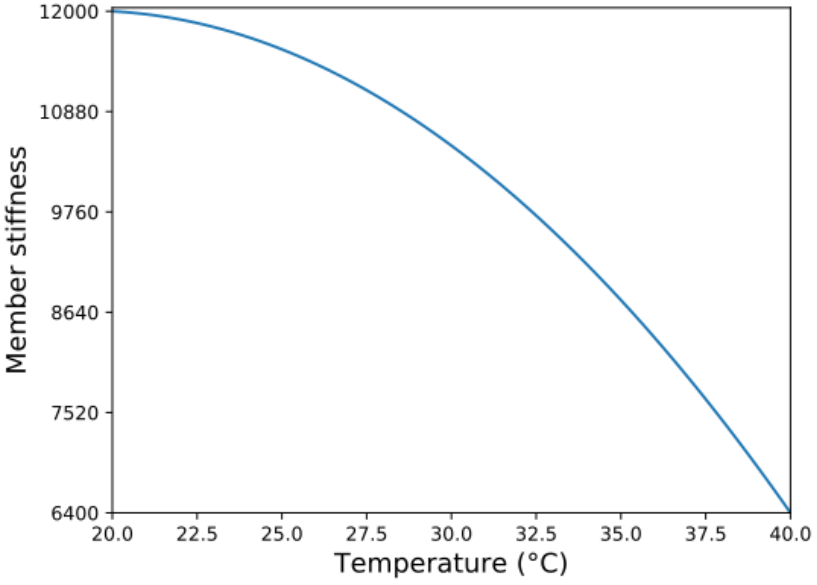
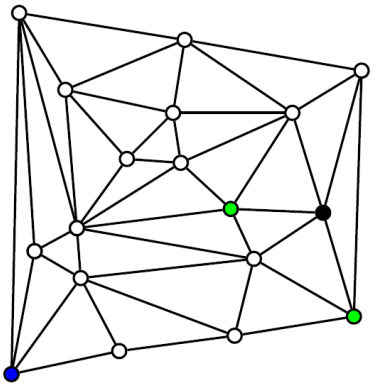
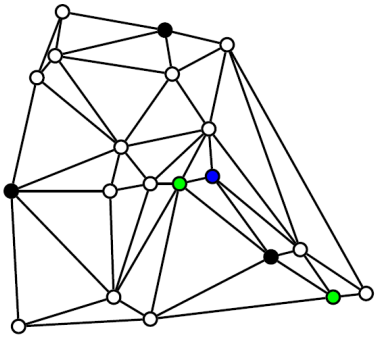


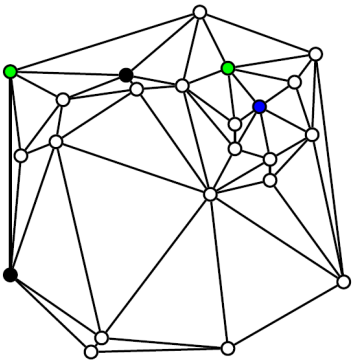
Figure 6.14: Nonlinear relationship between temperature and EA of second type of members (Case Study Three).



(a)



(b)



(c)

Figure 6.15: Three randomly-selected truss structures generated as part of the training data for the GNN algorithm in Case Study One. Black nodes represent fully fixed nodes, green nodes are fixed in the x direction, blue are fixed in the y direction and in white are free (pinned) nodes).

hardware. Therefore, the approach followed was closer to a 'hit and miss' strategy, where different parameters were tested in a limited range of parameters. The results do not necessarily represent the best possible performance, but as will be seen, they are acceptable to show that the algorithm is able to solve the problem in hand.

As mentioned, the quantity that was approximated was the first natural frequency of the trusses, which was set as the target for the global value of the output graph. Since the problem is a regression problem, a normalised mean square error is used, defined by,

$$NMSE = \frac{100}{N\sigma_{\omega}^2}\sum_{i=1}^{N}(\hat{\omega}_i - \omega_i)^2 \tag{6.6}$$

where ω_i is the target value for the natural frequency and $\hat{\omega}_i$ is the value estimated by the algorithm, σ_{ω}^2 is the variance of ω over the data set concerned. As already mentioned the NMSE is an unbiased way of evaluating the performance of a regression model.

The training was performed using an *averaging aggregative function* for the ρ functions of the algorithm. This approach was proposed in [58]. The sizes of the neural networks were tried incrementally and 10 random initialisation were performed for every architecture. It was observed that as the sizes of the neural networks of the first CB were increased, the sizes of the neural networks of the rest of the CBs needed to be increased for the algorithm to perform well. Therefore, sizes of 20 to 600 units were tested with an increment of 20 for the first CB and larger neural networks for the rest of the CBs. In Table 6.1 the architecture of the best performing model is shown. The numbers represent the number of nodes of the neural networks' layers. For example the array $3, 64, 32$ means a neural network with 3 nodes in its input layer, 64 nodes in its hidden layer and 32 nodes in the output layer. In some cases, the input size is fully defined by the number of features of the nodes (e.g the input features of the nodes are 4 as described before), or the edges, and by the connectivity of the GNN model; for example, the edge network of the third CB has an input layer of 290 nodes, which is the result of the concatenation of the updated edge attributes (50) coming from the edge NN of the second CB, the updated node attributes (2×70) (sender and receiver nodes) and the updated global attributes of the second CB (100). In each neural network, the activation function was selected to be a *rectified linear* or *ReLU* function [146]. The training history of the model is shown in Figure 6.16.

	First CB	Second CB	Third CB
Edge NN	3, 64, 32	132, 80, 50	290, 180, 80
Node NN	4, 72, 50	100, 100, 70	250, 300, 72
Global NN	-	120, 200, 100	252, 300, 72, 1

Table 6.1: Sizes of neural networks used in the GNN model with mean aggregative function: first case study.

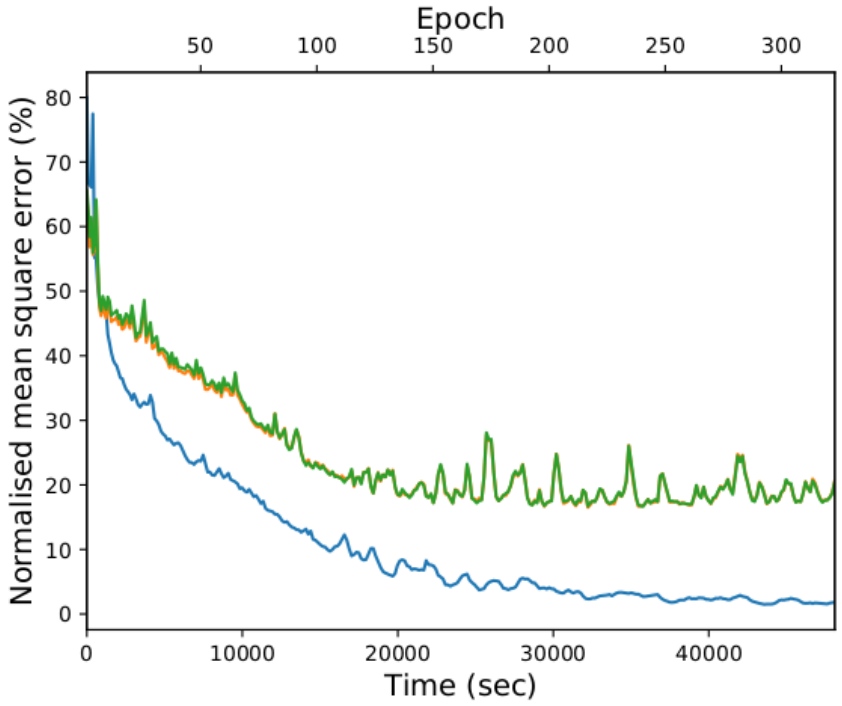


Figure 6.16: Training history of model using the summation aggregative function, training (blue), validation (orange) and testing (green) datasets: first case study.

Although the errors for the training dataset are quite low (1.55%), those on the validation and testing datasets are 17.16% and 17.20% respectively. A positive aspect of the figure, is that the errors of the validation and the testing datasets are quite similar, meaning that the algorithm is consistent in generalising. This consistency indicates also that the cross-validation procedure followed, worked as intended.

In order to reduce the error on the validation and training datasets, as well as the training time, a slightly different aggregative function is considered. A lot of information is potentially lost from the averaging (or summation) aggregation and therefore, an attempt is made to maintain more information about the quantities that are aggregated. To maintain more information, the aggregative function is selected to be both an averaging function and the calculation of the variance of the aggregated vectors. The results of these two functions is then concatenated into one vector, yielding an augmented aggregated vector. Following such a scheme, a more informative aggregative function is defined and the performance of the algorithm is expected to be better.

Following the same procedure as for the first case study, the architecture of the best performing model is shown in Table 6.2. Since the augmented aggregative functions are used, the size of the neural networks is obviously larger. The training history of the same model is shown in Figure 6.17.

	First CB	Second CB	Third CB
Edge NN	3, 64, 32	132, 80, 50	290, 180, 80
Node NN	4, 72, 50	150, 100, 70	330, 300, 72
Global NN	-	240, 200, 100	404, 450, 150, 1

Table 6.2: Sizes of neural networks used in the GNN model with the augmented aggregative function first case study.

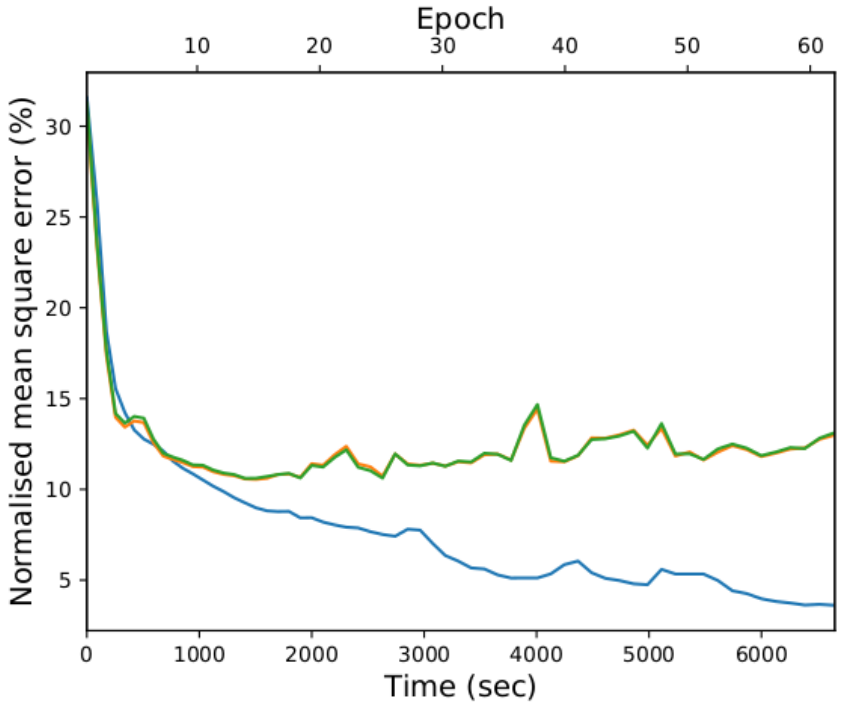


Figure 6.17: Training history of model using the augmented aggregative function, training (blue), validation (orange) and testing (green) datasets: first case study.

The minimum errors in this case were 10.54% and 10.61% on the validation and testing datasets. The errors of the training and testing datasets are again similar throughout the training history, and this time the minimum was achieved much faster than before, i.e. after 18 epochs (1500 seconds = 25 minutes). Overall the training history seems improved by the use of the augmented aggregation function.

6.3.2.2 Case study Two

For the second case study, a similar training procedure was followed. The architecture of the best performing model is shown in Table 6.3 and the corresponding training history in Figure 6.18. The minimum errors were 10.96% and 11.21% for the validation and testing datasets respectively. The results are considered satisfactory, since the algorithm performs well, even for different temperatures.

	First CB	Second CB	Third CB
Edge NN	3, 64, 32	167, 80, 50	220, 180, 80
Node NN	4, 72, 50	235, 100, 70	330, 300, 72
Global NN	1, 120, 85	325, 200, 100	404, 450, 150, 1

Table 6.3: Sizes of neural networks used in the GNN model with the augmented aggregative function: second case study.

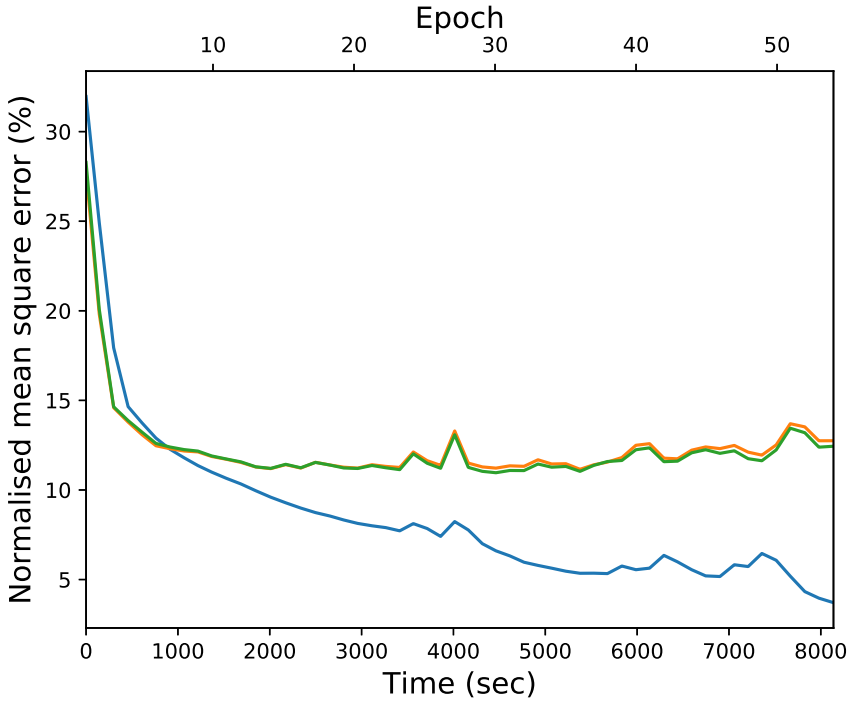


Figure 6.18: Training history of model using the augmented aggregative function, training (blue), validation (orange) and testing (green) datasets: second case study.

6.3.2.3 Case study Three

The same procedure was followed for the third case study, which is considered the most complicated. In Table 6.4, the architecture of the model that performed best is shown and in Figure 6.19 the corresponding training histories are presented. The validation and testing errors were 9.76% and 9.90% respectively. Given the complexity of the third case, the results are quite impressive.

	First CB	Second CB	Third CB
Edge NN	5, 100, 64	234, 100, 64	264, 250, 120
Node NN	4, 120, 85	298, 200, 100	440, 450, 150
Global NN	1, 120, 85	413, 200, 100	640, 450, 150, 1

Table 6.4: Sizes of neural networks used in the GNN model with the augmented aggregative function: third case study.

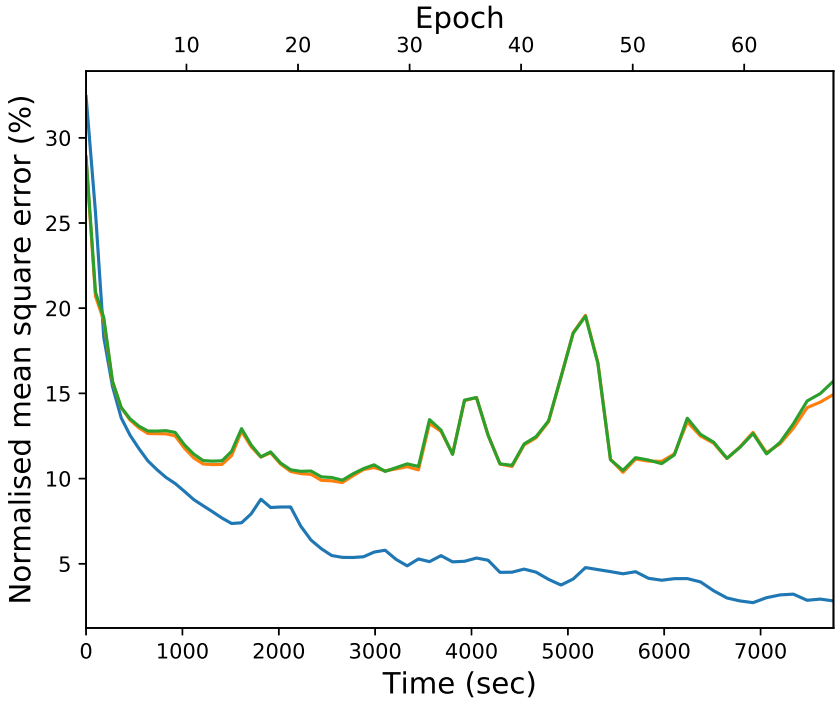


Figure 6.19: Training history of model using the augmented aggregative function, training (blue), validation (orange) and testing (green) datasets: third case study.

The algorithm is performing for different sizes of trusses in all the case studies; however, its generalisation potential is also tested here for trusses that have more nodes than the trusses of the datastes used for training and testing. A new dataset was created, similarly to before, but with the number of nodes randomly sampled from the interval $[41, 60]$. The

total error in the newly defined dataset was 7.06%, which is an interesting result, implying that the algorithm has learnt the physics of the interaction of different members of the trusses.

The two latter case studies reveal that the algorithm is able to perform data normalisation, together with inference within the populations of the structure. The GNN model proves quite versatile in solving such complicated problems, and could be further exploited for PBSHM applications. Although the application presented might seem quite complicated, a schematic representation of the whole fibre bundle idea for PBSHM is shown in Figure 6.20. The algorithm essentially offers an alternative to explicitly embedding structures on the manifold $\mathcal{M}$, which most probably is not a trivial procedure and maybe requires a straightforward embedding of the structures. Bypassing the embedding step, the algorithm provides a way of directly performing inference on the structure/graph as an entity.

6.4 Summary

The method proposed in this chapter proves quite versatile and learns the physics of the structures of the population quite efficiently. The three case studies here reveal that the method can provide accurate predictions for structures with a variety of structural members, which were encoded using categorical variables rather than their quantitative characteristics. Moreover, the algorithm was able to perform in structures with more structural members than those of the training population, enforcing further the belief that the underlying physics of the interaction between structural members has been learnt. It would be very interesting to test the method on populations of experimental structures, as well as on tasks other than predicting the normal-condition characteristics, e.g. in building predictive temporal models that can sufficiently perform throughout the population.

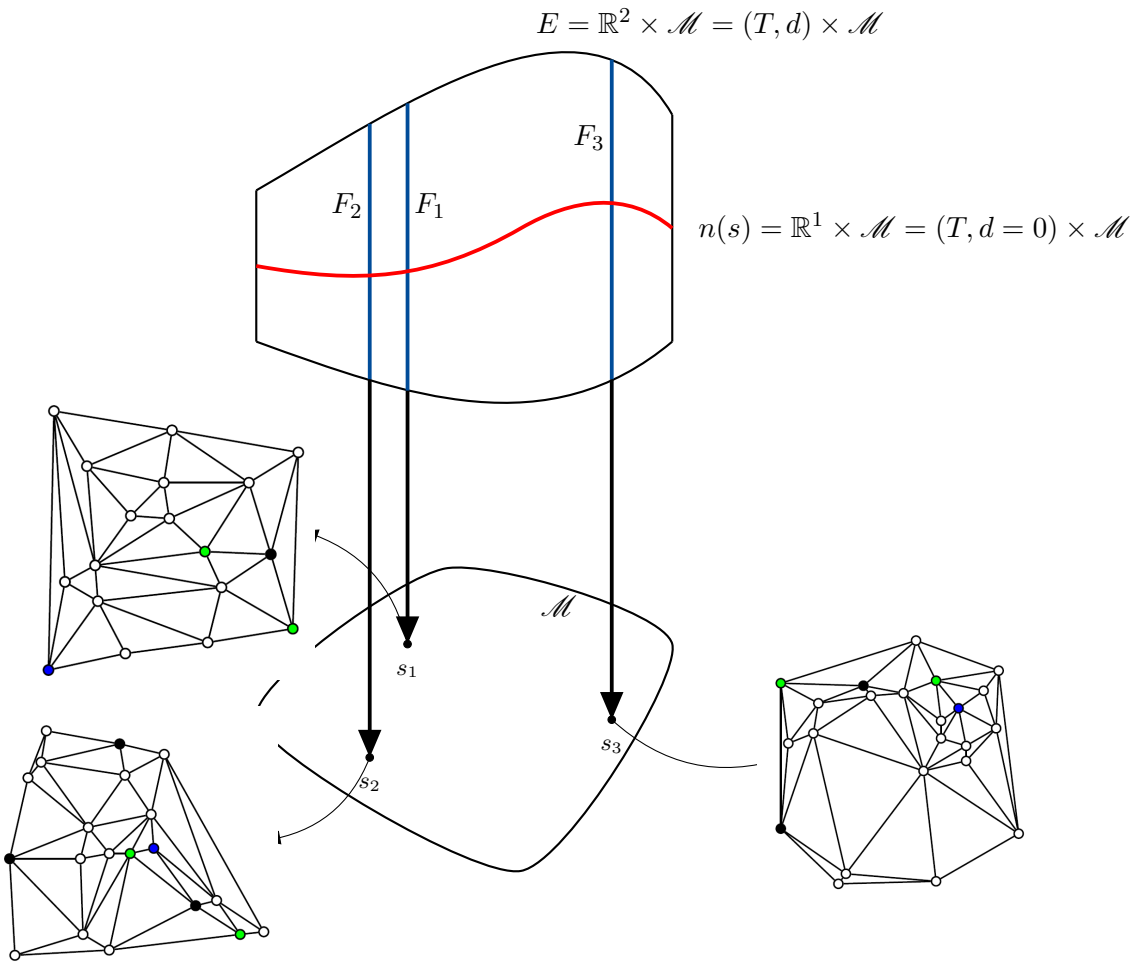


Figure 6.20: Abstract scheme of the application. Nodes within $\mathscr{M}$ represent random trusses from within the population (three examples are shown here; black nodes represent fully-fixed nodes, green nodes are fixed in the x direction, blue are fixed in the y direction and in white are free (pinned) nodes). A fibre F_i for each structure s_i is schematically shown as a blue line; it is parametrised by all potential first natural frequencies of the structure for varying temperature T, and damage coefficient d. The algorithm here has approximated the normal condition cross section $n(s)$ where damage d is equal to 0 and for any potential temperature T.

Chapter 7

Exploring structural-state manifolds

7.1 The conditional adversarial network (cGAN)

A particularly-useful variation of the generative adversarial networks [12], is that of the *conditional adversarial networks* (cGAN) [147]. The algorithm provides a generative model whose output is conditioned on a vector of variables, called the *code* here. The model learns from data, to generate sets of samples that look real, and change according to some known measured variables defined by the user. Being able to learn such dependencies between generated samples and control variables, the algorithm fits really well the needs of modelling the behaviour of structures under uncertainty and varying environmental and operational variables (EOVs).

The layout of the cGAN (Figure 7.1) is quite similar to the layout of the traditional GAN, as defined in [12]. As shown in Figure 7.1, the output of the generator depends on some noise vector $\boldsymbol{z}$, but also on the aforementioned code vector $\boldsymbol{c}$. The decision on the probability of the discriminator, regarding whether a sample is considered real or fake, also depends on the code $\boldsymbol{c}$, forcing the discriminator to learn a different decision boundary for different values of the code. The generator, in turn, is forced to generate batches of

samples that depend on the code, in order for its samples to be accepted as real by the discriminator.

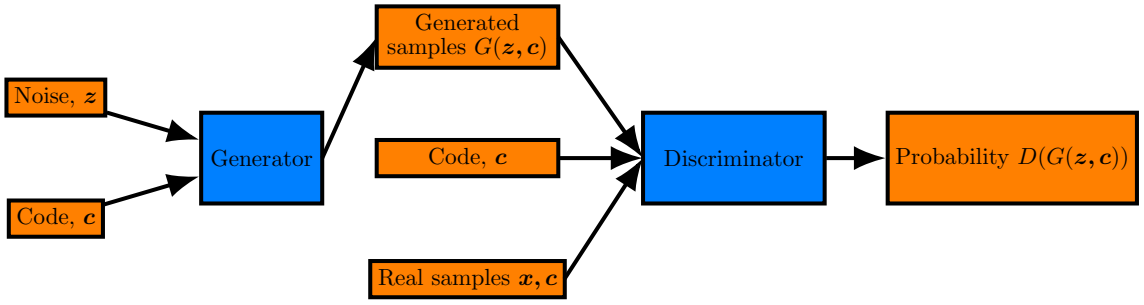


Figure 7.1: Layout of a cGAN.

The training procedure is similar to the training procedure of the GAN. As a result, the generator learns to generate distributions of data, or manifolds of samples conditioned on the code. The code can be categorical or continuous. In the case of categorical code, which can be encoded as a set of binary variables, the generator simply learns to generate data within the given-by-the-code class of data. For SHM purposes, this might result in generating data for some structure that corresponds to its undamaged state or to some specific damage case. More specifically, the generator G is able to generate artificial data $G(\boldsymbol{c}, \boldsymbol{z})$ so that,

$$G(\boldsymbol{c}, \boldsymbol{z}) \sim P(\boldsymbol{x}|\boldsymbol{c}) \tag{7.1}$$

where $\boldsymbol{x}$ are the real data samples, $\boldsymbol{z}$ is the noise vector, $\boldsymbol{c}$ is some code vector and $P(\boldsymbol{x}|\boldsymbol{c})$ is the probability density function of the real data, conditioned on the code $\boldsymbol{c}$. Such an approach can be exploited in order to generate artificial data in order to better train some data-driven algorithm.

In the case of a continuous code, the results are more interesting. The continuous code variables force the decision boundary of the discriminator, and in turn the manifolds of generated data, to gradually transform according to the values of the code. These transformations in structural dynamics could represent the effect of known variables on the recorded behaviour of structures. At the same time, the noise vector is used by the algorithm to model the effect of unknown variables and noise on the behaviour of the structure.

Because of its ability to encode the effect of known parameters in the code vector and the effect of unknown ones in the noise vector, the algorithm may be exploited in several ways. A first use, could be to generate data for varying EOVs that affect a structure, knowing the values of only a subset of them. Since the algorithm can learn the transformation of the manifold of potential states of some structure for different values of known EOVs, the generator can be used with a view to generating manifolds for values of EOVs that have not been recorded. A second use can be to consider a cGAN as a basis for a mirror model [35] of some structure. As already mentioned, generative models could be used as mirrors of structures, in order to deal with the inherent uncertainty that such modelling procedures require.

7.2 Generating parametrised structural data using cGANs

As described, structures exist within an environment, which affects their behaviour. Various parameters of the environment cause variations in the recorded features of structures. Temperature, humidity, variations on the characteristic of the loading, boundary condition variations, ground characteristics etc. may cause variations in the acquired data from the structure of interest. For example, lower temperatures, in general, increase the stiffness of structural members. Such an increase could lead to higher natural frequencies of the structure in the recorded data.

When one monitors a structure, such benign variations may be misleading, creating the impression that damage is present and initiating unnecessary maintenance procedures. In order to deal with such problems, the model that is used to perform inference on the status of the structure should be able to take into account the effect of unknown parameters and noise that may be present in the acquired data. In the current section, the use of a cGAN model is proposed, in order to generate structural data for different values of some recorded environmental parameters along with the effect of unknown underlying parameters.

Considering an observation $\underline{p}$, of some feature of a structure, then the model M that models the behaviour of the structure according to the environmental parameters is,

$$\underline{p} = M(\boldsymbol{e}_c, \boldsymbol{e}_u) \tag{7.2}$$

where $\boldsymbol{e}_c$ is the vector containing the known and recorded environmental parameters that affect the observation $\underline{p}$, and $\boldsymbol{e}_u$ are the unknown ones and the potential noise that might affect the recordings of the data. The observation could be the first n natural frequencies of the structure or an FRF or a transmissibility. In any case the observed value of such features is affected by the environmental parameters.

A common approach to damage detection under varying EOVs, is to define a model that predicts the behaviour of the structure for different values of the EOVs. The known parameters, of course, could be the only ones used in a deterministic model in order to perform such inference. The error of such an approach could be considered the effect of epistemic and aleatory uncertainty, but excessive errors could also indicate the existence of damage. However, cGANs offer a way to generate potential outcomes, taking into account such uncertainties. Instead of generating a single prediction for every value of the known parameters, the cGAN is able to generate a set of potential results. This way, for every value of the code $\boldsymbol{c}$, a range of potential outcomes is generated, assisting in avoiding mistakes and false alarms because of benign variations.

Instead of using equation (7.2), the potential outcomes of the cGAN generator G are given by,

$$\underline{p}(\boldsymbol{c}) \in \{\underline{p}_i = G(\boldsymbol{c}, \boldsymbol{z}_i),\ i = 1, 2..., N,\ \boldsymbol{z}_i \sim P(\boldsymbol{z})\} \tag{7.3}$$

where $\underline{p}(\boldsymbol{c})$ is an observation of the structural features, given the value of the code $\boldsymbol{c}$, N is the number of artificially-generated samples by the generator, and $P(\boldsymbol{z})$ is the probability distribution of the noise vector, which is predefined. The set of $\underline{p}_i$ defined in equation (7.3), is the set of potential outcomes for some value of the code $\boldsymbol{c}$.

7.2.1 Application example

The proposed algorithm can be applied to a simulated dataset to illustrate its potential. The dataset here is created by considering the structure shown in Figure 7.2. The excitation force, applied to the first degree of freedom of the mass-spring system, was white Gaussian noise. As environmental parameters, temperature and humidity were considered to affect the structure. Temperature is considered here to affect the stiffness of the springs and humidity is affecting the value of the damping parameters. More specifically, the

relationship between stiffness and temperature is given by,

$$k(T) = k_0 * (1 - \frac{T - 30}{100}) \tag{7.4}$$

where T is the value of the temperature, k_T is the stiffness of the members, for some temperature T and k_0 is the value of the stiffness for $T = 30$, considered the reference temperature of the environment. According to equation (7.4), higher values of temperature imply lower stiffness values, and lower temperatures have the opposite result.

Humidity was considered to affect the damping coefficients in a similar way; the relationship between humidity and the damping coefficients is given by,

$$c(h) = c_0 * (1 - \frac{h - 90}{100}) \tag{7.5}$$

where h is the value of the humidity, $c(h)$ is the damping coefficient for the value h and c_0 is the base damping coefficient considered for $h = 90\%$.

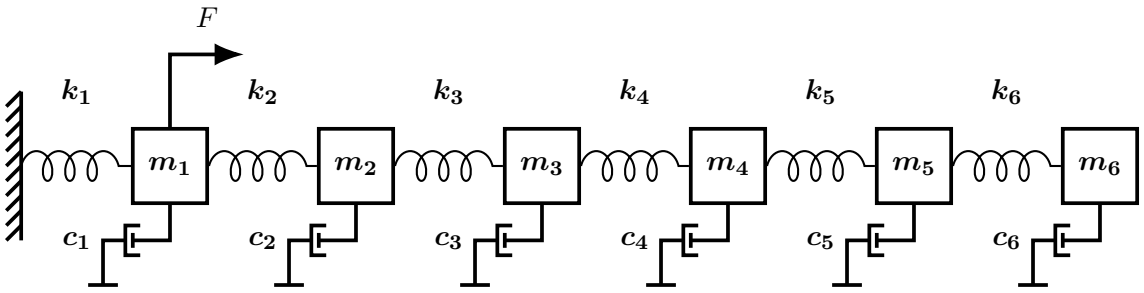


Figure 7.2: The six-degree-of-freedom mass-spring system used in the current application.

In order to create a dataset, simulations were performed for temperature values varying in the interval $[20, 40]$, and humidity in the interval $[80\%, 100\%]$. The base value of the stiffness was $k_0 = 10^4$, for the damping coefficient $c_0 = 10$ and $m_i = 1,\ i = 1, 2...6$. The number of discrete temperatures considered within the aforementioned interval was 11 and for each discrete temperature, 1000 simulations were performed. For every simulation, the transmissibilities between masses $1 - 2$ and $2 - 3$ were recorded, polluted with Gaussian noise (r.m.s. equal to 5% of the signal's variance) and concatenated. A sample of the concatenated transmissibilities is shown in Figure 7.3.

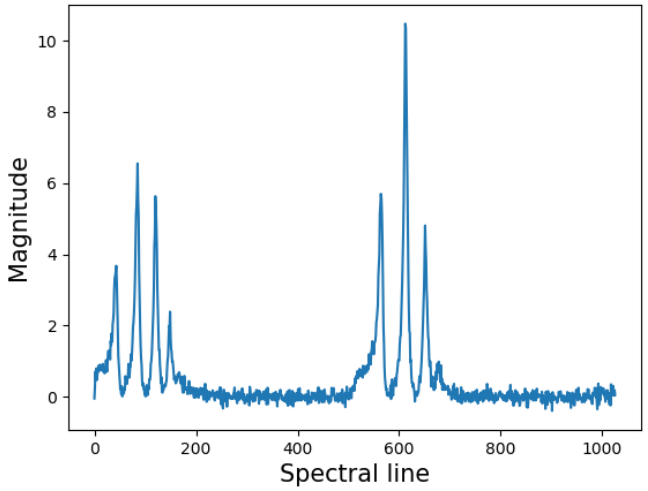


Figure 7.3: Example of simulated transmissibilities recorded.

In order to reduce the dimensionality of the dataset and to visualise it, principal component analysis (PCA) was performed on the samples. The first three principal components explained 95% of the data and so they were considered a sufficient summary. In Figure 7.4, the available dataset is shown. Different temperatures are shown with different colours. Each small batch of points corresponds to a discrete temperature, and the variance within the batch is caused only by the varying humidity. Moreover, one can notice the transformation of the small manifolds as a function of the temperature. The cGAN is called to learn such a transformation and generate potential transmissibilities for values of the temperature that have not been recorded.

The procedure followed to train the cGAN model, is a cross-validation procedure, which is commonly used in machine learning model training. From the 11 temperatures of the simulated dataset, one of them is considered to be the validation dataset (the one corresponding to $24°C$) and one was considered the testing dataset (the one corresponding to $34°C$). The generator and the discriminator were both neural networks with an input layer, a hidden layer and an output layer. The noise vector was chosen to be a two-dimensional noise vector, since higher-dimensional noise vectors did not yield better results. The noise variables were uncorrelated random variables, sampled uniformly from the interval $[-1, 1]$. Different sizes for the hidden layers from the set $\{10, 20, ...790, 800\}$ were used, initialising the trainable parameters ten times for each size. The activation function used in each case

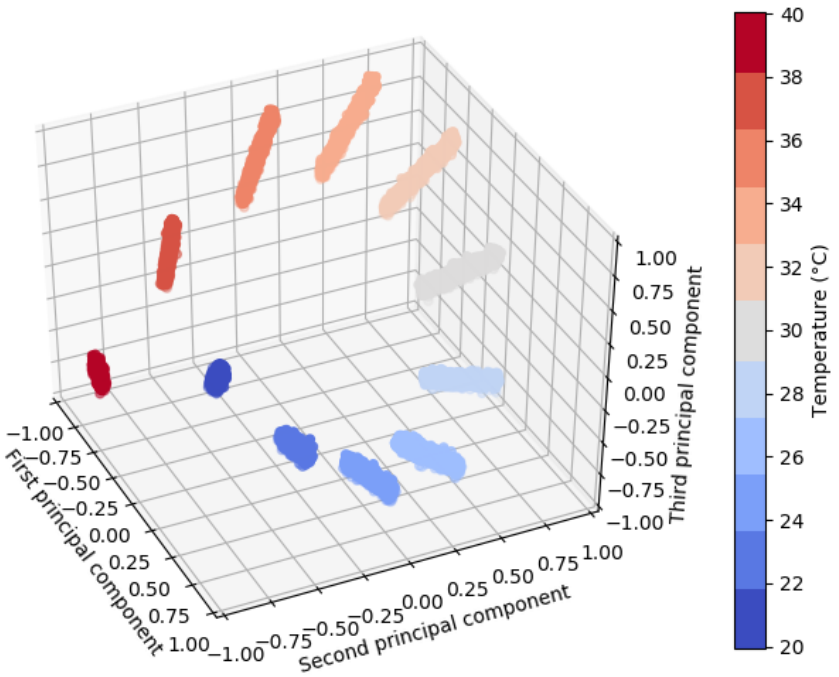


Figure 7.4: First three principal components of simulation data.

was a *hyperbolic tangent*, except for the output node of the discriminator, which was a *sigmoid* function, since it should correspond to the probability of the input sample being real or artificial.

The validation procedure followed was to calculate how close are the manifolds of the generated and the real data for some specific code. In the current work, to calculate such a metric, the *Kullback–Leibler divergence* (KL divergence) was used [148]. KL divergence is often used as a metric to measure the distance between two probability density functions. The training procedure of the GAN implicitly minimises the KL divergence between the probability density functions of the generated and the real data. However, the adversarial loss used in training does not explicitly reflect the quality of training. Therefore, the model that is considered to perform best shall be the one that has the minimum KL divergence for the codes in the validation dataset.

The total KL divergence of a model is calculated in its discrete form and is given by,

$$\sum_{n=1}^{n_{val}} D_{KL}(P_i||Q_i) = \sum_{n=1}^{n_{val}} \sum_{x \in X} P_i(x) \log(\frac{P_i(x)}{Q_i(x)}) \tag{7.6}$$

where the P_i and Q_i distributions are kernel density estimates fitted to the real and generated data, for the i^{th} code. On both the generated and real data of the i^{th} code, a kernel distribution [64] is fitted and the above equation is used to calculate the KL divergence over a grid of points in the feature space of the data. The range of all variables of the feature space is the interval $[-1, 1]$, since they are outputs of neural networks and therefore normalised in the aforementioned interval. The bandwidth of the kernel distributions used was 0.2, since it was considered that a value equal to $\frac{1}{10}$ of the total range can sufficiently describe the distribution of data.

Following the cross-validation approach, the model that performed best on the validation dataset had 200 neurons in its hidden layer. The performance of the model on the testing set can be visually evaluated from Figure 7.5. As shown, the generated points for the testing temperature value are very close to the points of the testing dataset. Such results reveal that the algorithm is able to learn the transformations of the manifolds of potential structural states according to known EOVs - the temperature in this case. At the same time, the algorithm takes into account the unknown parameters - the humidity in this case. In order to evaluate the reconstructed transmissibilities, in Figure 7.6 a real transmissibility for $20°C$ is shown with, a real and a generated transmissibility for $34°C$. The algorithm seems to have captured correctly the movement in the diagram as a result of the temperature change.

Another use of a generator trained according to the procedure described, would be to generate the whole manifold of potential states of the structure. Even if the accuracy, in KL divergence terms, on the testing set is not sufficient, the algorithm might be able to generate a compact manifold for every temperature in the desired interval. In contrast to the small manifolds shown in Figure 7.4, the structural states could belong to a larger manifold without any gaps. By generating samples for every value of temperature in the interval $[20, 40]$, the created manifold of potential states is shown in Figure 7.7.

Novelty detection approaches [76, 111, 149] to damage detection, require a complete set

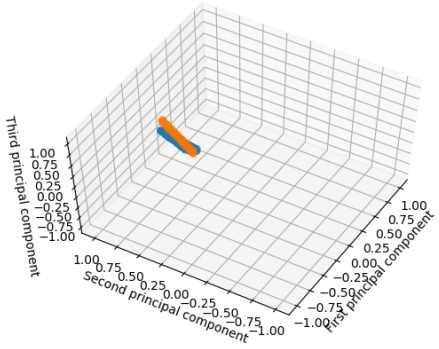


Figure 7.5: Real (orange), and generated (blue) points on the testing set for the temperature value of 34°C.

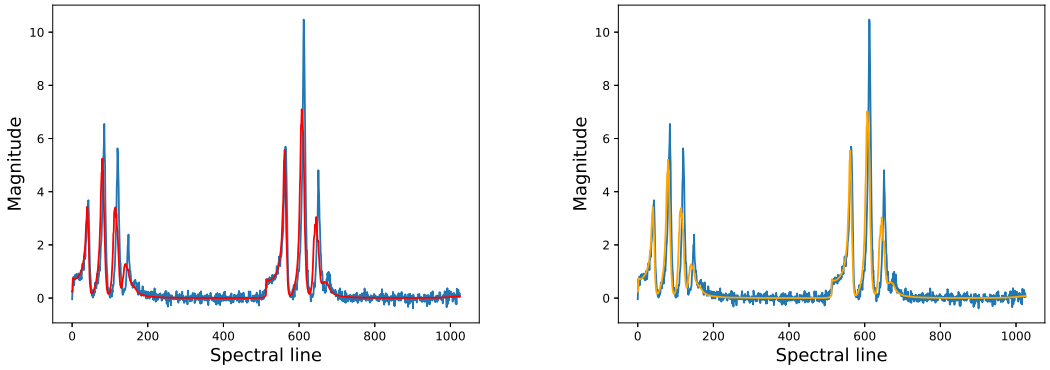


Figure 7.6: Transmissibility recorded from simulations with temperature 20°C (blue) compared to recorded transmissibility in 34°C (orange) and generated (red).

of observations in order to perform as intended. Such approaches essentially learn the distribution of the data or the manifold that the healthy data form. Therefore, they are used in order to indicate when a test sample is not likely to belong to the distribution of the healthy data or their manifold. By using the current approach in order to define a more integrated dataset of potential states, the performance of novelty detection algorithms could be enhanced, avoiding unnecessary damage alerts.

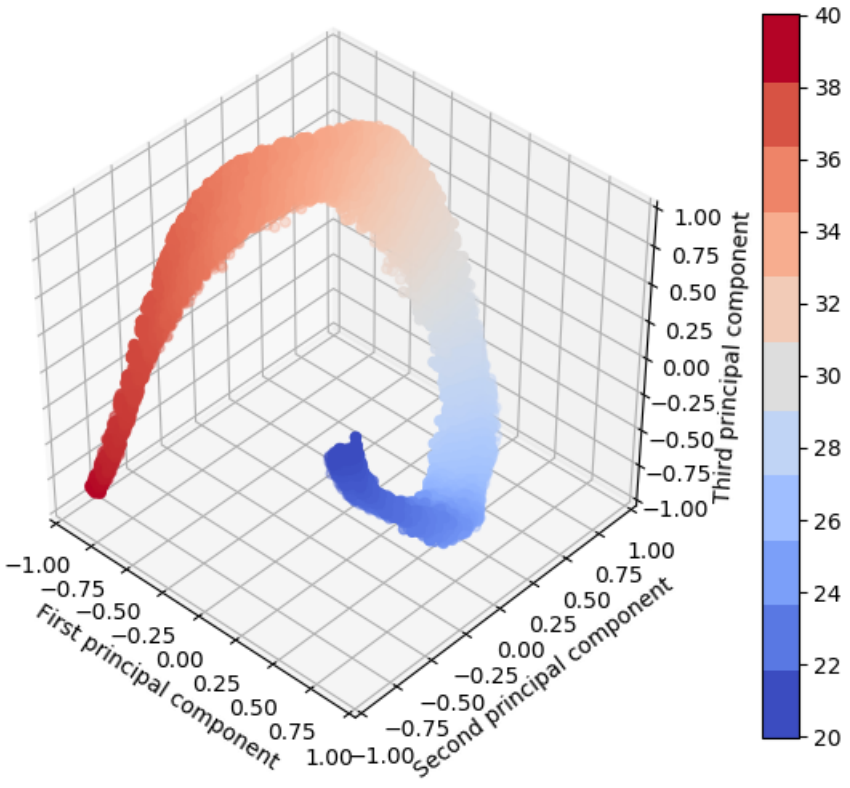


Figure 7.7: Manifold representing samples for all temperature values in the range $[20, 40]$.

7.3 Generative models as mirrors of structures

As mentioned in Section 2.2, generative models can be used as mirrors of structures. In particular, cGANs fit the framework well, since they are able to generate distributions of data, given some deterministic known input. Following the terminology of mirrors, the inputs to a generative model are separated into controlled $\underline{e}_c^C$ and uncontrolled $\underline{e}_u^C$. The controlled are known and recorded variables that affect the output of the model, and in terms of a cGAN they can be represented by the code $\boldsymbol{c}$. The uncontrolled variables, on the other hand, affect the output of the model. As shown in the previous section, the uncontrolled variables can be both noise and unknown parameters affecting the behaviour of a structure.

In order to illustrate the ability of cGANs to serve as mirrors of structures, two applications and a comparison are presented here between the specific algorithm and a physics-based generative model, the *stochastic finite element method* (SFEM). For further information about the method, one can refer to the Appendix or to [150, 151]. The applications refer to simulated datasets from a linear and a nonlinear structure. In both cases the quantities of interest are displacements of the structure and the two methods are tested regarding their ability to predict the distribution of these quantities. Apart from the two approaches, a hybrid model is defined as the combination of the two and presented.

7.3.1 Simulated datasets definition

The structure considered here was a cantilever with random Young's modulus E. In the linear case, the random quantity is defined as a stochastic field along the length of the cantilever, as shown in Figure 7.8. The length of the cantilever was 5 (length units), its cross section was rectangular with height equal to 0.4 length units and width equal to 0.1. The stochastic field of the Young's modulus was Gaussian, with mean value equal to 2×10^9 and standard deviation equal to $0.2 \times 2 \times 10^9$ (pressure units). The correlation length used in the application was 3. In order to generate data, an SFEM procedure was followed using an order of $m = 2$ for the expansion of the random field [151].

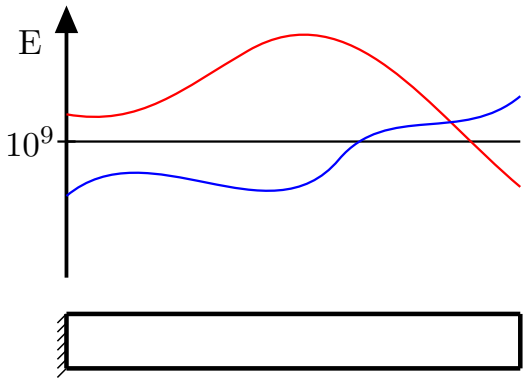


Figure 7.8: Cantilever beam with constant Young's modulus (black line) or spatially varying (red and blue lines).

The input for the simulations, which played the role of the controlled variables, was a distributed load f along the cantilever with values in the interval $[10, 200]$ (force units/length units). For each load, 1000 samples were generated, and in Figure 7.9 samples of the tip

displacements for different input loads f are shown. The tip displacement is the quantity whose distribution was to be modelled using the generative mirror models. In order to follow a machine learning training procedure, the dataset was split into three subsets; one for training/calibrating the model, one for validation and one for testing.

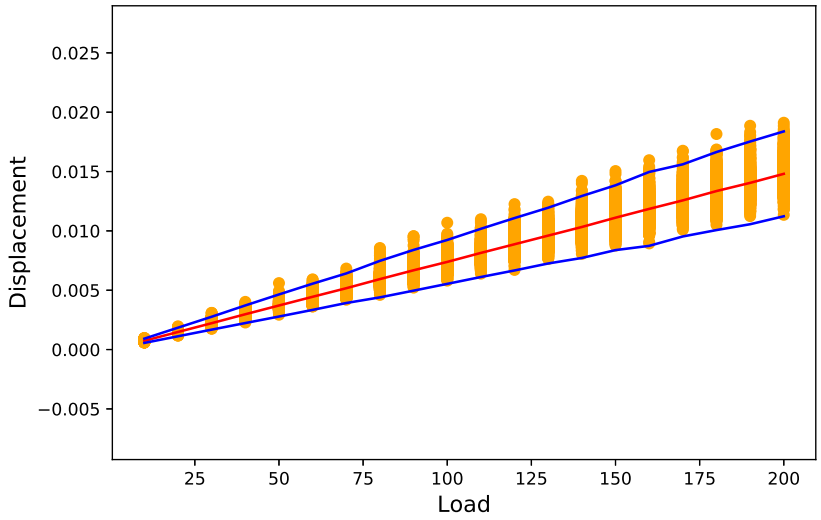


Figure 7.9: Samples of tip displacements (orange points), their mean values (red line) and ±3 standard deviations (blue line).

The nonlinear dataset referred to a cantilever with the same geometrical characteristics. This time the uncertainty was that the Young's modulus was random from a normal distribution with mean value equal to 2×10^9 and standard deviation $0.1 \times 2 \times 10^9$. The simulations were performed using 40 loadsteps and *Newton-Raphson* iterations. The total load this time was 400 load units. The difference in the displacement-load curves between the linear and the nonlinear case can be seen in Figure 7.10. Again the dataset was split in three subsets, one for training (load units $\{10, 40, 70..., 400\}$), one for validation (load units $\{20, 50, 80..., 380\}$) and one for testing (load units $\{30, 60, 90..., 390\}$).

Both the SFEM model and the cGAN were fitted on the data of the corresponding datasets, and the results are presented. Moreover, a combination of the two methods is developed and presented as a hybrid approach to the problem. The results are compared and conclusions are drawn.

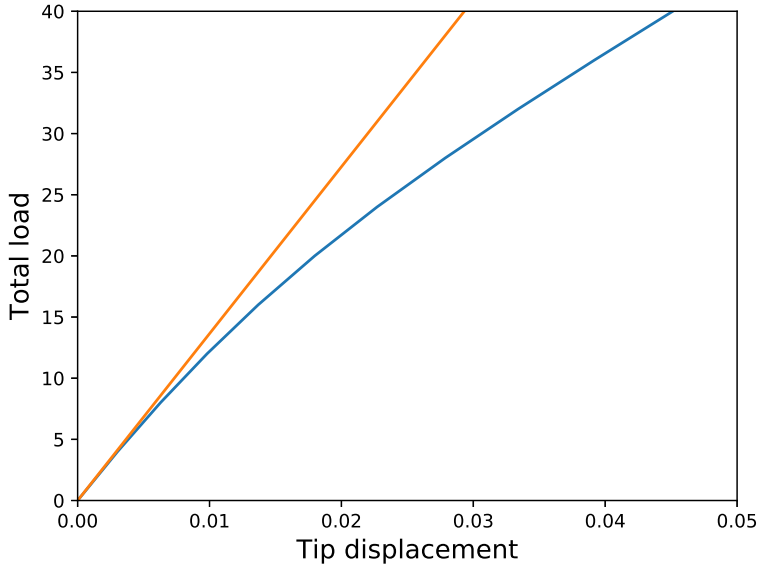


Figure 7.10: Load curves for the linear (orange), and the nonlinear (blue), material cantilevers.

7.3.2 SFEM model as mirror

The calibration of the SFEM model was performed by searching for the model parameters that minimise the error in terms of distribution distance. The adjustable parameters of the SFEM model are the mean value of the Young's modulus stochastic field (μ_E), the standard deviation of the Young's modulus (σ_E) and the correlation length (l_{corr}). As far as the search is concerned, an exhaustive search over a range of potential values for the parameters was performed. The best-fitting parameters were considered to be the ones that yielded the best results in terms of KL divergence of the distributions, for the loads of the training dataset.

In the linear case of the current application, epistemic uncertainty does not exist. The procedure followed to generate the data was almost the same as the procedure that the SFEM model follows in order to make predictions. Therefore, it is expected that the error will be really low. This is confirmed by the plot of generated distributions compared to the real ones in Figure 7.11. As shown, the predicted and real distributions are almost identical. This is also confirmed by the average KL divergence, which is equal to 0.0028.

In real-life situations, it is not expected to have so good results. Quite often, more parameters would affect the structure, which are not included in the modelling procedure, such as humidity and temperature. It is, however expected that the algorithm could be able to incorporate such variations via the stochastic parameters that are incorporated, in this case the Young's modulus.

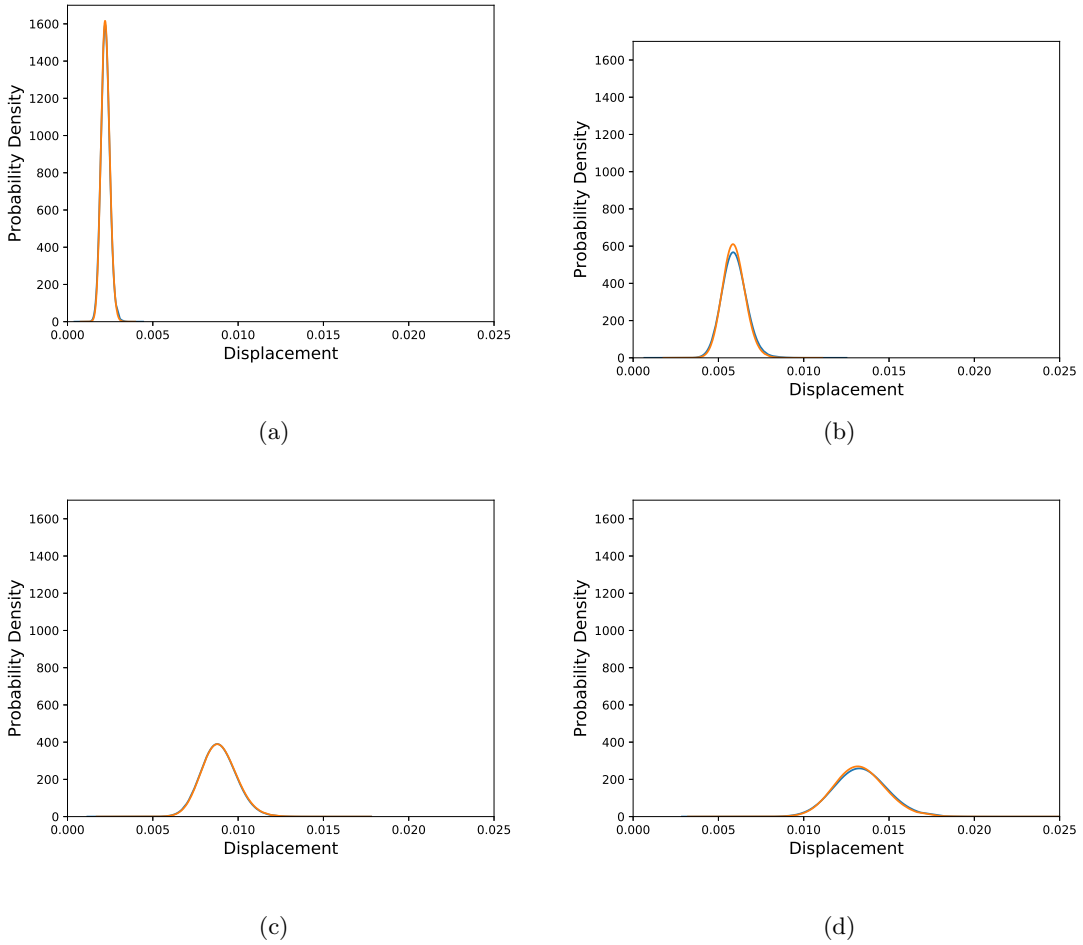


Figure 7.11: Distributions of tip displacements corresponding to Monte Carlo samples (orange), and SFEM generated samples (blue) and different load cases, (a) 30 load units, (b) 80 load units, (c) 120 load units and (d) 180 load units.

As far as the ability of the model to serve as an ϵ-mirror is concerned, the maximum KL divergence observed between the available and the predicted datasets was 0.0029. Therefore, the specific SFEM model for the specific cantilever and for the range of loads $[10, 200]$, can be considered an ϵ-mirror with tolerance $\epsilon = 0.0029$. Regarding its ability to be used as an α-mirror, a curve $\alpha - P(\alpha)$ should be provided. The curve gives the probability of the observations to fall into the interval defined in equation (2.3) as a

function of α. The curve for this case is shown in Figure 7.12. A certain way to evaluate such a curve is not available; however, it is clear that it can be used in order to perform a probabilistic cost-benefit analysis of the structure, using the specific model.

The same method was used in order to define a digital mirror of the nonlinear simulated structure. An exhaustive search was performed for the parameters of the SFEM model for the nonlinear dataset. The yielded model was tested on the testing data of the nonlinear structure. The results for some loads of the testing dataset are shown in Figure 7.13. As expected, the performance of the model is not as good as before. The average KL divergence in this case is 3.43. The observed divergence is because of the epistemic uncertainty that is present in the application. The model is a linear one and the data refer to a nonlinear structure. Although nonlinear SFEM models exist [152], they were not considered in the current application, since the goal is to illustrate how a physics-based model would fail in the presence of epistemic uncertainty. Even if a nonlinear model was used, knowledge of the exact mechanism of the nonlinearity would still be required in order for the model to perform as intended. Such knowledge is not always available in such applications.

Even though the results in the nonlinear case are not good, an SFEM model is still considered a viable option for a mirror of a structure. Such models are generative models and therefore can incorporate uncertainty during simulation of some structure. Moreover, they provide a probability density function over potential outcomes, such as equation (2.4) defines. In the aforementioned equation, the role of the generative model of the unknown parameters M_u^{EC} is played by the Karhunen-Loeve expansion of the stochastic field of the Young's modulus and the known and controlled parameters $\underline{e}_c^C$ are the different roles. As the context of the model, one may consider the range of loads for each application as the environmental context and the displacements of the tip as the predictive context. However, as in every physics-based method, SFEM models are vulnerable to epistemic uncertainty, and this is confirmed by the application of the method on the nonlinear case.

7.3.3 cGAN as mirrors of structures

As an alternative to the physics-based model presented previously, a black-box machine learning approach to defining mirrors of structures is presented in the current paragraph.

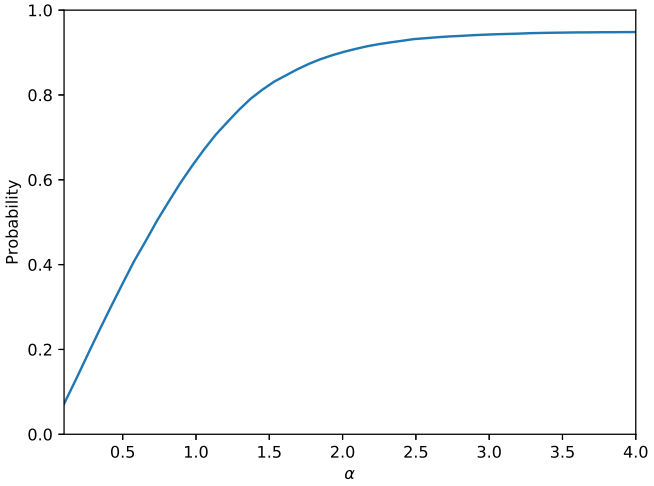


Figure 7.12: Probability defined in equation (2.3) as a function of the parameter α for the SFE model applied on the linear cantilever case study.

The selected method is the cGAN algorithm, which was shown to have potential in modelling structures under a generative framework and with the presence of unknown variables. The model is tested on the same applications as the SFE model and results are presented.

At first, the method was applied on the linear structure dataset. The method followed for training was similar to the one described in Section 7.2.1. The model was trained on data, corresponding to loads $\{10, 40, 70, 100, 130, 160,$
$200\}$. The validation procedure was performed every several epochs on the validation dataset, which included data corresponding to loads $\{20, 50, 80, 110, 140, 170, 190\}$. Finally, the algorithm was tested on a dataset which were the data corresponding to loads $\{30, 60, 90, 120, 150, 180\}$.

The generator and the discriminator were both chosen to be three-layered feedforward neural networks; they both had an input layer, a hidden one and an output. The size of the input layer is defined by the dimensionality of the noise and the dimensionality of the samples in the dataset. The noise was chosen to be ten-dimensional, since higher dimensions of noise did not yield any better results. The code was the input load and therefore the total dimension of the input layer of the generator was 11 nodes. The samples are the tip displacement, therefore the input of the discriminator, which includes a sample and the code value, is two-dimensional. The output of the generator is a sample, and is

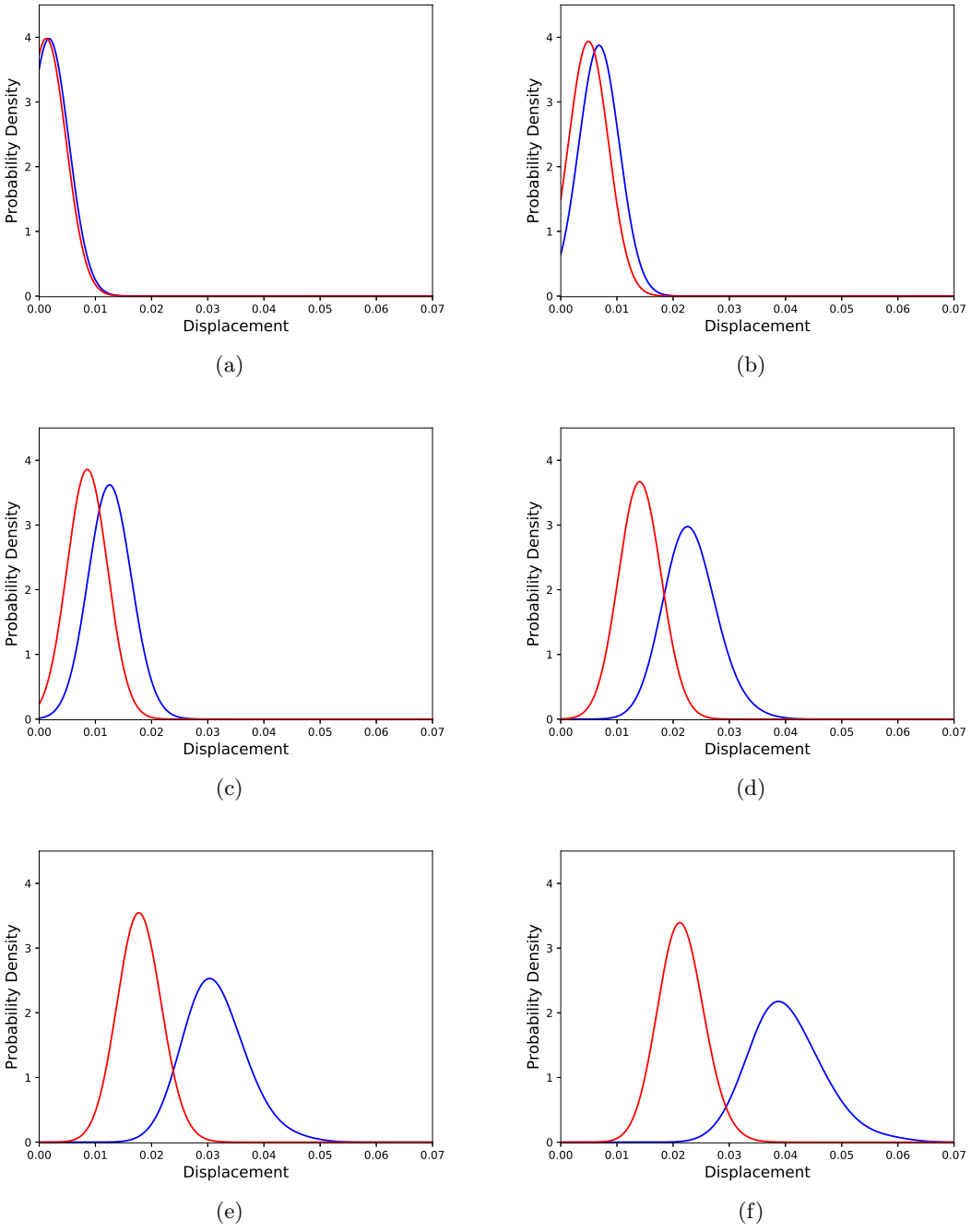


Figure 7.13: Distributions of tip displacements corresponding to Monte Carlo samples (blue) and SFEM generated samples (red) regarding the nonlinear problem, for different load cases; (a) 30 load units, (b) 90 load units, (c) 150 load units, (d) 240 load units, (e) 300 load units and (f) 360 load units.

therefore one-dimensional, as is the output of the discriminator, which is the probability of a sample being real.

The activation function which yielded the best results for the hidden layers was the hyperbolic-tangent. The activation function of the discriminator is a sigmoid function, in order to map the output into the interval $[0, 1]$, which represents the probability of the input sample to be real or not. The size of the hidden layers is a quantity which should be optimised; in order to do so, different sizes were tried in the set $\{10, 20, ..., 3000\}$. The best model was considered to be the one that yielded the minimum KL divergence on the validation set. The size of the hidden layers of the best model was 3000 nodes. The same size was used for both the generator and discriminator hidden layers, since the discriminator is an auxiliary network and its optimisation is not sought. It is also believed that using the same size conceals a physical meaning, since the generator decodes the noise into the real feature space and the discriminator maps the feature space into some latent code (in its hidden layer), in order to distinguish real and fake samples.

In order to define the real and the generated distributions, kernel density estimates were used [153]. The distributions of the samples were defined by fitting the kernel density estimates on them. The method requires definition of a bandwidth parameter. All samples are normalised in the interval $[-1, 1]$ for the purposes of the training of the neural networks and therefore a bandwidth equal to 0.1 was considered an appropriate value, as it is equal to $\frac{1}{20}$ of the total range of values. Validation was performed every 100 epochs and KL divergence was calculated according to equation (7.6). Some results regarding the performance of the cGAN on the testing dataset of the linear case are shown in Figure 7.14. The average KL divergence in the validation dataset was 0.081 and in the testing dataset 0.083

As far as the ability of the cGAN to perform as an ϵ-mirror is concerned, the maximum KL divergence observed on the testing data was 0.168. Therefore the model could be considered an ϵ-mirror with $\epsilon = 0.168$. Regarding its ability as an α-mirror, the plot in Figure 7.15 shows the curve of the probability defined in equation (2.3). As mentioned, this curve could be used by someone as a guide in order to use the generative model without any interest in the exact shape of the distribution of samples. Such applications are often used during design of structures, for example, in earthquake resilient design.

The same procedure was followed for the nonlinear case. The neural network that yielded the best results in this case, had 1110 neurons in the hidden layer. Results of the generated

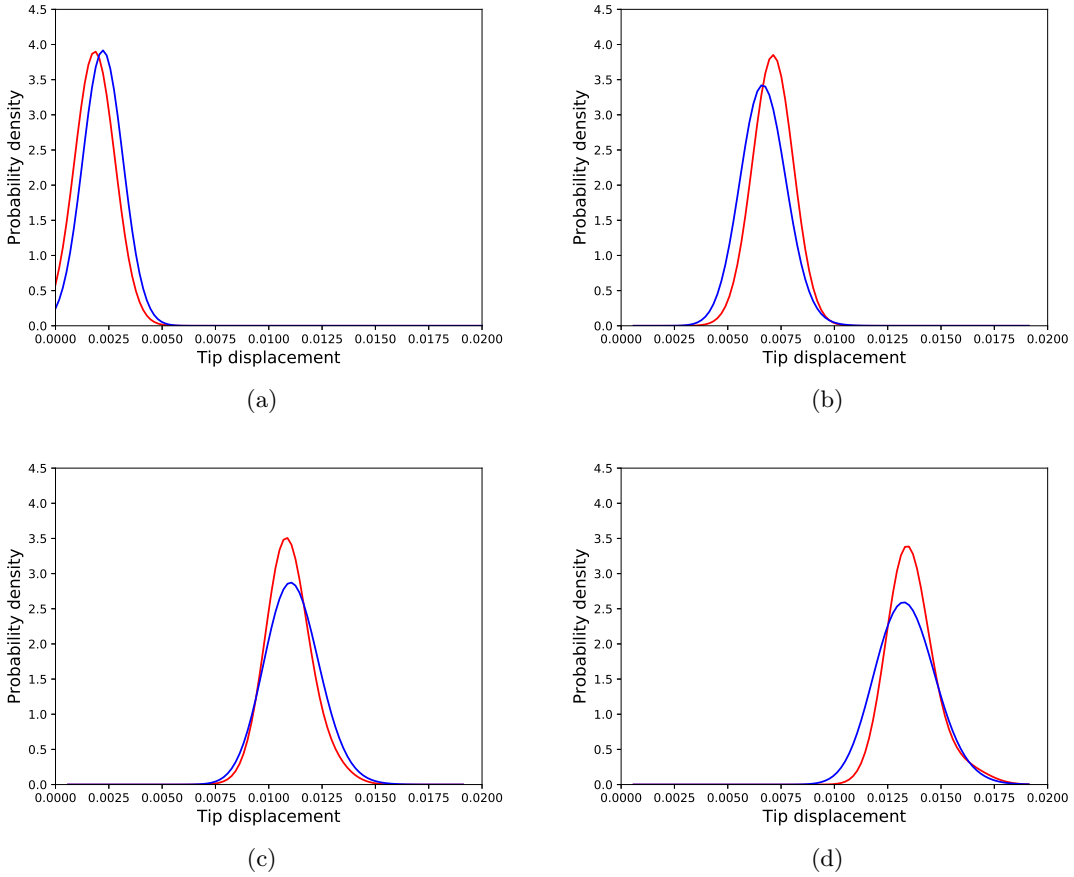


Figure 7.14: Distributions of tip displacements corresponding to Monte Carlo samples (blue) and cGAN generated samples (red) regarding the linear problem, different load cases, (a) 30 load units, (b) 90 load units, (c) 150 load units and (d) 180 load units.

and real distributions for the testing dataset are shown in Figure 7.16. The KL divergence between the generated and the real distributions of the validation dataset was equal to 0.045, and of the testing dataset equal to 0.050. The algorithm seems to have efficiently learnt both the movement of the distribution towards larger displacements for higher loads, as well as the larger spread. The results are obviously better than the results of the SFEM model in the nonlinear case. The maximum KL divergence on the testing dataset was 0.25, making the model a potential ϵ-mirror of the structure with $\epsilon = 0.25$. The $\alpha - P(\alpha)$ curve is also shown in Figure 7.17.

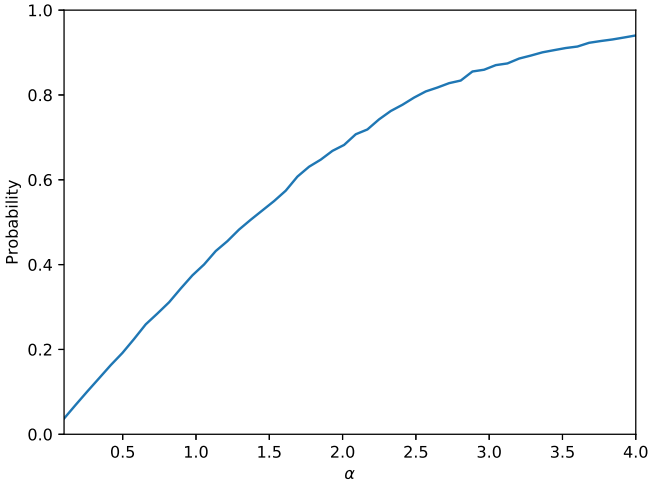


Figure 7.15: Probability as defined in equation (2.3) as a function of the parameter α for the cGAN model applied on the linear cantilever case study.

7.3.4 A hybrid mirror model

A common approach to modelling structures is to combine physics-based methods (white-box modelling) with data-driven approaches (black-box modelling) in order to define hybrid models (grey-box modelling), which are an attempt to combine the advantages of both strategies [154]. In the current work a hybrid approach to defining a generative mirror of a structure is proposed. The approach is an attempt to combine the SFEM model with the cGAN model in order to define a more accurate model in terms of KL divergence, as well as, in terms of extrapolation capabilities.

As described in [154], two types of grey-box models can be defined. The first group is referred to as A-type models, and is focused on inferring the error between the white-box model predictions and the observations using a black-box model. In the current framework, this approach is not appropriate, since specific observations do not correspond to specific predictions, rather a set/distribution of predictions corresponds to a set/distribution of observations and errors cannot be computed. Therefore only B-type models are considered. In such an approach, one tries to use the output of a white-box model together with some control variables, as inputs to one's black-box model, i.e.,

$$y(X) = g(X, f(X)) \tag{7.7}$$

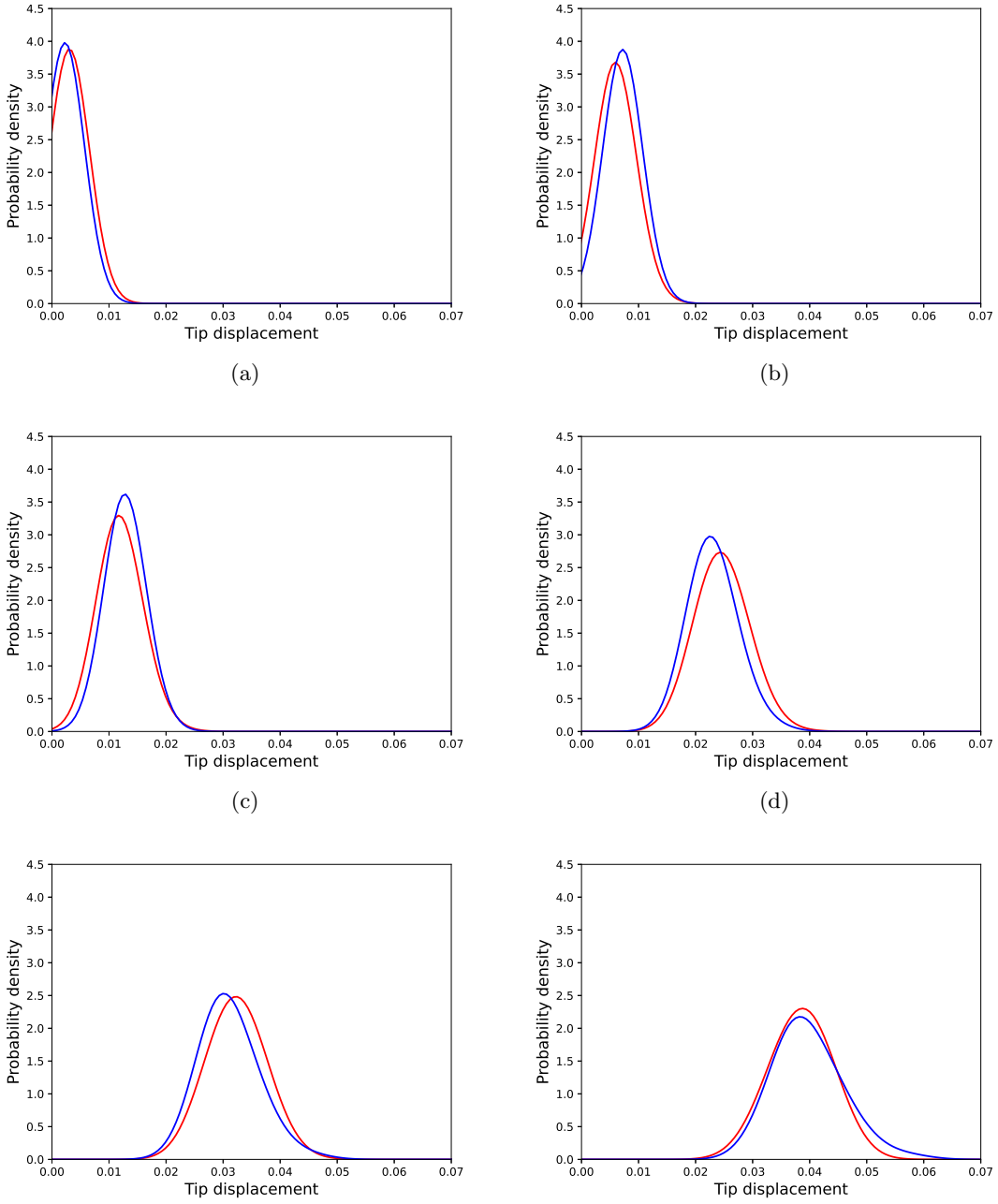


Figure 7.16: Distributions of tip displacements corresponding to Monte Carlo samples (blue), and cGAN generated samples (red), regarding the nonlinear problem, for different load cases: (a) 30 load units, (b) 90 load units, (c) 150 load units, (d) 240 load units, (e) 300 load units and (f) 360 load units.

where g is the black-box model, X is the controlled variable, $f(X)$ is the prediction of the white-box model given X and $y(X)$ is the observation. The approach is focussed on

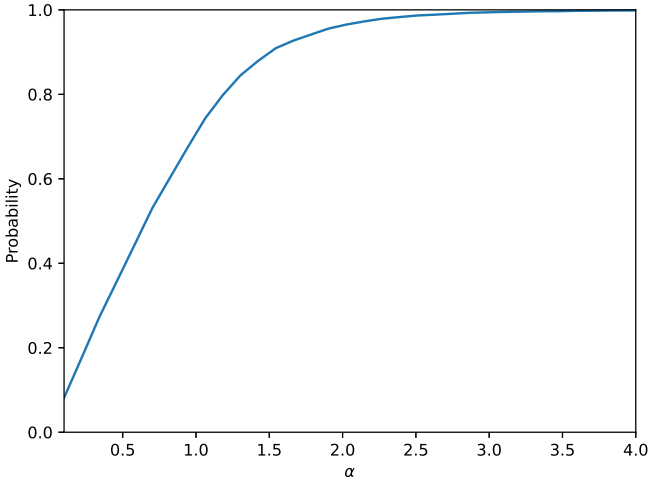


Figure 7.17: Probability defined in equation (2.3) as a function of the parameter α, for the cGAN model applied on the nonlinear cantilever case study.

informing the black-box model with the output of the white-box model. If the physics-based model has correctly captured part of the total physics of the phenomenon which is modelled, the grey-box model is expected to exploit this part and use the black-box model in order to cope with the part of the physics that is not incorporated in the white-box model.

In the current application, the white-box model has correctly captured part of the physics of the problem. The movement of the distribution towards higher values of tip displacements as the load increases, is the part of the physics that is correctly incorporated. As shown in Figure 7.13, even though the distributions do not fit, the movement of the generated distribution follows the increase of the load. Using a hybrid approach, this knowledge, which corresponds to the linear part of the physics, should be exploited by the black-box model and provide accurate predictions.

The hybrid model herein is schematically shown in Figure 7.18. The output samples of the physics-based SFEM model are used as the input noise to the generator of the cGAN. The rest of the model is similar to the cGAN model described in the previous section. Via such a model, the distribution of the noise is a function of the code, informing the generator further about the target distributions of the algorithm.

Following the same cross-validation procedure as before, the neural network that yielded

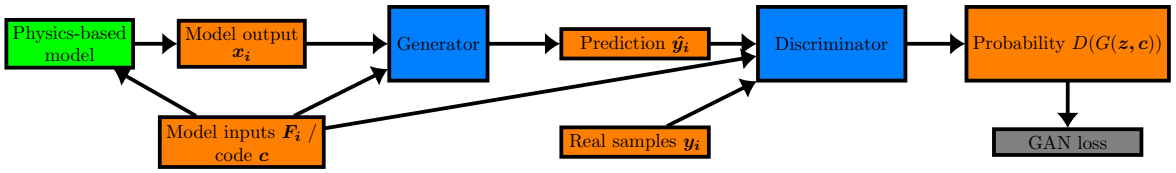


Figure 7.18: Layout of the SFE-cGAN hybrid model.

the best results had 1500 neurons in its hidden layers. The average KL divergences achieved in the validation and testing dataset were 0.048 and 0.044 respectively. In Figure 7.19, some generated and real distributions are shown. Regarding the model's ability to serve as an ϵ-mirror, the maximum KL divergence in the testing dataset was 0.22, making the model a candidate ϵ-mirror with $\epsilon = 0.22$. The corresponding α curve is shown in Figure 7.20.

7.3.5 Extrapolation capabilities

A major advantage that one expects the white-box models to have, and the hybrid models to inherit, is their extrapolation capabilities. Since white-box models are built to model the physics of the structure, they can potentially be used outside the range of data that were used to calibrate it. In black-box models, especially in machine learning, such a strategy is called extrapolation and is usually discouraged. Since grey-box models are built in order to exploit the positive elements of both white and black-box models, in the current section, the extrapolation capabilities of the hybrid model are studied.

In order to do so, the available dataset of the nonlinear structure is split into two subsets. The first one contains the load units $\{10, 20...310\}$, and will be used for the cross-validation procedure. The second subset, referred to as the extrapolation dataset, contains the rest of the data corresponding to load units $\{320, 330...400\}$. The models are trained using the first subset and are tested on the second set, in order to expose their extrapolation capabilities.

A first step that needs to be performed before attempting extrapolation is to redefine the interval into which the inputs and the outputs of the neural networks are scaled. A common approach to defining such an interval, which was also followed in the applications so far, is to scale every dataset in the interval $[-1, 1]$. This strategy, however, might saturate

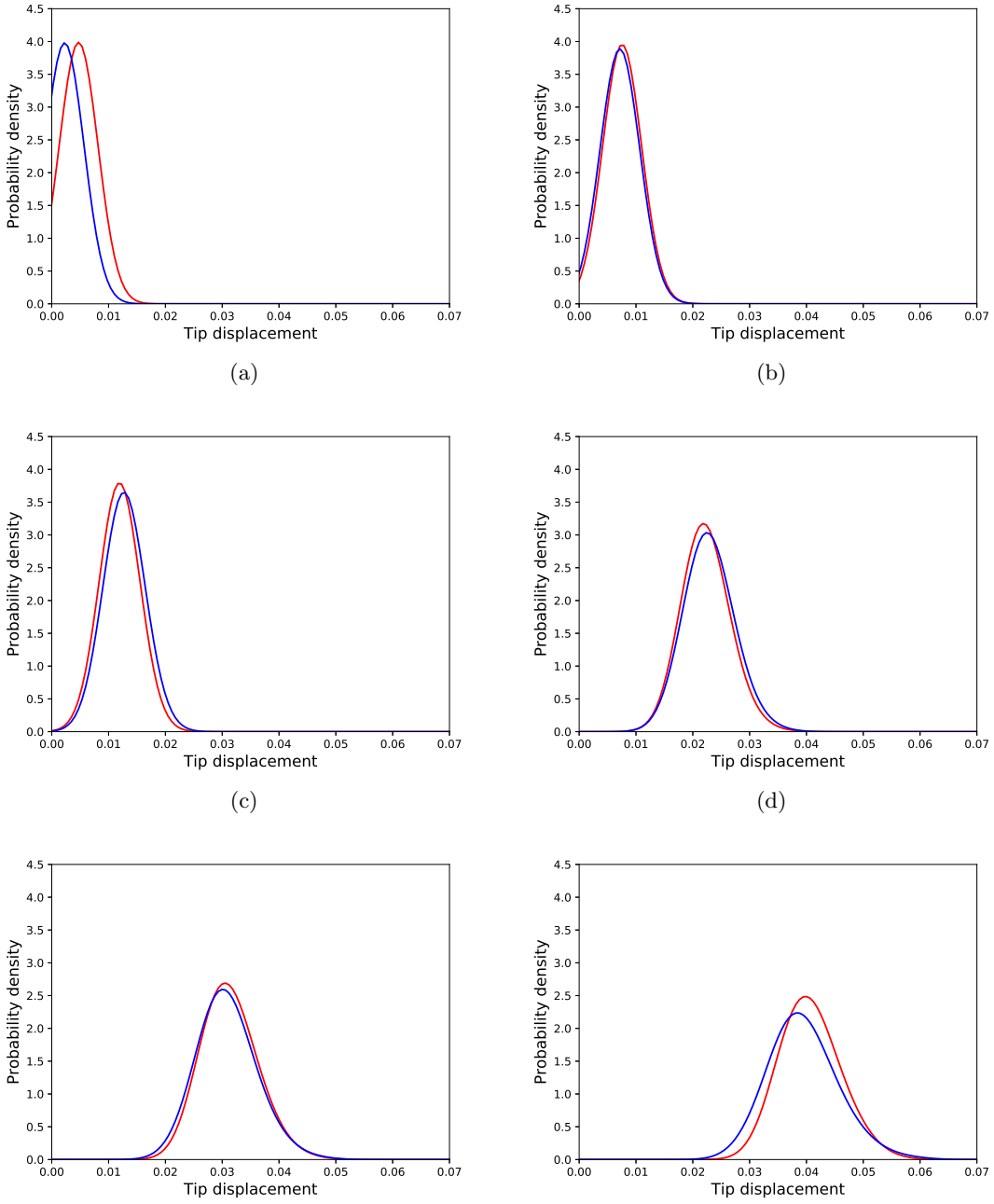


Figure 7.19: Distributions of tip displacements corresponding to Monte Carlo samples (blue), and generated samples by the cGAN-SFEM hybrid approach (red), regarding the nonlinear problem, for different load cases: (a) 30 load units, (b) 90 load units, (c) 150 load units, (d) 240 load units, (e) 300 load units and (f) 360 load units.

the sigmoid activation functions of the algorithm, whose outputs are always within the interval $[-1, 1]$, and makes extrapolation even more difficult. To deal with such an issue,

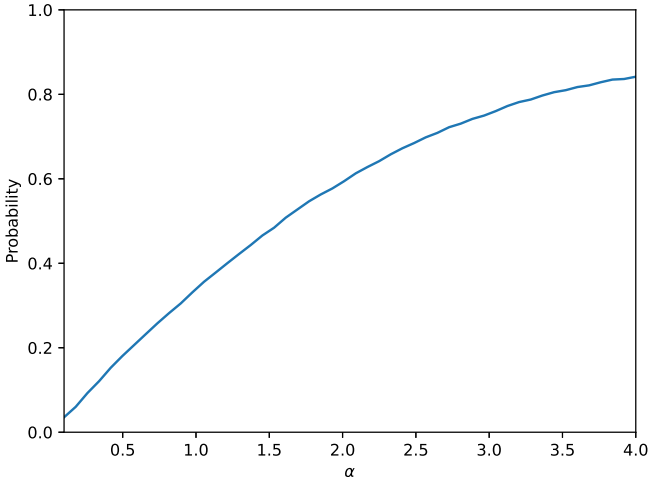


Figure 7.20: Probability defined in equation (2.3) as a function of the parameter α for the cGAN-SFEM hybrid model applied on the nonlinear cantilever case study.

for the current application, the inputs and the outputs of the cGAN are scaled onto the interval $[-0.8, 0.8]$. By doing so, the sigmoid functions may stop being saturated, and thus the probability of the model to extrapolate increases.

The hybrid and the black-box models were tested as far as their extrapolation capabilities are concerned. The black-box model yielded a KL divergence on the validation dataset equal to 0.048, and equal to 0.054 on the testing dataset. In Figure 7.21, a similar comparison to the ones shown before, is presented. The predictions of the model on the extrapolation dataset, had an average KL divergence equal to 0.77, and some of the distributions are shown in Figure 7.22. It can be seen that the distributions move slightly towards higher values of displacements as the load increases, but in any case the generated distributions do not fit the real ones.

Following the same procedure for the hybrid model, the model that yielded the best results had 1500 neurons in its hidden layer. The average KL divergences achieved in the validation and testing datasets were 0.034 and 0.038 respectively. In Figure 7.23, some comparisons between real and generated distributions are shown. As before, the model is also evaluated on the extrapolation dataset. The average KL divergence for this latter dataset was 0.288 and some distribution comparisions are shown in Figure 7.24. It is clear that the contribution of the white-box model is positive in the overall result.

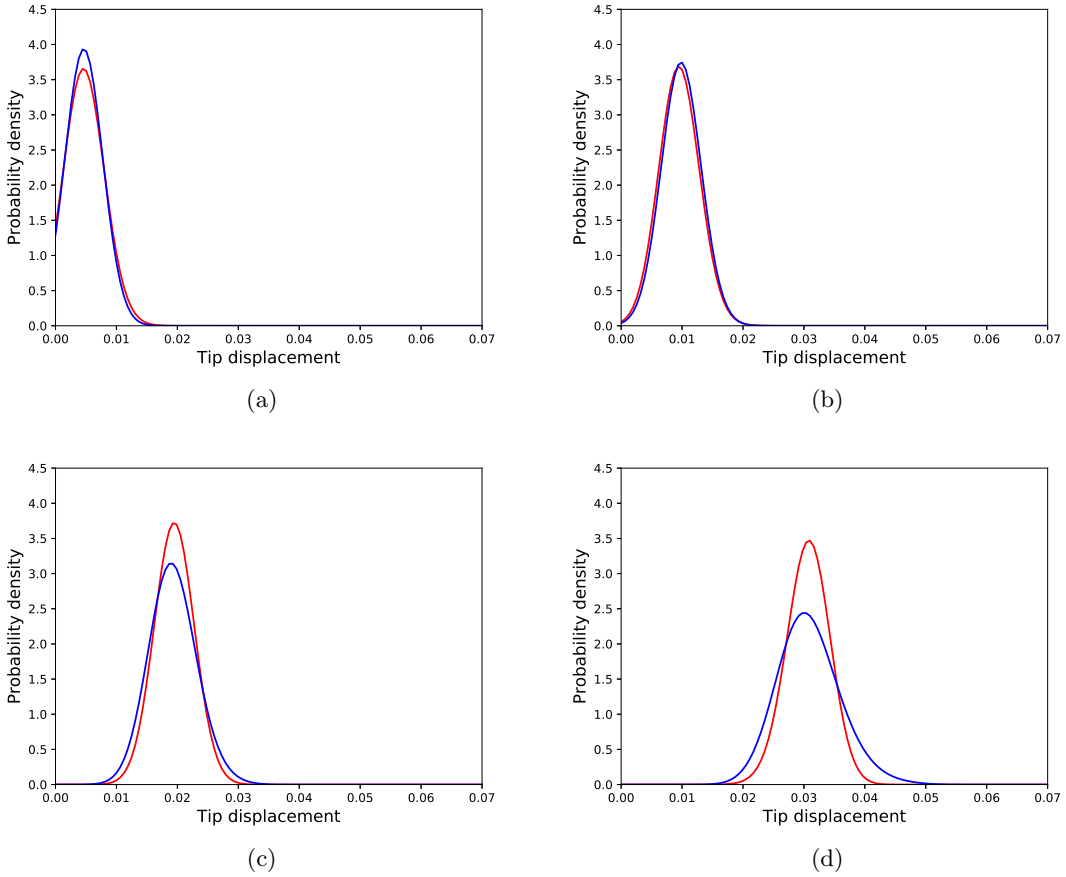


Figure 7.21: Distributions of tip displacements corresponding to Monte Carlo samples (blue), and generated samples by the cGAN model (red), regarding the nonlinear problem, trained according to the reduced dataset for different load cases: (a) 60 load units, (b) 120 load units, (c) 210 load units, (d) 300 load units.

The distributions are moving more efficiently towards higher displacements as the load increases, meaning that the correct part of the physics of the SFEM model is inherited.

The results encourage the belief that a hybrid model may be able to inherit positive aspects from both its black-box part and its white-box part. In the current application, the combination of a black-box model and a white-box model for simulations under uncertainty, was able to provide a model that is performing better than the plain black-box model and at the same time is able to extrapolate, an ability that black-box models lack completely.

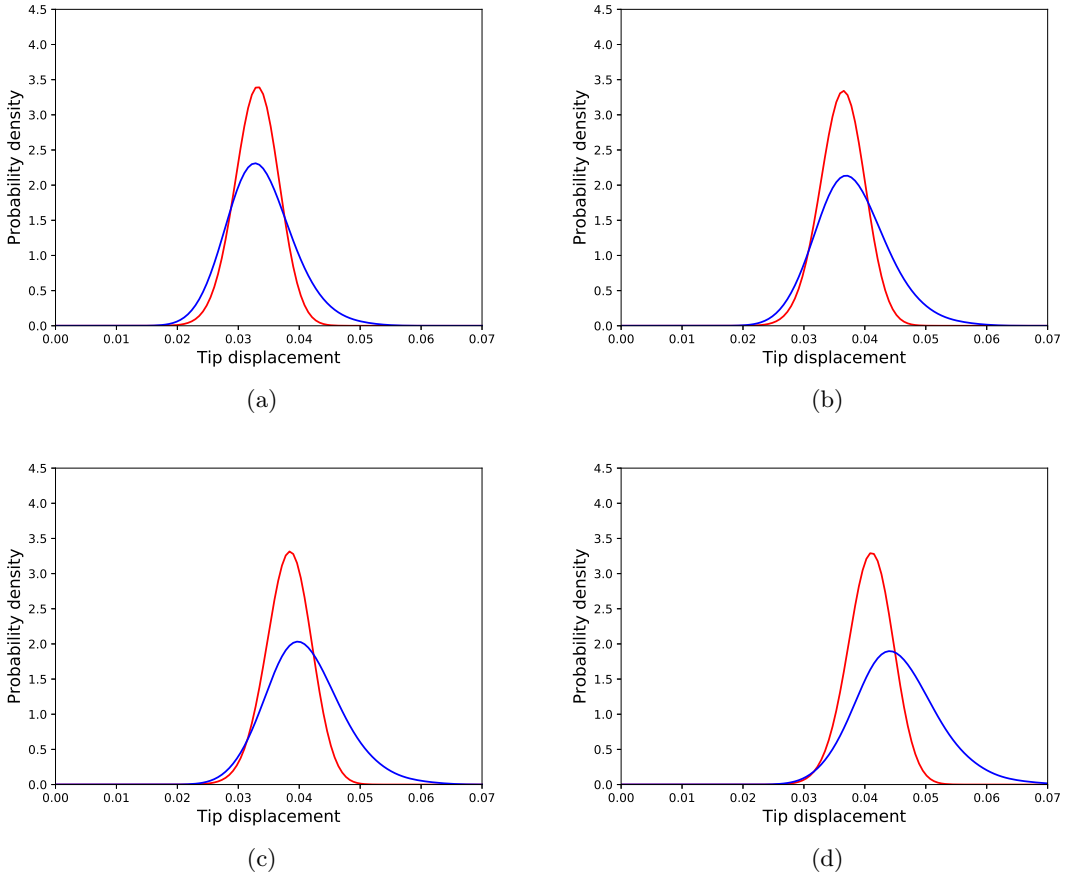


Figure 7.22: Distributions of tip displacements corresponding to Monte Carlo samples (blue), and generated samples by the cGAN model (red), regarding the 'extrapolation dataset' for the nonlinear problem for different load cases: (a) 320 load units, (b) 350 load units, (c) 370 load units, (d) 400 load units.

7.4 Summary

Three generative approaches to defining mirrors of structures were presented. The first was to use a physics-based generative model, such as an SFE model. As with most physics-based methods, the SFE model is prone to epistemic uncertainty, but in the absence of epistemic uncertainty, it outperforms every other method in providing accurate predictions about the system under study. The second method was a data-driven method based on the use of cGANs. The cGAN is a generative model which is able to learn transformations of distributions according to some varying known variables. The method is able to perform for both linear and nonlinear systems and is less prone to epistemic uncertainty, because of its purely-data-driven nature. Finally, a hybrid approach was presented which combined

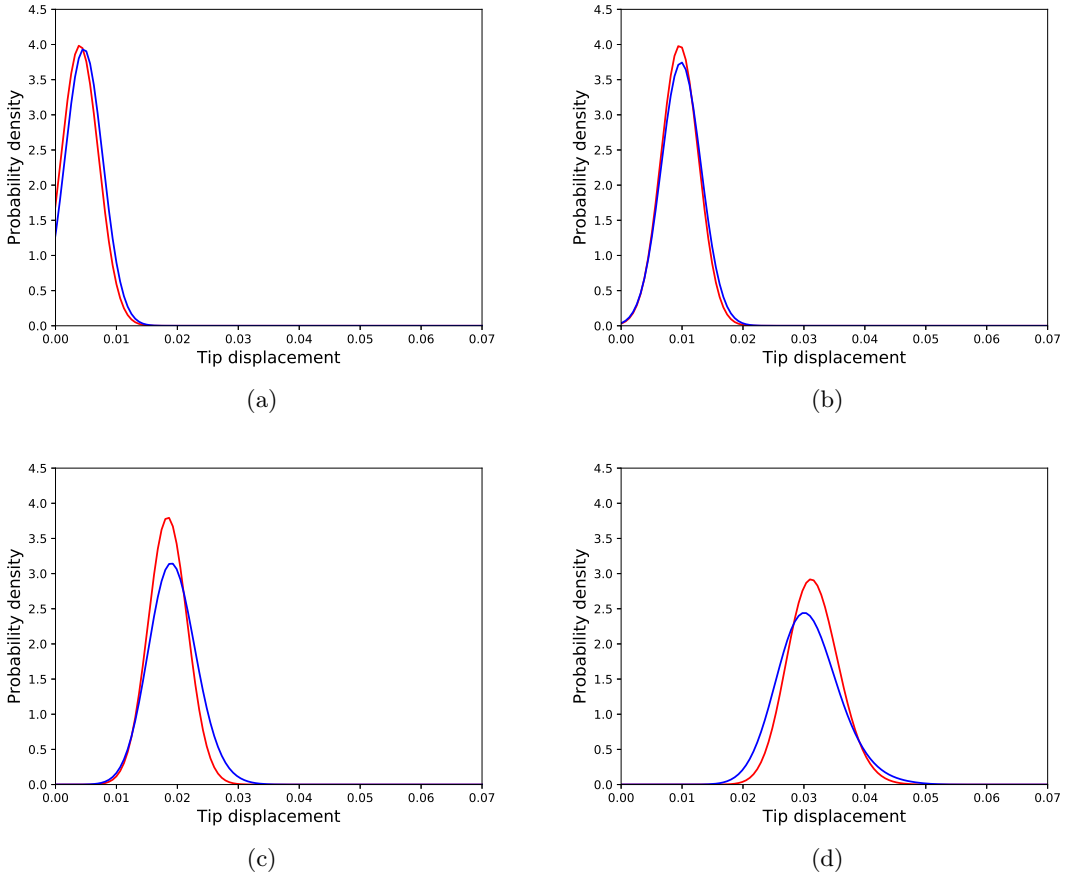
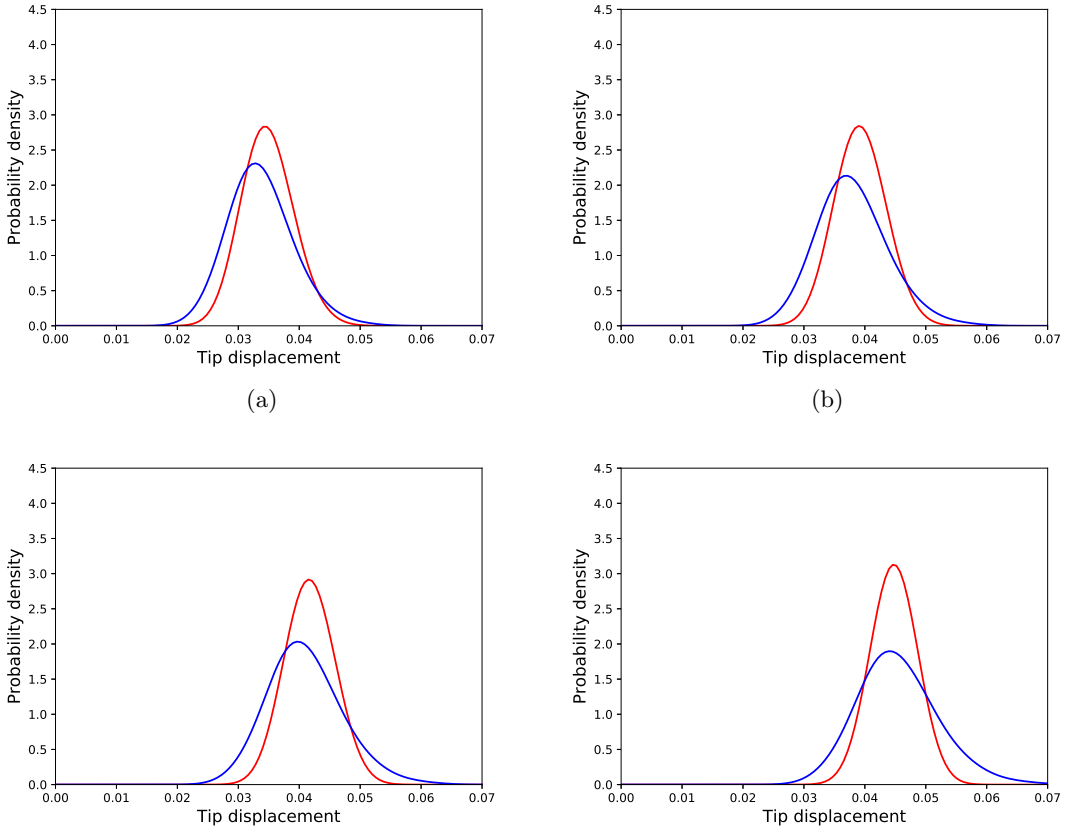


Figure 7.23: Distributions of tip displacements corresponding to Monte Carlo samples (blue), and generated samples by the hybrid model (red), regarding the nonlinear problem, trained according to the reduced dataset for different load cases: (a) 60 load units, (b) 120 load units, (c) 210 load units, (d) 300 load units.

the two methods aiming at exploiting the positive aspects of both. The combined model yielded results and accuracy slightly better than those of the black-box data-driven model. In addition, the hybrid model, was able to perform outside its training domain better than the purely data-driven model, a characteristic which might be inherited from its physics-based part.



(a) (b)

(c) (d)

Figure 7.24: Distributions of tip displacements corresponding to Monte Carlo samples (blue), and generated samples by the hybrid model (red), regarding the 'extrapolation dataset' for the nonlinear problem for different load cases: (a) 320 load units, (b) 350 load units, (c) 370 load units, (d) 400 load units.

Chapter 8

Generative adversarial networks for damage prognosis

8.1 Time-series generative adversarial networks

Generative adversarial networks have yielded quite good results so far for various purposes. Their initial goal was to generate images that look real and so convolutional neural networks were used. In the current work, they also proved able to generate data of vector type and to assist in nonlinear modal analysis. Naturally, the training scheme has been extended to algorithms to generate time-series. In [155, 156] the architecture of the GAN algorithm is directly applied using recurrent neural networks with a view to generating music. Moreover, in [157] another approach is presented, according to which, medical time-series are generated, dependent on additional inputs - similarly to conditional GANs (cGANs) [147]. For the purposes of the current work, TimeGANs [158], were chosen, since they are able to operate on arbitrary time-series data and they efficiently learn the temporal dynamics of the data rather than simply generating time-series that some discriminator will classify as real.

The goal of the algorithm, as already mentioned, is to generate artificial time-series that look real and, at the same time, to learn transition rules from data. It is also desired

that this procedure is made in a stochastic manner, since the next step in the time-series should be given via a probability density function, which is a function of the steps so far. The first condition is similar to the classic GAN condition of generating samples that look real, i.e. the probability density function of the artificial time-series $\hat{\boldsymbol{x}}_{1:T}$ (where $\boldsymbol{x}_{1:T}$ are the values of some feature vector $\boldsymbol{x}$ of the time-series from time-step 1 up to time-step T), should be approximately equal to the probability density function of the real time-series $\boldsymbol{x}_{1:T}$ ($\hat{p}(\hat{\boldsymbol{x}}_{1:T}) \approx p(\boldsymbol{x}_{1:T})$ or $\hat{p}(\boldsymbol{C}, \hat{\boldsymbol{x}}_{1:T}) \approx p(\boldsymbol{C}, \boldsymbol{x}_{1:T})$) if the generation is conditioned on some input variables $\boldsymbol{C}$). The second condition means that the algorithm should learn to approximate $p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})$, where $\boldsymbol{x}_t$ is the vector of values of the time-series at time t. Furthermore, if the time-series is conditioned on some parameters $\boldsymbol{C}$, which might be constant throughout time or might have different values in time, the probability to be approximated is $p(\boldsymbol{x}_t|\boldsymbol{C}_t, \boldsymbol{x}_{1:t-1})$. The two conditions are expressed as,

$$\min_{\hat{p}} D(p(\boldsymbol{C}, \boldsymbol{x}_{1:T})||\hat{p}(\boldsymbol{C}, \hat{\boldsymbol{x}}_{1:T})) \tag{8.1}$$

where D is some appropriate probability density distance measure, and,

$$\min_{\hat{p}} D(p(\boldsymbol{x}_t|\boldsymbol{C}_t, X_{1:t-1})||\hat{p}(\boldsymbol{x}_t|\boldsymbol{C}_t, X_{1:t-1})) \tag{8.2}$$

for every t.

The general framework that is followed in order to achieve a model that satisfies the aforementioned conditions is shown in Figure 8.1. The original time series $\boldsymbol{x}_{1:t}$ is mapped onto the corresponding time series $\boldsymbol{h}_t$ in the latent space $\mathcal{H}$ using an embedding function e, i.e. ,

$$\boldsymbol{h}_t = e(\boldsymbol{C}_t, \boldsymbol{h}_{1:t-1}, \boldsymbol{x}_t) \tag{8.3}$$

where $\boldsymbol{C}_t$ are the controlled input variables at the time-step t, $\boldsymbol{h}_{1:t-1}$ are the embeddings of the states of the previous time-steps ($\boldsymbol{x}_{1:t-1}$) and $\boldsymbol{x}_t$ is the state of the time-series at the current time-step. Similar to an autoencoder scheme [52], the encoded time-series are subsequently mapped back to the original space $\mathcal{X}$. This map is achieved using a function r which respects the temporal characteristics of the time-series $\boldsymbol{h}_{1:t}$, and yields

an approximation of the original time-series $\boldsymbol{x}_{1:t}$, i.e.,

$$\hat{\boldsymbol{x}}_{1:t} = r(\boldsymbol{h}_{1:t}) \tag{8.4}$$

where $\hat{\boldsymbol{x}}_{1:t}$ is the approximation of the reconstruction provided by r. In practice, both functions are modelled using recurrent neural networks [56], and more specifically *long short-term memory* (LSTM) neural networks [159], or *gated recurrent units* (GRU) networks [160], which take into account the temporal characteristics of the time-series during prediction.

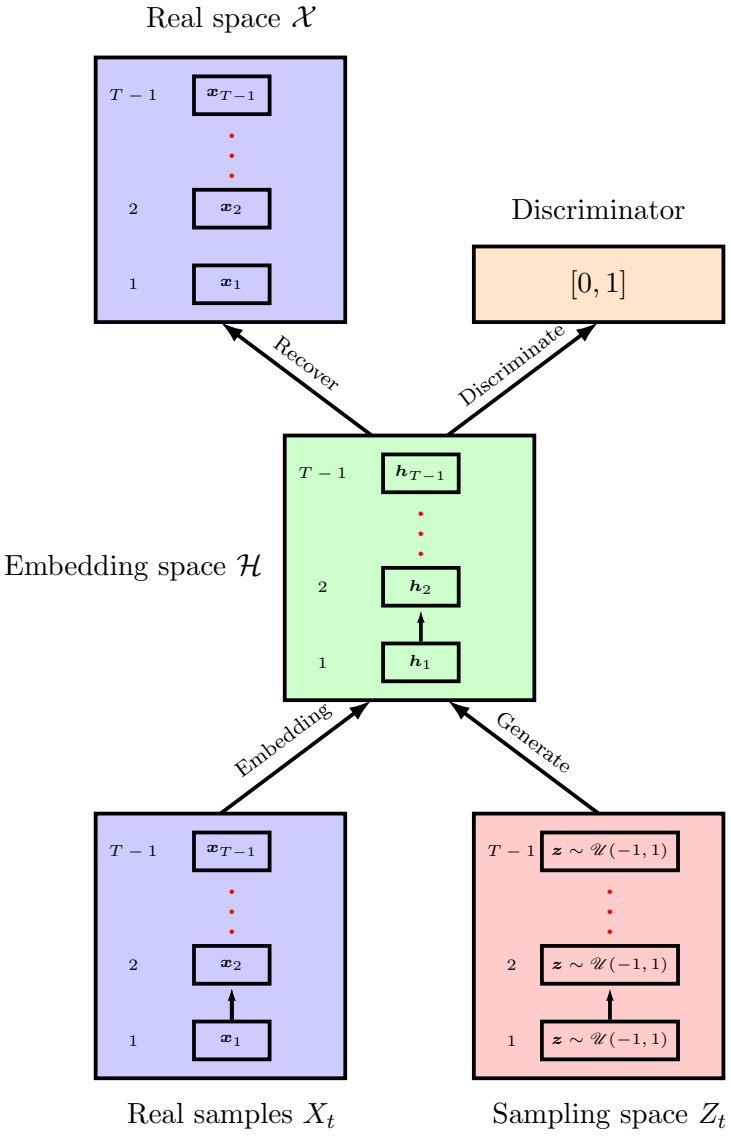


Figure 8.1: Schematic layout of a TimeGAN model.

The generator and the discriminator are also functions that are informed by the temporal characteristics of the time series. More specifically, the generator takes as input time series with noise variables $\boldsymbol{z}_t$ from some predefined distribution; often this is a uniform distribution in the interval $[-1, 1]$, $\boldsymbol{z}_t \sim \mathcal{U}(-1, 1)$. The generator g transforms the noise time-series samples into samples in the latent space $\mathcal{H}$, and should be informed by the previous embeddings, i.e.,

$$\hat{\boldsymbol{h}}_t = g(\boldsymbol{C}, \hat{\boldsymbol{h}}_{1:t-1}, \boldsymbol{z}_t) \tag{8.5}$$

where $\hat{\boldsymbol{h}}_t$ is the generated sample in the latent space. The discriminator, as in the classic GAN scheme, is trying to discriminate between artificial and real time series. In the current framework the discriminator acts in the embedding space $\mathcal{H}$, instead of the real space $\mathcal{X}$, where it acts in the original work [12]. The function d of the discriminator is given by,

$$y = d(\boldsymbol{C}, \boldsymbol{h}_{1:t}) \tag{8.6}$$

where y is the output label of the discriminator - zero for a fake time-series and unity for real. The discriminator is also a recurrent neural network, LSTM or GRU, with a single neuron output with a sigmoid activation in order to yield values in the interval $[0, 1]$

Various loss functions are used during training in order to achieve the multiple goals of the model. First, the *reconstruction loss* is used between the recovered data $\hat{\boldsymbol{x}}_{1:t}$ and the original time-series $\boldsymbol{x}_{1:t}$. The reconstruction loss $\mathcal{L}_R$ is given by,

$$\mathcal{L}_R = \mathbb{E}_{\boldsymbol{C}, \boldsymbol{x}_{1:T} \sim p}[\sum_t \|\boldsymbol{x_t} - \hat{\boldsymbol{x}}_t\|_2] \tag{8.7}$$

where $\mathbb{E}[\ ]$ is the mean value of the quantity within the brackets. The loss function is the mean-squared error (mse) loss function, used widely in regression schemes.

The second loss function refers to the adversarial training. It is a similar loss function to that of the GAN training, and is considered an *unsupervised loss*, since the minimal supervision during this type of training is indicating which samples are real and which are generated. The unsupervised loss $\mathcal{L}_U$ is given by,

$$\mathcal{L}_U = \mathbb{E}_{\boldsymbol{C}, \boldsymbol{x}_{1:T} \sim p}[\sum_n \log y_n] + \mathbb{E}_{\boldsymbol{C}, \boldsymbol{x}_{1:T} \sim \hat{p}}[\sum_n \log(1 - \hat{y}_n)] \tag{8.8}$$

Both terms are used during training the discriminator, and the minimisation of the loss function is sought; however when the generator is trained, the second term only is used, and the maximisation is attempted this time.

The last loss function used in the original TimeGAN framework is targeted to encouraging the generator to create time-series further based on the underlying physics of the data. This objective is achieved via training the generator as an autoregresive network, and is done by using embeddings of the real time-series as training data for the generator. The supervised loss function $\mathcal{L}_S$ is given by,

$$\mathcal{L}_S = \mathbb{E}_{\boldsymbol{C}, \boldsymbol{x}_{1:T} \sim p}[\sum_t \|\boldsymbol{h}_t - g(\boldsymbol{C}, \boldsymbol{h_{1:t-1}}, \boldsymbol{z_t})\|_2] \tag{8.9}$$

where each $\boldsymbol{h}_t$ comes from the real-sample time-series dataset. The algorithm's ability to generate time-series that comply with the rules of the underlying physics is based on the distinguishing process of the discriminator and on the ability of the LSTM or GRU networks to learn according to steps in the time-series in the past of the current time-step, in contrast to a Markov chain framework. In order to force the algorithm even further to learn according to the aforementioned physics, the last supervised loss function is introduced and the generator is further informed by the "rules" of the data.

In the current work, as embedding of time-series into the sampling space $\mathcal{Z}$ will be required, another loss function is used. This loss function trains a neural network k that finds the best possible sequence of noise $\boldsymbol{z}_{1:t}$ in order to generate a specific real sequence $\boldsymbol{x}_{1:t}$. The loss function of this $\mathcal{L}_e$ is given by,

$$\mathcal{L}_e = \mathbb{E}\,\|r(g(k(\boldsymbol{x}_{1:t}))) - \boldsymbol{z}_{1:t}\| \tag{8.10}$$

During minimisation of this loss function, only the network k that finds the best encoding $\hat{\boldsymbol{z}}_{1:t}$ for the real time-series $\boldsymbol{x}_{1:t}$ into the noise space (i.e. $\hat{\boldsymbol{z}}_{1:t} = k(\boldsymbol{x}_{1:t})$) is trained, and the trainable parameters of the generator and the recovery networks remain constant.

A TimeGAN model trained on time-histories of data, which are created according to some rules (most often the rules are defined as the underlying physics of the problem), can be exploited with a view to properly augmenting a dataset. The augmented dataset could, on

the one hand, assist in training more robust models and, on the other hand, provide more time-series samples in order to assist the user to further understand the processes that the data describe. The time-series could be some acceleration records from some structure or, as it is in the current work, the evolution in time of some feature of the structure that is affected by damage.

8.2 TimeGANs for damage prognosis

The problem of estimating the remaining useful life of some structure, can be formulated as a time-series prediction problem. Monitoring some quantity of interest that is sensitive to damage, one can use a model to predict the evolution of the quantity in time and infer the time that is needed until the structure reaches some state in which it definitely needs maintenance, repair or replacement. A major aspect of modelling the evolution of the damage sensitive feature, is that it should take into account uncertainty.

Uncertainty is an integral element in the procedure of damage evolution, as random events and random environmental conditions affect the development of damage, and in some cases are even the cause of it [161]. Classic machine learning approaches to the problem would involve training autoregressive models, such as LSTM neural networks, with a view to observing the current state of the structure and predicting the next one, and recurrently the next state. One could also try to train such an algorithm in order to directly infer the remaining useful life of the structure, or a probability distribution over it [162]. Although these approaches could yield satisfactory results, they are inherently deterministic. The output of the model is a single value for the next state and for the remaining useful life. In an attempt to capture uncertainty in such a model, one could feed the network with inputs representing the potential random events and environmental conditions. Then, a Monte Carlo-like simulation scheme could be followed and many potential outputs would be produced. This approach would require sufficient modelling of the uncertain conditions, which are sometimes even *unobservable* or *unknown*.

An approach to modelling such degradation processes of structural members is presented in [163], using a first-order Markov chain approach. Moreover, in the aforementioned

work, it is stated that the form of the noise process used, usually affects the performance of some damage-prognosis model. Using the TimeGAN model as proposed here, two major advantages are expected, compared to existing approaches. First, the model that transforms noise into observations is a neural network and the type of noise is not expected to largely affect the result. Second, since LSTM or GRU networks are used, instead of a first, second or higher-order Markov chain assumptions, the model is expected to better capture the physics of the underlying phenomenon evolution and be able to generate the required data more efficiently.

The framework chosen here is to define a Turing mirror for a structure [35], regarding the evolution of some damage-sensitive feature. As mentioned in [35], such a model shall be the oracle, which should be able to behave in such a way that some interrogator cannot distinguish whether it is the real structure or some model of it. The interrogator in the current framework is the discriminator, which at some point during training should not be able to distinguish between real and artificial time-series.

Within a PBSHM framework, the algorithm seems to be even more appealing in predicting the remaining useful life of structures. A requirement to train such a model is to have a dataset with time-histories of damage-sensitive features acquired from a structure until the point of its failure. Since it is difficult to acquire such a dataset, the proposed framework seems more fitting for a PBSHM application. Considering a homogeneous population of structures, the model can serve as a form [90] (i.e. a behaviour rule that the members of the population follow), in modelling the degradation process until failure of similar structures.

A TimeGAN model can be trained on an acquired dataset comprising time-histories of monitored features from a homogeneous population of structures, which have reached failure. Considering that the state of the structure, in terms of the monitored quantity, is recorded in equidistant time instants, the dataset would be of the form,

$$D = \{(S_i, \boldsymbol{x}^i_{1:T_i}) \quad i = 1, 2..., N,\ T_i \sim p_T\} \tag{8.11}$$

where S_i is the ith structure, N is the number of available time-histories, T_i is the time of failure of the structure and p_T is the probability density function of the lifetime of the

structures. Having trained the model so that the discriminator cannot identify whether a time-series is real or artificial, the model can then be exploited in order to define the remaining life T_c of a monitored structure S_c, currently in operation.

In the absence of any other model of the remaining life of some structure, one could exploit the probability density function (PDF), of the total life expectancy of the structures which have reached the end of their lifetime in the dataset, and assume that the testing structure will follow the same PDF, i.e. $T_c \sim p_T$. The PDF is a form that explains the behaviour of the population in terms of the life-time of the structures. Using this PDF and subtracting the time that the tested structure has lived so far, one gets a new PDF for the remaining lifetime of the structure of interest.

A less naive way would be to try and classify the structures within the population according to their external influence. Via the classification, a set of clusters is defined such as,

$$S_i \in C_j \quad i = 1, 2..., N,\ j = 1, 2...N_C \tag{8.12}$$

where S_i is the i^{th} monitored structure, C_j the j^{th} cluster and N_C the number of clusters. If clusters are defined in a proper way, structures within the same cluster are expected to behave in a similar way, and will have similar external influence. Therefore, PDFs of the total lifetime of the structures defined within the clusters are potentially more accurate than a global PDF of total lifetime for every structure. According to this assumption, one gets,

$$T_c \sim p_T^j \mid S_c \in C_j \tag{8.13}$$

where p_T^j is the PDF of the total lifetime of a structure in the j^{th} cluster. This approach should be more effective than the previous, but it still does not take into account the monitored features up until the present point in time of the tested structure.

A more sophisticated approach to take into account the state history of the tested structure, is to consider only structures that had similar state sequences. In order to do that, one would need to define some distance criterion between the time-series of the structures of the dataset and the testing structure, and filter out the data that do not meet some

defined-by-the-user tolerance, i.e.,

$$d(\boldsymbol{x}_{1:t_c}, \boldsymbol{x}^i_{1:t_c}) < \epsilon,\ i = 1, 2..., N \tag{8.14}$$

where ϵ is the tolerance and t_c is the current time point at which the structure is tested. Using only the time-series data that satisfy equation (8.14), a better estimation of the the remaining life PDF of the testing structure can be defined. The problem with such an approach is that not many time-series within the dataset will be close enough to the so-far-recorded time-series of the testing structure. This problem reveals a first advantage in training a timeGAN - that of augmenting the dataset by generating random time-series. Via such an augmentation, more samples will be available in order to define a PDF in the aforementioned manner more properly.

In order to exploit even further the Turing model that is defined by training the TimeGAN, the ability of the generator to generate potential states according to the already elapsed structural states is employed. According to the TimeGAN loss functions, the generator has been trained to generate time-series that the discriminator will misjudge as real, but also to respect the physics of the time-series ($\boldsymbol{h}_{1:t-1}$), and randomly generate the next step $\boldsymbol{h}_t$; as shown by equation (8.9). The information enclosed in the states time-history of the tested structure ($\boldsymbol{x}^e_{1:t_c}$ where t_c is the current time-step and the time elapsed from the deployment of the structure until the present) can be further exploited. At first, the latent code of the sampling space corresponding to the incomplete time-history, has to be calculated. The search of the aforementioned code can be performed in many ways, such as training an inverse model from the embedding space $\mathcal{H}$ to the sampling space Z. A simple way of doing this would be using a simple "hit and miss" search, that is performed by generating random noise series and considering the latent code corresponding to $\boldsymbol{x}^e_{1:t_c}$ to be $\boldsymbol{z}^e_{1:t_c}$ so that it is closest to the testing time series in terms of some distance metric similar to the one of equation (8.14), i.e.,

$$\boldsymbol{z}_{1:t_c} = \min_{\boldsymbol{z}_{1:t_c}} d(\boldsymbol{x}_{1:t_c}, r(g(\boldsymbol{z}_{1:t_c}))) \tag{8.15}$$

However, in the current work, an embedding network k has been introduced. Using k, one can get an approximation of the embedding of the time-series given by,

$$\boldsymbol{z}_{1:t_c} = k(\boldsymbol{x}_{1:t_c}) \tag{8.16}$$

The latent code $\boldsymbol{z}^c_{1:t_c}$ is then used in order to define the latent code of a set of generated time-series, up to point t_c in time. A set of latent codes $D_{\boldsymbol{z}}$ is generated according to,

$$D_{\boldsymbol{z}} = \{\boldsymbol{z}^k_{1:T_f} \mid \boldsymbol{z}^k_{1:t_c} = k(\boldsymbol{x}_{1:t_c}), \boldsymbol{z}^k_{t_i} \sim \mathcal{U}(-1,1) \; \forall \; t_i = t_c + 1, t_c + 2..., T_f\} \tag{8.17}$$

where t_c is the current time-step of the tested structure, and T_f is a number of time-steps large enough for the structure to have certainly reached its failure point. Feeding the latent codes from $D_{\boldsymbol{z}}$ to the generator, and afterwards the recovery network, a new set of time-series $D^g_{\boldsymbol{x}}$ is generated in the real space $\mathcal{X}$, such that,

$$D^g_{\boldsymbol{x}} = \{\boldsymbol{x}^k_{1:T_f} = r(g(\boldsymbol{z}^k_{1:T_f}))\} \tag{8.18}$$

Every time-series in the new set is close enough to the reference time-series $\boldsymbol{x}^e_{1:t_c}$ up to point t_c, and the remaining points are considered potential time-histories that the structure could follow until its failure. The time-series in $D^g_{\boldsymbol{x}}$ all have the exact same values up to point t_c, since their latent codes are the same up to that point and the generator, which is an recurrent model (with LSTM or GRU units), generates the next point in the series as a function of the previous points. From the point t_c onwards, the generated time-series will vary, since the latent codes $\boldsymbol{z}_{t_c:T_f}$ are randomly sampled.

By studying the set of generated time-series $D^g_{\boldsymbol{x}}$, a more properly-defined PDF of estimated lifetime of the structure can be extracted. As already mentioned, the time-series have enough steps that each one has at some point reached the state that the structure is considered not useful. By locating the time instant of failure T^g_i, for every generated time-series, an appropriate probability kernel can be fitted to the values, defining a PDF of the remaining lifetime of the structure. The last method is considered more effective and more robust than the previous methods, since it is applied specifically for a structure and its exact time-history up to the evaluation point. It is expected that the potential

random events of the environment of the structure so far, and in the future, as well as the stochastic damage-evolution procedure will be sufficiently encoded by the timeGAN's functionality of converting random noise series $\boldsymbol{z}$ into appropriate real-looking time-series.

8.3 Application of TimeGANs

The simulated dataset developed to evaluate the efficiency of the algorithm was acquired from a four-degree-of-freedom lumped-mass system like the one shown in Figure 8.2. The excitation F was Gaussian white noise with variance equal to 5. In order to simulate damage to the system, stiffness reductions were applied on springs 2 and 3. The damage-sensitive feature selected for monitoring was the *frequency response function* (FRF) of the second degree of freedom.

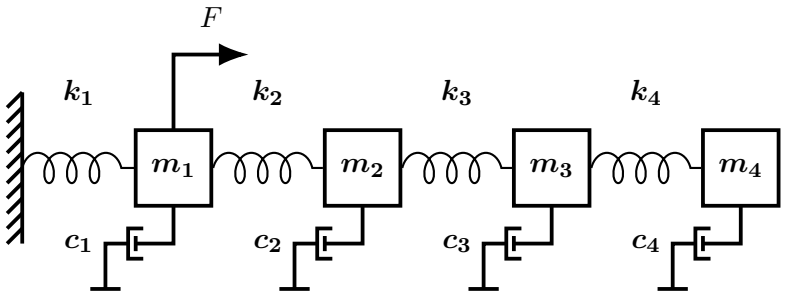


Figure 8.2: The mass-spring system used in the current application.

The damage reductions considered followed some rules induced, in order to define the underlying physics of the problem and evaluate whether the TimeGAN algorithm had learnt to generate time-series according to them. First of all, since the structure is expected to fit a homogeneous PBSHM framework, 1000 structures with the same layout were considered, but each one had stiffnesses k_2 and k_3 sampled independently from a Gaussian distribution with mean value equal to 6000 and variance equal to 120. For every structure, the state of failure was considered to be when the second natural frequency of the structures reached a specific value. The rules that defined the damage evolution at every time-step were that each structure had a nominal degradation step k_n, randomly sampled in the beginning of the simulations, in the interval $[42, 90]$. At every time-step, the value of k_2 of the degradation step was the sum of the nominal step and the absolute value of a random variable sampled from a normal distribution with mean equal to zero and standard

deviation equal to $0.1 \times k_n$, i.e.,

$$k_2^{t+1} = k_2^t - (k_n + \mathcal{N}(0, 0.1 \times k_n)) \tag{8.19}$$

Following these damage evolution rules, there are two parameters that should be learnt by the algorithm; the first is the nominal step, which is random, but can be estimated as the mean of previous steps, if some of them are available. The second parameter is the variance of the normal distribution in equation (8.19), which is inherently random and should be encoded by the timeGAN. The value of k_3 was also reduced at every step, but with a much lower nominal step, i.e. $0.1 \times k_n$. Samples of paths from the beginning of the lifetime of such structures until the point that they are considered unusable are shown in Figure 8.3. The result of such a procedure is that each structure begins from different stiffness values and fails when a particular limit is reached and in a varying number of time-steps. It becomes clear from the evolution process of damage that failure of different structures will be achieved in different time-steps for every structure. The maximum simulation time considered was 80 time-steps, and by the 80^{th} time-step, every structure had reached failure. In real-life applications each time-step should correspond to a predefined time interval.

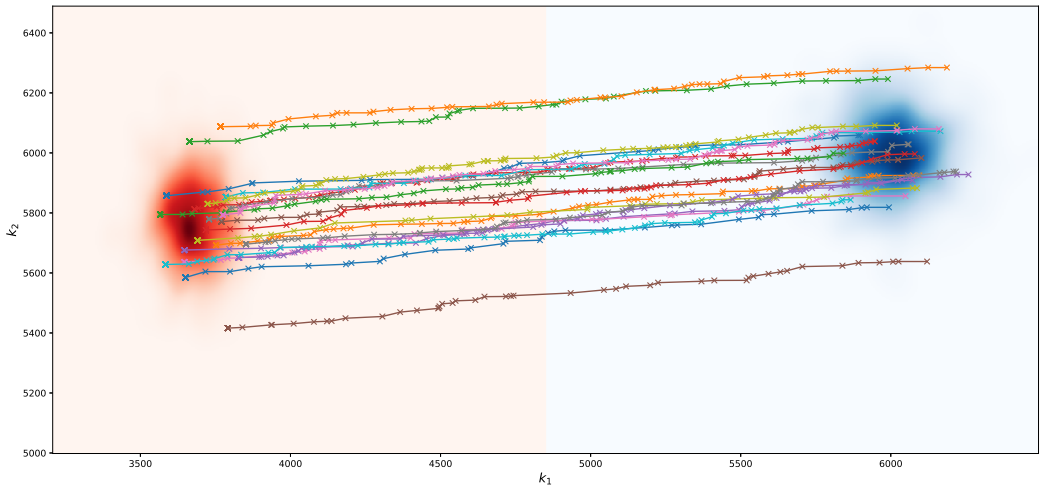
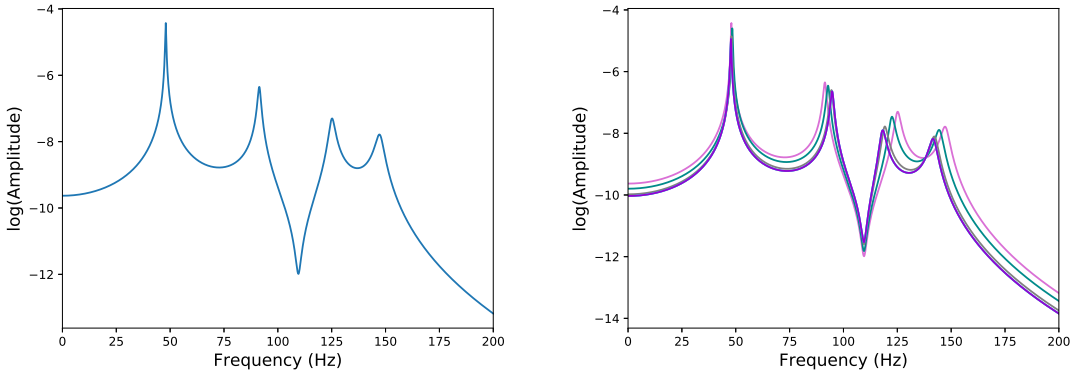


Figure 8.3: Samples of paths from initial state (blue distribution) until total failure (red distribution).

One of the recorded FRFs is shown on the left of Figure 8.4, and next to it, the effect of gradual stiffness reduction is shown as a gradual colour transition from pink to purple. To visualise more effectively the whole dataset, *principal component analysis* (PCA) [51] was performed on the FRFs, and some paths for different structures from the time of deployment until their failure are shown in Figure 8.5.



(a) (b)

Figure 8.4: FRF sample at the beginning of the simulations (left), and FRFs with increasing damage (right), from low damage (pink), to higher stiffness reduction (purple).

The TimeGAN model was trained on the dataset described in the previous section. The neural networks involved in the model had all *hyberbolic tangent* (tanh) activation functions. All LSTM layers had 128 neurons in their neural networks and every model had one layer of LSTM recurrent neural networks and a fully-connected feedforward layer. The sizes of the output-layer neurons of every network are defined by the size of the space into which their outputs belong. The sampling space Z was chosen to be a three-dimensional space, the embedding space was 128-dimensional, and the real space was three-dimensional, since the PCA components of the FRFs were used in order to reduce the dimensionality of the samples. Using three principal components, 96% of the variance of the data is explained. Training was performed using an Adam optimiser [119].

Being trained for several epochs, the model can be considered the oracle defined in the Turing mirror framework. Since a universal framework to evaluate the quality of training of GANs does not exist, for the current work, the distribution of the lifetimes of the real structures and of the generated time-series is compared. In Figure 8.6, a comparison of the two distributions is shown. The probability density functions were calculated using

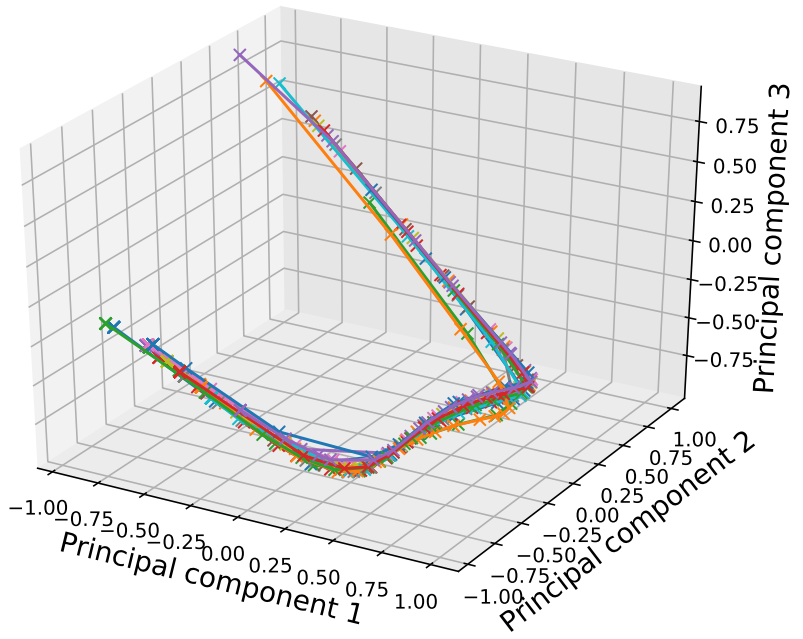


Figure 8.5: Principal components of paths from deployment until the failure of structures within the population; different colours corespond to different structures.

Silverman's algorithm [45]. The distributions seem close and the KL divergence [148], calculated between them was 0.25, which for the purposes of the current work was considered low enough. Therefore, instead of interrogating the available data from the population, the oracle can be interrogated.

The point that the recorded and the generated time-series reach failure was defined by monitoring the second natural frequency of the structures; the damage limit was taken equal to 119Hz herein. It is a simple criterion, and definitely, more sophisticated approaches could be used (like training an recurrent neural network in order to separate failed from not-failed time-series). However, it is a simple and interpretable criterion that could also be easily implemented in real-life structures.

Another example to illustrate how the algorithm yields more certain results when more data are available from the testing structure, is shown in Figure 8.7. Every PDF in the

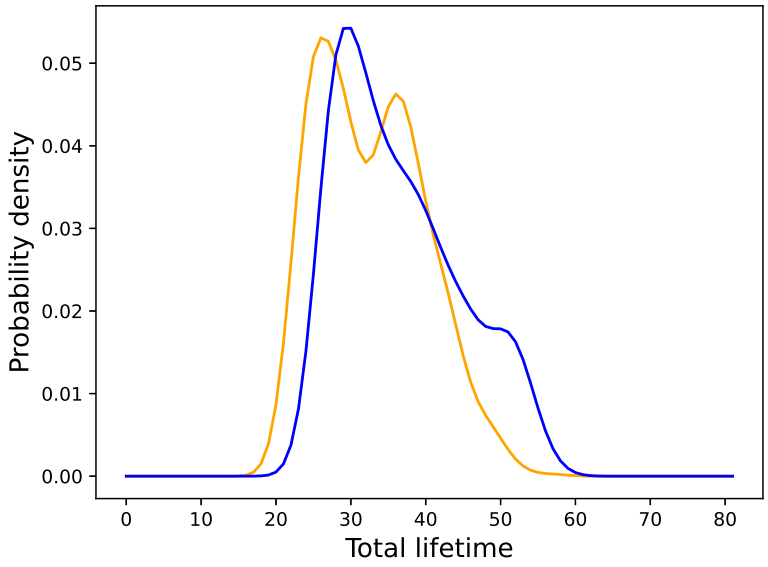


Figure 8.6: Probability density function of the total lifetime of the real samples (blue) and of the artificial samples (orange).

aforementioned figure is created using the described procedure. Every time one more time-step of observed data was added in the available time-series of the tested structure and the procedure was used to define the remaining time until failure. The first PDF corresponds to five recorded time-steps and the last to 30. The real total lifetime is shown with a red line and is equal to 37 time-steps.

For comparison, a PDF of the total lifetime of all the structures in the population is shown in Figure 8.6. If one would follow the naive approach, this PDF should have been used in order to define the remaining lifetime of the tested structure. To compare the naive approach with the TimeGAN algorithm, a testing population of structures was simulated following the same rules with the simulations of the training dataset. Having collected the time-histories until failure of all the testing structures, a random time-step was chosen for each one to be the time that it is being tested. The metric $\mathcal{P}$ used to evaluate the two approaches is the total probability assigned by each approach given by,

$$\mathcal{P} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} p(T_i) \tag{8.20}$$

where p is the PDF defined either by the TimeGAN algorithm or by the available dataset,

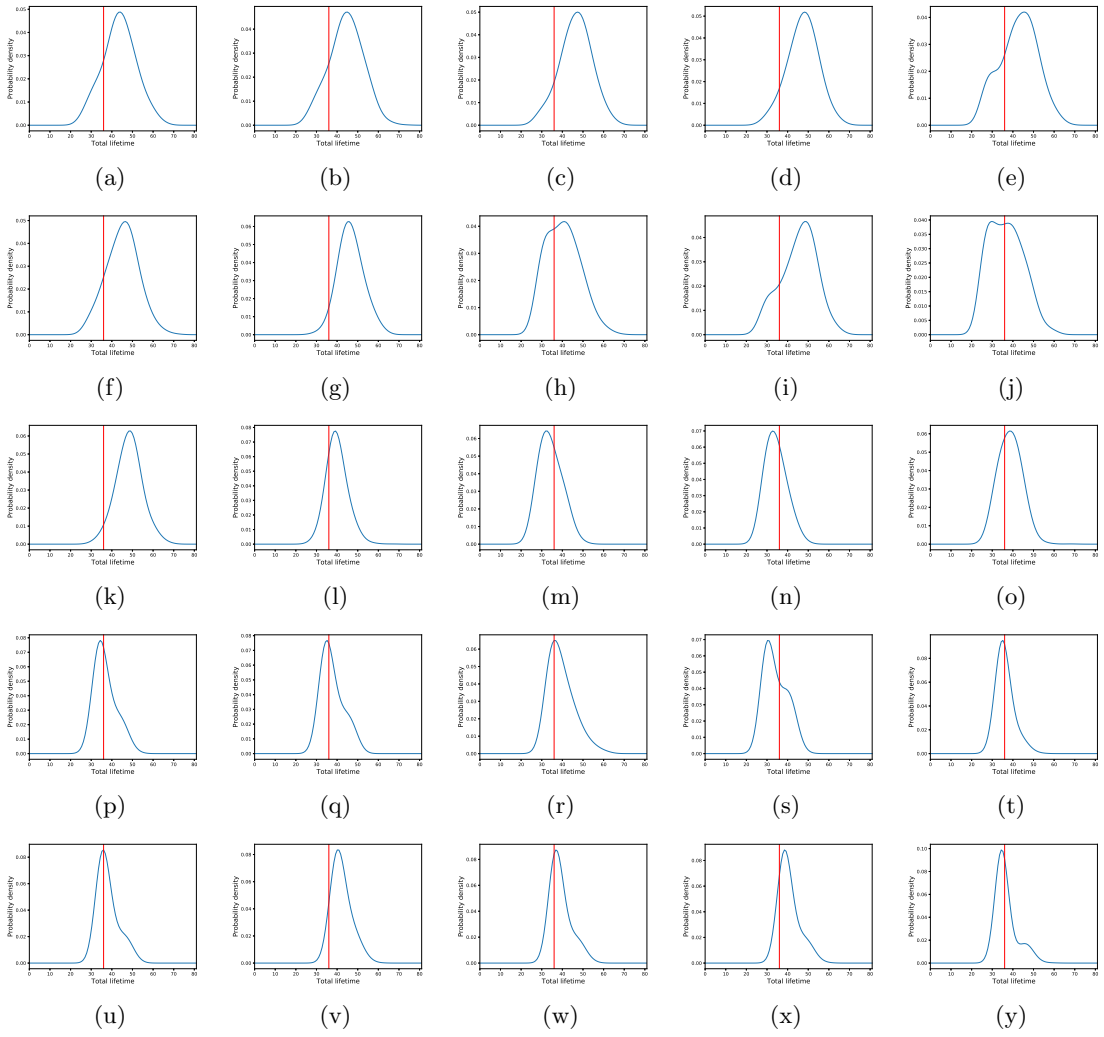


Figure 8.7: Real life-time of tested structure (red vertical line) and evolution of predicted total life-time PDF using data from one more time-step in every figure; from left to right and from top to bottom.

n_{test} is the number of structures in the testing dataset and T_i is the total lifetime of the i^{th} structure. The higher the value of the metric, the better the model, since this means that it assigns higher probability to the real value of total lifetime that is sought to be approximated. The TimeGAN model yielded $\mathcal{P} = 0.058$, and the naive approach $\mathcal{P} = 0.037$. Using the total probability, given by equation (8.20) one could even perform optimisation of hyperparameters of the method, for example of the bandwidth of the kernel density function used to define the probability density functions of the TimeGAN method, but such a procedure would be computationally inefficient.

8.4 Summary

In the current chapter, a generative method for definition of the remaining useful life of structures is presented. The method is based on the TimeGAN algorithm, which is used to generate artificial timeseries based on an available dataset. The algorithm is built in such a way, that it attempts to learn the physics of the available timeseries and is able to generate new ones in a generative/stochastic manner. The algorithm is trained using timeseries data of some damage-sensitive feature of structures within the population, from initiation of damage until failure. To provide predictions about the remaining lifetime of a testing structure, the damage-sensitive feature of the new structure is monitored and used as an input timeseries into the algorithm. The algorithm then generates potential outcome trajectories of the feature, each trajectory reaching some predefined failure criterion at a different time. Using the collection of different times that the generated trajectories reach the failure criterion, a probability density function of the remaining useful life can be extracted. The method proves to work efficiently for a simulated population of structures and provides more accurate predictions than extracting a probability density function over the total lifetime of the structures of the population and considering that all structures should follow the same PDF regardless of the data acquired from them.

Chapter 9

Conclusions and next steps

9.1 Machine learning methods for digital twins

A selection of machine learning methods were presented in the current work. Their common theme, was the modelling of specific aspects of structures. All such models offer a data-driven approach to solving problems that might arise during deployment of digital twins. Their data-driven nature has certain advantages over corresponding physics-based methods, but probably also some disadvantages. What is clear though, is that such models can be integrated into the digital twin and offer modelling solutions to occasional problems.

In Chapter 4 an ontology about SHM was presented. The ontology is an attempt to connect different methods, datasets, physical parts and knowledge in general, that play some role within an SHM project. The ontology proves useful for sharing knowledge. It might also be used as a database and a guide to develop software relevant to SHM. Methods like the ones presented in the rest of the chapters can potentially be included in a set of models considered to be a digital twin of a structure. The models have different functionalities, which are useful for structural modelling and monitoring in different ways. In order to organise many different methods and clearly define their purpose, functionality and interaction with entities of the digital twin, a similar ontology like the one presented in the third chapter could be used. Such a framework may prove useful especially for

multidisciplinary projects and to provide a comprehensive list of the capabilities of the digital twin.

In Chapter 5, a data-driven machine-learning-motivated method to perform modal analysis for nonlinear structures is presented. The method proves to be efficient in decomposing the movement of structures with nonlinearities, into components with certain desired characteristics of modal analysis. The components have PSDs with single dominant modes, which means that each component dominates a different range of frequencies of the movement of the structure. Moreover, the components in most cases were statistically independent, meaning that inferring the value of a component from observations of the rest of the components is impossible. Finally, the presented algorithm offers an inverse mapping from the modal space to the physical space, something equivalent to nonlinear superposition of the modes, which is essential for modal analysis.

Modal analysis offers an extremely useful decomposition for the purposes of structural analysis. Decompositions are a type of model, often used for the purposes of better understanding of the data by the users. Linear modal analysis provides the users with simple single degree-of-freedom systems to study, instead of a large dynamic system. Such a decomposition is not trivial for nonlinear systems. Via the use of the method described in Chapter 5, a completely data-driven decomposition is achieved. The decomposition is efficient in many cases, and may be invariant of the type of nonlinearity. Integration of such a method in a digital twin framework, could allow better analysis of the monitored structure, as well as better understanding of the underlying physics of the structure.

In Chapter 6, a PBSHM framework for inferring the normal-condition characteristics of structures within a heterogeneous population is presented. The framework is based on the idea of fibre bundles. The potential structural states of members of the structure population are considered to be the fibres of the bundle. The points of the base manifold represent the structures of the population. The task of inferring the normal conditions of the structures, reduces to navigation through the manifold and calculation of the normal-condition cross-section of the fibre for every point. A major problem, is that structures within a diverse population are not easily embeddable on a manifold. To bypass this issue, the graph neural network algorithm is used in combination with a graph representation of structures. As a result, the graphs representing structures are used as inputs to the

inference algorithm. An application of the algorithm on a simulated population of truss structures proves the potential of the algorithm to predict normal condition characteristics of the structures quite efficiently.

In Chapter 7, two ways of using the cGAN for modelling structures are presented. The first refers to generating artificial data for structures under varying environmental conditions. The method focusses on generation of data for temperatures for which data have not been acquired. Moreover, the algorithm performs its job even under the influence of unknown and unmeasured parameters - in the current case, humidity. Digital twins rely largely on data, therefore, missing data may reduce their accuracy. An algorithm like this can be exploited to complete a dataset with missing data, and provide data for appropriate training of algorithms, such as novelty detection algorithms.

The second algorithm presented in the same chapter is about using cGANs as the basis for a digital-twin generative model. The algorithm, being able to generate distributions of data according to some controlled input variables, is a great candidate for such a model. It is a completely data-driven model, and it is shown that the cGAN outperforms a physics-based generative model (SFEM), in the presence of epistemic uncertainty in the latter. The algorithm is able to learn efficiently, transformations of the distributions of the quantities of interest, according to some known input variable, and also incorporates the effect of underlying uncertain variables on the distributions. Furthermore, a combination of the data-driven and the physics-based algorithms is presented - a hybrid model - which performs slightly better than the data-driven algorithm. Such algorithms are often preferred, because they are partially driven by the physics of the problem, allowing them sometimes to even perform outside the domain of the training data.

In Chapter 8 an algorithm based on GANs is presented, which is used to perform damage prognosis in a population-based SHM framework. The algorithm is purely data-driven, in the sense that the physics of the damage mechanism are not taken into account in the modelling procedure. The method is based on learning about damage evolution of structures within a population, in order to perform damage prognosis for different structures of the population. Given observations of some damage-sensitive feature of the structure, potential time-series of the feature, until failure of the structure (or to some predefined point) are generated. Each time-series reaches the final point at a different time and,

considering all the time-series, one is able to define a probability density function on the remaining useful life of the structure. The efficiency of the algorithm is evaluated via a simulated example of damage that could reflect degradation of some material.

9.2 Further work

The methods presented were applied mainly to simulated data. Their further validation is certainly desired by applying them to experimental data, or data from deployed structures in operation. Each method can be useful in different situations, and they can all together be included in a digital-twin framework. Therefore, an appropriate next step would be to combine the discussed methods with existing ones, aiming at defining an integrated digital twin of a structure. As a first step the structure could be in a laboratory and appropriately monitored and tested.

Obviously, the primary buisiness of a digital twin is maximisation of profit. By increasing the modelling capabilities of a digital twin, it is expected to also increase the efficiency of its operation by reducing costs and better handling resources. The proposed methods may solve existing problems of modelling structures, but a further analysis regarding their benefit in structural operation should be performed.

The nonlinear modal analysis method presented in Chapter 5 may allow the application of methods, which are based on linear modal analysis, on nonlinear structures. An example of application of linear modal analysis, is that of ground vibration testing for aircraft [164], which fail if the structure exhibits nonlinear behaviour. The *modal assurance criterion* (MAC) [11], is also a method used for SHM or for calibration of models, which is mainly applied on linear structures and could be enhanced using the nonlinear approach presented herein. The proposed method should be checked on such applications, and its contribution to their performance should be evaluated.

The population-based aspect of digital twins could prove quite useful. Since lack of data is a common problem in modelling of structures, the population-based scheme presented in Chapter 6, could solve major issues of definition of digital twins. Similar to sharing knowledge to perform SHM between populations of structures, one could follow a similar

framework to define digital twins of structures. Digital twins of different structures could share knowledge in order to maximise their accuracy. A digital twin of a structure should have captured part of the physics of the structure and transferring this knowledge to other structures could increase the modelling capabilities even further.

Extending the idea of population-based digital twins even further, one could try to explicitly exploit digital-twin models to built digital twins of newly-observed systems. The idea of transferring models between systems could give the opportunity to accurately model structures from which data have not been acquired. By following such an approach, established digital twins of structures can unlock the potential definition of digital twins for new structures or structures, that have not been deployed yet. Similarly to the strategy followed by engineers for many years, models can be built for structural members, which are used in different structural systems, according to their characteristics.

Appendix A: Toplogy, Manifolds and Vectors: Basic Definitions

The idea of a manifold begins with the concept of a *topological space*, which is essentially a space with a notion of continuity. One also needs the idea of a *homeomorphism*, which is essentially a continuous map with a continuous inverse. Homeomorphisms between topological spaces exist if they are essentially the same in terms of topology e.g. have the same number of holes etc. Many topological spaces are not homeomorphic – i.e. topologically equivalent – to flat Euclidean spaces $\mathbb{R}^n$. This inequivalence presents a problem in physics and engineering because one usually wants to go beyond notions of continuity and do calculus i.e. one needs *differentiability*. However, it is only clear how to do calculus in flat spaces; this would not be a problem if the space of interest, X, was globally homeomorphic to some $\mathbb{R}^n$, one would then map the calculus problem into the $\mathbb{R}^n$, solve it (if possible), and map back. However, many problems of interest will be on curved spaces – like the sphere. However, if one is concerned with differentiation, e.g. the problem is to solve some differential equation, one can exploit the fact that differentiation is a local operation. This observation means that one only needs to map some local region of the space of interest into $\mathbb{R}^n$, solve the problem and map back – one only needs *local homeomorphism*. However, one may want a solution that covers X, and that means that many local homeomorphisms may be needed; furthermore, if one wishes a well-behaved solution to the problem on X, the solutions on different regions will need to knit together in some appropriate manner. Generally speaking, if one has 'solutions' on two overlapping regions of X, the solutions should agree on the region of overlap. Despite the rather woolly

nature of this discussion, it does motivate the notion of a *differentiable manifold*.

The basic theory of manifolds is explained well in [124]. From a mathematical point of view, one of the classic texts on differentiable manifolds is [165]; a discussion at a much more sophisticated (and modern) level can be found in [126], which also covers applications in gauge theories, which are mentioned later in this appendix.

One begins with a topological space X, and assumes that it is equipped with a family of open sets $\{U_i\}$ such that any point $x \in X$ also satisfies $x \in U_i$ for for at least one i.

A *chart* on X is a pair of one of the U_i, together with a homeomorphism ψ_i which carries U_i onto an open set $\psi_i(U_i) = V_i$ in $\mathbb{R}^n$. The homeomorphism condition ensures that there is an inverse $\psi_i^{-1}(V_i) = U_i$. Thus, $\psi_i : U_i \longrightarrow \mathbb{R}^n$ and $\psi_i^{-1} : \mathbb{R}^n \longrightarrow U_i$. One can regard $\psi_i(x) = (x_1, \ldots, x_n) \in \mathbb{R}^n$ as providing a *coordinate system* on U_i.

If there is a map ψ_i for every U_i in the open cover of X, then one has coordinates for any point in X, and the set $\{U_i, \psi_i\}$ is referred to as an *atlas* for X. This construction establishes the first thing needed, a set of local homeomorphisms from X into a flat $\mathbb{R}^n$, that covers X. One now needs the condition that the maps interact sensibly on the regions of overlap of the charts, i.e. $U_i \cap U_j$.

Suppose that some point $x \in U_i \cap U_j$. As U_i and U_j are both open, so then is $U_i \cap U_j$ (X is a topological space). Furthermore, both ψ_i and ψ_j restrict to homeomorphisms on $U_i \cap U_j$. These facts mean that both $\psi_{ij} = \psi_i \circ \psi_j^{-1}$ and $\psi_{ji} = \psi_j \circ \psi_i^{-1}$ are both flat-space homeomorphisms i.e. $\psi_{ij} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$. Thus both ψ_{ij} and ψ_{ji} are subject to normal flat-space calculus rules.

So, staying on $U_i \cap U_j = U_{ij}$, one can denote the coordinate system induced by ψ_i as $\psi_i(x \in U_{ij}) = \{x\} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. (throughout this appendix, curved brackets will denote vectors, while square brackets will denote matrices.) If one similarly denotes the coordinate system induced by ψ_j by $\psi_j(x \in U_{ij}) = \{x'\} = (x'_1, \ldots, x'_n) \in \mathbb{R}^n$, it is clear that $\psi_{ji}(\{x\}) = \{x'\}(\{x\}) = (x'_1(\{x\}), \ldots, x'_n(\{x\}))$ is nothing more than a *change of coordinates* on U_{ij} (Figure 1).

Now, the only conditions ensured on the maps so far (because they are homeomorphisms) is that they are continuous. However, because they are maps/functions on flat space, it

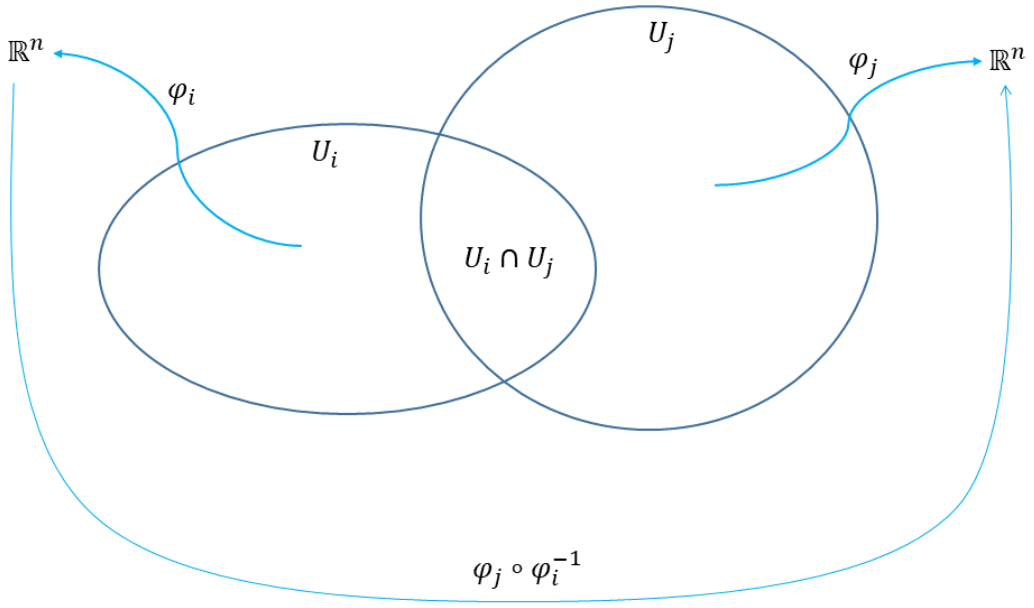


Figure 1: Coordinate mappings on open sets of a manifold, and the corresponding 'change of coordinates' on the region of overlap.

is straightforward to determine their level of *differentiability*. The maps are referred to as C^m if they are m-times differentiable under the conditions of normal flat-space calculus. The charts are *compatible* if all the maps ψ_{ij} are C^m, and one can then say that the atlas is C^m, and gives X the structure of a C^m *differentiable manifold*. Some special cases are:

- C^0 denotes a *continuous* manifold, so no real extra structure over the topology.
- C^1 is a differentiable manifold.
- C^∞ indicates an infinitely-differentiable manifold.
- C^ω denotes an *analytic* manifold (C^∞ and all Taylor series converge).

It is now necessary to define vectors on/in manifolds. If one thinks of a vector at a point as an actual straight line segment, pointing in the direction of interest, with a certain magnitude, curved manifolds present a problem; the line would leave the manifold. Only an infinitesimal line would stay in the manifold, and in the limit of zero length, would clearly be tangent to the manifold at the point of interest. In order to define vectors at a point more generally, one assumes that a vector space is attached to the manifold at

the point of interest; this is then where vectors 'live', and the space is called the *tangent space*. At a point x in a manifold M, the tangent space is denoted $T_x(M)$.

In order to do any sensible physics or engineering, one actually needs the idea of a *vector field*. Suppose that the manifold of interest M is spacetime; in order to look at fluid mechanics for example, one needs a way of defining vectors at all points within the flow field. Furthermore, if one is going to use calculus on the vector fields, it will be necessary to impose ideas of continuity etc. of the fields as one moves between points on M. What is needed is a means of collecting together the tangent spaces of all points in M in such a way that one can move between the tangent spaces smoothly, as one moves on the manifold M. Gluing the tangent spaces together leads to the idea of the *tangent bundle* $T(M)$; however, one can make the construction more general, and this leads to the concept of the *fibre bundle*.

Appendix B: Stochastic finite elements method

Finite element method (FEM) models [7] have been a very powerful and useful tool to numerically solve differential equations which describe mechanical systems. FEM transforms a continuous problem and a continuous differential equation into a discrete system of equations. Solutions are calculated only for a discrete number of points. Solutions for intermediate points are calculated using interpolation functions called shape functions. The continuous static differential equation most commonly used in FE models is given by,

$$\int_V \sigma \epsilon dV = \int_V f^V dV + \int_{S_e} f^S dS \tag{1}$$

where the LHS is the internal potential energy of a body, where σ is stress, ϵ is strain and V is the volume of the body of interest, the RHS is the potential work of the forces applied on the body, which is the integral over the volume of all the volume forces (f^V), plus the integral over the surface of the surface forces (f^S). Using a finite element formulation and minimising the total potential energy or the difference between the two sides of the equation, one gets,

$$[K]\{U\} = \{F\} \tag{2}$$

where $[K]$ is the stiffness matrix of the structure for a specific meshing scheme applied, $\{U\}$ is the displacement vector of the nodal displacements and $\{F\}$ is the equivalent nodal force vector to the total applied forces on the body.

For dynamic problems, inertia forces are introduced into the RHS of equation (1) resulting in,

$$\int_V \sigma \epsilon dV = \int_V f^V dV + \int_{S_e} f^S dS - \int_V \rho \ddot{u} dV - \int_V c\dot{u} dV \tag{3}$$

where ρ refers to the mass density function of the body, $\ddot{u}$ is the acceleration, c is the damping parameter and $\dot{u}$ is the velocity at every point. This equation holds for every time instant t of the simulation. Once again, following the FEM formulation and defining a discretisation of the body, the system of equations are,

$$[M]\{\ddot{U}\}(t) + [C]\{\dot{U}\}(t) + [K]\{U\}(t) = \{F\}(t) \tag{4}$$

where $[M]$, $[C]$, $[K]$ are the mass, damping and stiffness matrices and $\{\ddot{U}\}$, $\{\dot{U}\}$, $\{U\}$ the nodal accelerations, velocities and displacement vectors respectively.

The matrices in equations (2) and (4) are often calculated assuming deterministic structural parameters, e.g. Young's modulus (E), Poisson's ration (ν), mass density (ρ) etc. However, these parameters are quite often *not* deterministic. Especially in composites, such parameters are almost certainly spatially random. Young's modulus might vary within the volume of the body one tries to analyse using FEM. These variations are not just discrete variables, almost certainly they are *stochastic processes* [166]. A stochastic process has a correlation function that defines how values over some distance (spatial or temporal) are correlated. Furthermore, every point has a mean value and a variance defined by functions $\mu(x)$ and $\sigma^2(x)$ respectively. If the two functions are constant everywhere over the space where the process is defined, then it is called a *stationary* process.

An example of a deterministic consideration of Young's modulus and a stochastic Young's modulus for a cantilever beam problem would look like the functions shown in Figure 2. The black line represents how conventional FEM is applied, assuming a constant and deterministic value for structural parameters, while the red and blue lines show samples drawn from a stochastic process. A way to address problems like this, in general, would be to sample from the stochastic process and follow a Monte Carlo scheme. This approach would require solving a large number of deterministic FEM problems, performing a sensitivity analysis and post-processing the results in order to infer the statistics of the quantities of interest. In this case, the quantity of interest could be the displacement of

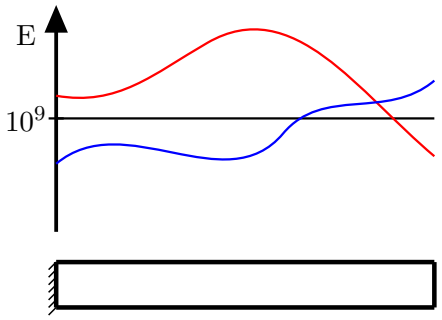


Figure 2: Cantilever beam with constant Young's modulus (black line) or spatially varying (red and blue lines).

the beam tip and a probability distribution would be defined over the potential values of this displacement.

The SFEM already mentioned, is a quite popular means of propagating uncertainty from material properties and randomness in the excitation forces into the response characteristics of a structure. In contrast to a Monte Carlo approach, SFEM infers the distribution of interest as a function of a set of discrete normally-distributed variables. In order to explain the SFEM formulation of a problem, first a method to decompose the random field is needed. The expansion method used herein is the Karhunen-Loève expansion [151, 167].

Karhunen-Loève (KL) expansion

The KL expansion is based on the spectral (i.e. eigenvalue) decomposition of the autocovariance function of the given random field. Given a random field $H(x)$ and its autocovariance function $C_{HH}(x, x')$, any realisation of the field $H(x)$ is expanded over a basis of deterministic functions, defined by the eigenvalue problem [151],

$$\int_{\Omega} C_{HH}(x, x')\phi(x')d\Omega_{x'} = \lambda_i \phi_i(x) \tag{5}$$

where the kernel $C_{HH}(x, x')$ is a kernel autocovariance function; bounded, symmetric and positive definite and $\Omega_{x'}$ is the total space of x'. The set of eigenvalues λ_i and eigenfunctions/eigenvectors $\{\phi_i\}$ form a complete basis to express every realisation of the field as,

$$H(x,\theta) = \mu(x) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\xi_i(\theta)\phi_i(x) \tag{6}$$

where $\xi_i(\theta)$ are the coordinates of the realisation, which are independent random variables and θ is the random event.

In practice, if one wishes to define an expansion of a random field and either generate random realisations or use it, as it will subsequently be used, for the purposes of stochastic FEM, a truncation is performed at mth order yielding,

$$H(x,\theta) = \mu(x) + \sum_{i=1}^{m} \sqrt{\lambda_i}\xi_i(\theta)\phi_i(x) \tag{7}$$

where it is assumed that the eigenvalues λ_i are sorted in ascending order.

Solution of static SFEM problems

After defining the stochastic field in the finite-element formulation of a problem, this field is expressed via the KL expansion. This approach leads to the stiffness matrix of equation (2) appearing as a summation over stiffness matrices constructed according to the eigenfunctions of equation (7); more specifically,

$$K(\theta) = K_0 + \sum_{i=1}^{m} \xi_i(\theta)K_i \tag{8}$$

where the $K_i s$ are deterministic matrices that are calculated using the eigenfunctions ϕ_i from equation (7). Realisations from the set ξ_i can be used in order to generate realisations for the stiffness matrix $K(\theta)$, in the case that a Monte Carlo simulation is to be followed.

Now, substituting equation (8) into equation (2) yields (assuming a deterministic load),

$$[K_0 + \sum_{i=1}^{m} K_i\xi_i(\theta)]U(\theta) = \sum_{i=0}^{m}[K_i\xi_i(\theta)]U(\theta) = F \tag{9}$$

In order to move further from this point, a *polynomial chaos expansion* (PCE) is used [150]. PCE has been a very efficient and widely used method in engineering applications,

for meta-modelling, structural health monitoring, etc. [168, 169]. Here, it is used to decompose the field of displacements $U(\theta)$ yielding,

$$U(\theta) = \sum_{j=0}^{\infty} U_j \Psi_j(\theta) \tag{10}$$

where the $\Psi_j(\theta)$, for $j = 0, 1, 2...$ are polynomials defined in $\xi_i(\theta)$ which satisfy,

$$\Psi_0 \equiv 1 \tag{11a}$$

$$\mathbb{E}_{\xi(\theta)}[\Psi_j] = \mathbb{E}_{\theta}[\Psi_j] = 0 \qquad j > 0 \tag{11b}$$

$$\mathbb{E}_{\xi(\theta)}[\Psi_j(\theta)\Psi_k(\theta)] = \mathbb{E}_{\theta}[\Psi_j(\theta)\Psi_k(\theta)] = 0 \qquad j \neq k \tag{11c}$$

i.e. the terms $\Psi_j(\theta)$ are orthogonal in expectation ($\mathbb{E}[]$).

By truncating at P terms and substituting into equation (9), one obtains,

$$\sum_{i=0}^{m} K_i \xi_i(\theta) \sum_{j=0}^{P-1} U_j \Psi_j(\theta) = F \tag{12}$$

The solution to this equation can be found by minimising the error,

$$\epsilon_{m,P} = \sum_{i=0}^{m} K_i \xi_i(\theta) \sum_{j=0}^{P-1} U_j \Psi_j(\theta) - F \tag{13}$$

The best approximation of the exact solution $U(\theta)$ in the space H_P spanned by the $\{\Psi_k\}_{k=0}^{P-1}$ is obtained by minimising this residual in a mean-square sense. In a Hilbert space this is equivalent to requiring that the residual be orthogonal to H_P, yielding,

$$\mathbb{E}_{\theta}[\epsilon_{m,P}\Psi_k] = 0 \qquad k = 0, ...P-1 \tag{14}$$

Substituting equation (14) in equation (12), one finds,

$$\mathbb{E}_\theta[\sum_{i=0}^{m}\sum_{j=0}^{P-1} K_i\xi_i\Psi_j(\theta)\Psi_k(\theta)U_j] = \mathbb{E}_\theta[\Psi_k(\theta)F] \tag{15}$$

Introducing the following notation,

$$c_{ijk} = \mathbb{E}_\theta[\xi_i\Psi_j\Psi_k] \tag{16}$$

$$F_k = \mathbb{E}_\theta[\Psi_k F] \tag{17}$$

and,

$$K_{jk} = \sum_{i=0}^{M} c_{ijk}K_i \tag{18}$$

the stochastic FEM equation finally becomes,

$$\sum_{j=0}^{P-1} K_{jk}U_j = F_k \qquad k = 0, 1...P-1 \tag{19}$$

In this equation, every U_j is an N-dimensional vector, where N is the number of degrees of freedom in the system. In total, the P equations from above can be written as,

$$\begin{bmatrix} K_{00} & \dots & K_{0,P-1} \\ K_{10} & \dots & K_{1,P-1} \\ \vdots & & \vdots \\ K_{P-1,0} & \dots & K_{P-1,P-1} \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_{P-1} \end{bmatrix} = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_{P-1} \end{bmatrix} \tag{20}$$

Having solved this system for U_j, samples $U(\theta)$ can be generated by sampling $\xi_i(\theta)$ values and using equation (10). Thus, samples of the distribution of all displacements are generated. Solving this system is equivalent to solving the problem for every potential value of the random parameters. The augmented matrices in equation (20) are of dimension $NP \times NP$, where N are the degrees of freedom of the deterministic problem and P the

order of the PCE. Solving such a system, instead of a deterministic one, is much more computationally intensive, i.e. $\mathcal{O}(N^3P^3)$. However, it might not be as computationally inefficient as a sufficient number of Monte Carlo simulations. Solving the system yields samples, and therefore distributions, establishing SFE models as generative models.

Bibliography

[1] Analysis of the national infrastructure and construction pipeline. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/759222/CCS207_CCS1118987248-001_National_Infrastructure_and_Construction_Pipeline_2018_Accessible.pdf. Online; accessed 01-October-2021.

[2] Dr j. Krieger, european bridge maintenance: monitoring safety in Germany. https://www.innovationnewsnetwork.com/european-bridge-maintenance/568/. Online; accessed 01-October-2021.

[3] American Society of Civil Engineers (2021) Infrastructure Report Card. https://infrastructurereportcard.org/. Online; accessed 01-October-2021.

[4] Wind energy in the UK: June 2021. https://www.ons.gov.uk/economy/environmentalaccounts/articles/windenergyintheuk/june2021. Online; accessed 06-October-2021.

[5] Offshore Wind in Europe. https://windeurope.org/wp-content/uploads/files/about-wind/statistics/WindEurope-Annual-Offshore-Statistics-2019.pdf. Online; accessed 06-October-2021.

[6] Net Zero by 2050. https://www.iea.org/reports/net-zero-by-2050. Online; accessed 06-October-2021.

[7] K.-J. Bathe. *Finite Element Procedures*. Klaus-Jurgen Bathe, 2006.

[8] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[9] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[11] C.R. Farrar and K. Worden. *Structural Health Monitoring: A Machine Learning Perspective*. John Wiley and Sons, 2011.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S.Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[13] A.E.H Love. *A treatise on the mathematical theory of elasticity*. Cambridge university press, 2013.

[14] H. Robbins and Sutton S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[15] J. Kiefer, J. Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[16] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[17] L. Davis. Handbook of genetic algorithms. 1991.

[18] J.H. Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[19] K. Worden. *Nonlinearity in structural dynamics: detection, identification and modelling*. CRC Press, 2019.

[20] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[21] N. Indurkhya and F.J. Damerau. *Handbook of natural language processing*, volume 2. CRC Press, 2010.

[22] S. Sagiroglu and D. Sinanc. Big data: A review. In *2013 international conference on collaboration technologies and systems (CTS)*, pages 42–47. IEEE, 2013.

[23] B. Settles. Active learning literature survey. 2009.

[24] L. Bull, K. Worden, G. Manson, and N. Dervilis. Active learning for semi-supervised structural health monitoring. *Journal of Sound and Vibration*, 437:373–388, 2018.

[25] R. Rosen, G. von Wichert, G. Lo, and K. D. Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 48(3):567–572, 2015.

[26] T. H. J. Uhlemann, C. Schock, C. Lehmann, S. Freiberger, and R. Steinhilper. The digital twin: Demonstrating the potential of real time data acquisition in production systems. *Procedia Manufacturing*, 9:113–120, 2017.

[27] M. Schluse, M. Priggemeyer, L. Atorf, and J. Rossmann. Experimentable digital twins—streamlining simulation-based systems engineering for industry 4.0. *IEEE Transactions on Industrial Informatics*, 14(4):1722–1731, 2018.

[28] F. Dembski, U. Wössner, M. Letzgus, M. Ruddat, and C. Yamu. Urban digital twins for smart cities and citizens: The case study of Herrenberg, Germany. *Sustainability*, 12(6):2307, 2020.

[29] M. Macchi, I. Roda, E. Negri, and L. Fumagalli. Exploring the role of digital twin for asset lifecycle management. *IFAC-PapersOnLine*, 51(11):790–795, 2018.

[30] D.J. Wagg, K. Worden, R.J. Barthorpe, and P. Gardner. Digital twins: State-of-the-art and future directions for modeling and simulation in engineering dynamics applications. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems Part B Mechanical Engineering*, 6(3), 2020.

[31] A. Fuller, Z. Fan, C. Day, and C. Barlow. Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 2020.

[32] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 2020.

[33] K. Tatsis, V. Dertimanis, I. Abdallah, and E. Chatzi. A substructure approach for fatigue assessment on wind turbine support structures using output-only measurements. *Procedia engineering*, 199:1044–1049, 2017.

[34] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[35] K. Worden, E.J. Cross, R.J. Barthorpe, D.J. Wagg, and P. Gardner. On digital twins, mirrors, and virtualizations: Frameworks for model verification and validation. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems Part B Mechanical Engineering*, 6(3), 2020.

[36] J. Berner, P. Grohs, G. Kutyniok, and P. Petersen. The Modern Mathematics of Deep Learning. *arXiv preprint arXiv:2105.04026*, 2021.

[37] H. B. Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[38] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[39] A. M. Turing. Computing machinery and intelligence. In *Parsing the turing test*, pages 23–65. Springer, 2009.

[40] D.A. Freedman. *Statistical models: theory and practice*. Cambridge university press, 2009.

[41] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[42] O.Z. Maimon and L. Rokach. *Data mining with decision trees: theory and applications*, volume 81. World scientific, 2014.

[43] D.J.C. MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[44] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[45] B.W. Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[46] N. Lawrence and M. Jordan. Semi-supervised learning via Gaussian processes. *Advances in neural information processing systems*, 17:753–760, 2004.

[47] L.A. Bull, K. Worden, and N. Dervilis. Towards semi-supervised and probabilistic classification in structural health monitoring. *Mechanical Systems and Signal Processing*, 140:106653, 2020.

[48] M. McDermott, T. Yan, T. Naumann, N. Hunt, H. Suresh, P. Szolovits, and M. Ghassemi. Semi-supervised biomedical translation with cycle wasserstein regression GANs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[49] L.A. Bull, T.J. Rogers, C. Wickramarachchi, E.J. Cross, K. Worden, and N. Dervilis. Probabilistic active learning: an online framework for structural health monitoring. *Mechanical Systems and Signal Processing*, 134:106294, 2019.

[50] G. Lample and D.S. Chaplot. Playing FPS games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[51] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1-3):37–52, 1987.

[52] M.A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233–243, 1991.

[53] A. Gisbrecht, A. Schulz, and B. Hammer. Parametric nonlinear dimensionality reduction using kernel t-SNE. *Neurocomputing*, 147:71–82, 2015.

[54] B. C. Csáji et al. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7, 2001.

[55] B.T. Polyak. Introduction to optimization. Technical report, 1987.

[56] L.R. Medsker and L.C. Jain. Recurrent neural networks. *Design and Applications*, 5, 2001.

[57] M. M Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[58] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V.F̃lores Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gülçehre, H.F. Song, A.J. Ballard, J. Gilmer, G.E. Dahl, A. Vaswani, K.R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu. Relational inductive biases, deep learning, and graph networks. (arXiv:1806.01261v3 [cs.LG]), 2018.

[59] C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[60] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[61] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang. Towards the automatic anime characters creation with generative adversarial networks. *arXiv preprint arXiv:1708.05509*, 2017.

[62] J. Wu, C. Zhang, T. Xue, W.T. Freeman, and J.B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. *arXiv preprint arXiv:1610.07584*, 2016.

[63] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[64] B. W. Silverman. Using kernel density estimates to investigate multimodality. *Journal of the Royal Statistical Society: Series B (Methodological)*, 43(1):97–99, 1981.

[65] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2180–2188, 2016.

[66] G. Tsialiamanis, E. Chatzi, N. Dervilis, D. Wagg, and K. Worden. An application of generative adversarial networks in structural health monitoring. In *EURODYN 2020:*

Proceedings of the XI International Conference on Structural Dynamics, volume 2, pages 3816–3831. European Association for Structural Dynamics (EASD), 2020.

[67] G. Perarnau, J. Van De Weijer, B. Raducanu, and J.M. Álvarez. Invertible conditional GANs for image editing. *arXiv preprint arXiv:1611.06355*, 2016.

[68] Z. Zhang, Y. Song, and H. Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818, 2017.

[69] J.Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

[70] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A.A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[71] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. *Advances in neural information processing systems*, 29:613–621, 2016.

[72] L. Tarassenko. *Guide to Neural Computing Applications*. Elsevier, 1998.

[73] C. Zhang, M. Liang, X. Song, L. Liu, H. Wang, W. Li, and M. Shi. Generative adversarial network for geological prediction based on TBM operational data. *Mechanical Systems and Signal Processing*, 162:108035, 2022.

[74] G. Gazetas, I. Anastasopoulos, O. Adamidis, and T. Kontoroupi. Nonlinear rocking stiffness of foundations. *Soil Dynamics and Earthquake Engineering*, 47:83–91, 2013.

[75] A. Rytter. *Vibrational based inspection of civil engineering structures*. PhD thesis, 1993.

[76] K. Worden, G. Manson, and D. Allman. Experimental validation of a structural health monitoring methodology: Part I. Novelty detection on a laboratory structure. *Journal of Sound and Vibration*, 259(2):323–343, 2003.

[77] N. Dervilis, M. Choi, I. Antoniadou, K.M. Farinholt, S.G. Taylor, R.J. Barthorpe, G. Park, K. Worden, and C.R. Farrar. Novelty detection applied to vibration data

from a cx-100 wind turbine blade under fatigue loading. In *Journal of Physics: Conference Series*, volume 382, page 012047. IOP Publishing, 2012.

[78] K. Worden. Structural fault detection using a novelty measure. *Journal of Sound and Vibration*, 201(1):85–101, 1997.

[79] N. Dervilis, I. Antoniadou, R.J. Barthorpe J, E.J. Cross, and K. Worden. Robust methods for outlier detection and regression for shm applications. *International Journal of Sustainable Materials and Structural Systems*, 2(1-2):3–26, 2015.

[80] K. Worden and E.J. Cross. On switching response surface models, with applications to the structural health monitoring of bridges. *Mechanical Systems and Signal Processing*, 98:139–156, 2018.

[81] G. Manson, K. Worden, and D. Allman. Experimental validation of a structural health monitoring methodology: Part III. Damage location on an aircraft wing. *Journal of Sound and Vibration*, 259(2):365–385, 2003.

[82] P. Gardner, L.A. Bull, N. Dervilis, and K. Worden. Overcoming the problem of repair in structural health monitoring: Metric-informed transfer learning. *Journal of Sound and Vibration*, page 116245, 2021.

[83] P. Paris and F. Erdogan. A critical analysis of crack propagation laws. 1963.

[84] R.B. Randall. *Vibration-based condition monitoring: industrial, automotive and aerospace applications*. John Wiley & Sons, 2021.

[85] K. Worden, G. Manson, and E.J. Cross. On Gaussian process NARX models and their higher-order frequency response functions. In *Solving Computationally Expensive Engineering Problems*, pages 315–335. Springer, 2014.

[86] J.M. Aburto, F. Villavicencio, U. Basellini, S. Kjærgaard, and J.W. Vaupel. Dynamics of life expectancy and life span equality. *Proceedings of the National Academy of Sciences*, 117(10):5250–5259, 2020.

[87] J. Gosliga, P. Gardner, L.A. Bull, N. Dervilis, and K. Worden. Towards population-based structural health monitoring, Part II: heterogeneous populations and structures as graphs. In *Topics in Modal Analysis & Testing, Volume 8*, pages 177–187. Springer, 2021.

[88] J. Gosliga, P. Gardner, L. Bull, N. Dervilis, and K. Worden. Towards population-based structural health monitoring, Part III: Graphs, networks and communities. In *Topics in Modal Analysis & Testing, Volume 8*, pages 255–267. Springer, 2021.

[89] P. Gardner, L.A. Bull, J. Gosliga, N. Dervilis, and K. Worden. Towards population-based structural health monitoring, Part IV: Heterogeneous populations, transfer and mapping. In *Model Validation and Uncertainty Quantification, Volume 3*, pages 187–199. Springer, 2020.

[90] L.A. Bull, P.A. Gardner, J. Gosliga, T.J. Rogers, N. Dervilis, E.J. Cross, E. Papatheou, A.E. Maguire, C. Campos, and K. Worden. Foundations of population-based SHM, part I: Homogeneous populations and forms. *Mechanical Systems and Signal Processing*, 148:107141, 2021.

[91] J. Gosliga, P.A. Gardner, L.A. Bull, N. Dervilis, and K. Worden. Foundations of population-based SHM, part II: Heterogeneous populations–Graphs, networks, and communities. *Mechanical Systems and Signal Processing*, 148:107144, 2021.

[92] P. Gardner, L.A. Bull, J. Gosliga, N. Dervilis, and K. Worden. Foundations of population-based SHM, part III: Heterogeneous populations–mapping and transfer. *Mechanical Systems and Signal Processing*, 149:107142, 2021.

[93] S.J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[94] Y. Zhang and Q. Yang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 2018.

[95] O. Day and T.M. Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1):1–42, 2017.

[96] K. Weiss, T.M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):1–40, 2016.

[97] P Gardner, X Liu, and K Worden. On the application of domain adaptation in structural health monitoring. *Mechanical Systems and Signal Processing*, 138:106550, 2020.

[98] S.J. Pan, I.W.Tsang, J.T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE transactions on neural networks*, 22(2):199–210, 2010.

[99] L. Duan, D. Xu, and I. Tsang. Learning with augmented features for heterogeneous domain adaptation. *arXiv preprint arXiv:1206.4660*, 2012.

[100] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[101] H. Knublauch, D. Oberle, P. Tetlow, and E. Wallace. A semantic web primer for object-oriented software developers. W3C Working Group Note 9^{th} March 2006, W3C, 2006.

[102] M.A. Musen. The protégé project: a look back and a look forward. *AI Matters*, 1:4–12, 2015.

[103] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, C. Ursu, M. Dimitrov, M. Dowman, N. Aswani, I. Roberts, Y. Li, et al. *Developing language processing components with GATE Version 5:(a User Guide)*. University of Sheffield, 2009.

[104] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5^{th} Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[105] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[106] D.J.C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2002.

[107] C.K. Chui. *An Introduction to Wavelets*. Academic Press, 1992.

[108] J.H. Hensman, K. Worden, M. Eaton, R. Pullin, K.M. Holford, and S.L. Evans. Spatial scanning for anomaly detection in acoustic emission testing of an aerospace structure. *Mechanical Systems and Signal Processing*, 25:2462–2474, 2011.

[109] R. Fuentes, E.J. Cross, N. Ray, N. Dervilis, T. Guo, and K. Worden. In-process monitoring of automated carbon fibre tape layup using ultrasonic guided waves. In *Special Topics in Structural Dynamics, Volume 6*, pages 179–188. Springer, 2017.

[110] N. Dervilis, M. Choi, S.G. Taylor, R.J. Barthorpe, G. Park, C.R. Farrar, and K. Worden. On damage diagnosis for a wind turbine blade using pattern recognition. *Journal of Sound and Vibration*, 333:1833–1850, 2014.

[111] K. Worden. Structural fault detection using a novelty measure. *Journal of Sound and Vibration*, 201(1):85–101, 1997.

[112] D.J. Ewins. *Modal Testing: Theory, Practice and Application*. John Wiley & Sons, 2009.

[113] A.K. Chopra. *Dynamics of Structures*. Pearson Education India, 2007.

[114] K. Worden and P.L. Green. A machine learning approach to nonlinear modal analysis. *Mechanical Systems and Signal Processing*, 84:34–53, 2017.

[115] J. Müller, R. Klein, and M. Weinmann. Orthogonal Wasserstein GANs. *arXiv preprint arXiv:1911.13060*, 2019.

[116] W. Li, L. Fan, Z. Wang, C. Ma, and X. Cui. Tackling mode collapse in multi-generator GANs with orthogonal vectors. *Pattern Recognition*, 110:107646.

[117] M. Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[118] P. Welch. The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.

[119] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[120] N. Dervilis, E. Thomas T. Simpson, D.J. Wagg, and K. Worden. Nonlinear modal analysis via non-parametric machine learning tools. *Strain*, 55(1):e12297, 2019.

[121] E. Figueiredo, G. Park Gyuhae, J. Figueiras, C. Farrar, and K. Worden Keith. Structural health monitoring algorithm comparisons using standard data sets. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2009.

[122] G. J. SzéKely and M.L. Rizzo. The distance correlation t-test of independence in high dimension. *Journal of Multivariate Analysis*, 117:193–213, 2013.

[123] S.W. Shaw and C. Pierre. Normal modes for non-linear vibratory systems. *Journal of Sound and Vibration*, 164(1):85–124, 1993.

[124] B.F. Schutz. *Geometrical Methods of Mathematical Physics*. Cambridge University Press, 1980.

[125] T. Eguchi, P.B. Gilkey, and A.J. Hanson. Gravitation, gauge theories and differential geometry. *Physics Reports*, 66:213–393, 1980.

[126] M.J.D. Hamilton. *Mathematical Gauge Theory*. Springer, 2017.

[127] R. Abraham and J.E. Marsden. *Foundations of Mechanics*. American Mathematical Society, 2008.

[128] G. Reeb. Variétés symplectiques, variétés presque-complexes et systèmes dynamiques. *Comptes Rendus de l'Académie des Sciences de Paris*, 235:776—-778, 1952.

[129] J.L. Synge. On the geometry of dynamics. *Philosophical Transactions of the Royal Society, Series A*, 220:31–106, 1926.

[130] G.W. Mackey. *Mathematical Foundations of Quantum Mechanics*. Benjamin-Cummings, 1963.

[131] A.J. Izenman. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer, 2008.

[132] L. McInnes, J. Healy, and J. Melville. Umap: uniform manifold approximation and projection for dimension reduction. *arXiv preprint:1802.03426v2 [stat.ML]*, 2018.

[133] H. Sohn, K. Worden, and C.R. Farrar. Novelty detection under changing environmental conditions. In *Smart Structures and Materials 2001: Smart Systems for Bridges, Structures, and Highways*, volume 4330, pages 108–118. International Society for Optics and Photonics, 2001.

[134] C.E. Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[135] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[136] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th AGM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[137] T.N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint:1609.02907*, 2016.

[138] S. Zhang, H. Tong, J. Xu, and R. Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6, 2019.

[139] P.D. Dobson and A.J. Doig. Distinguising enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330, 2003.

[140] L. Ralaivola, S.J. Swadimas, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural networks*, 18:1093–1110, 2005.

[141] M. Zitnik, M. Agrawal, and J. Leskovec. Modeling polypharmacy side effects with graph convolutional neural network. *Bioinformatics*, 34:i457–i466, 2018.

[142] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[143] Y. LeCun et al. LeNet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20(5):14, 2015.

[144] A.J. Hughes, R.J. Barthorpe, N. Dervilis, C.R. Farrar, and K. Worden. A probabilistic risk-based decision framework for structural health monitoring. *Mechanical Systems and Signal Processing*, 150:107339.

[145] B. Delaunay. Sur la sphère vide. a la mémoire de Georges Voronoi. *Bulletin de l'Académie des Sciences de l'URSS. Classe des Sciences Mathématiques et na*, 6:793–800, 1934.

[146] A. Geéron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* OReilly, second edition, 2019.

[147] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[148] S. Kullback. *Information Theory and Statistics*. Courier Corporation, 1997.

[149] N. Dervilis, R.J. Barthorpe, I. Antoniadou, W.J. Staszewski, and K. Worden. Damage detection in carbon composite material typical of wind turbine blades using auto-associative neural networks. In *Health Monitoring of Structural and Biological Systems 2012*, volume 8348, page 834806. International Society for Optics and Photonics, 2012.

[150] R.G. Ghanem and P.D. Spanos. *Stochastic Finite Elements: a Spectral Approach*. Courier Corporation, 2003.

[151] B. Sudret and A. Der Kiureghian. *Stochastic Finite Element Methods and Reliability: a State-of-the-art Report*. Department of Civil and Environmental Engineering, University of California, 2000.

[152] G. Stefanou. The stochastic finite element method: past, present and future. *Computer methods in applied mechanics and engineering*, 198(9-12):1031–1051, 2009.

[153] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, January 1969.

[154] T.J. Rogers, G.R. Holmes, , E.J. Cross, and K. Worden. On a grey box modelling framework for nonlinear system identification. In *Special Topics in Structural Dynamics, Volume 6*, pages 167–178. Springer, 2017.

[155] O. Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[156] J. Engel, K.K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and Adam A. Roberts. GANsynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019.

[157] C. Esteban, S.L. Hyland, and G. Rätsch. Real-valued (medical) time series generation with recurrent conditional GANs. *arXiv preprint arXiv:1706.02633*, 2017.

[158] J. Yoon, D. Jarrett, and M. van der Schaar. Time-series generative adversarial networks. pages 5508–5518, 2019.

[159] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[160] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[161] C.R. Farrar and N.A.J. Lieven. Damage prognosis: the future of structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1851):623–632, 2007.

[162] C. Mylonas and E. Chatzi. Remaining useful life estimation under uncertainty with causal graphnets. *arXiv preprint arXiv:2011.11740*, 2020.

[163] M. Corbetta, C. Sbarufatti, M. Giglio, and M.D. Todd. Optimization of nonlinear, non-Gaussian Bayesian filtering for diagnosis and prognosis of monotonic degradation processes. *Mechanical Systems and Signal Processing*, 104:305–322, 2018.

[164] B. Peeters, W. Hendricx, J. Debille, and H. Climent. Modern solutions for ground vibration testing of large aircraft. *Sound and vibration*, 43(1):8, 2009.

[165] L. Auslander and R.E. MacKenzie. *Introduction to Differentiable Manifolds*. Dover Publications, 1963.

[166] A. Papoulis and S.U. Pillai. *Probability, Random Variables, and Stochastic Processes*. Tata McGraw-Hill Education, 2002.

[167] M. Loeve. Elementary probability theory. In *Probability theory i*, pages 1–52. Springer, 1977.

[168] M.D. Spiridonakos and E.N. Chatzi. Metamodeling of dynamic nonlinear structural systems through polynomial chaos NARX models. *Computers & Structures*, 157:99–113, 2015.

[169] M.D. Spiridonakos, E.N. Chatzi, and B. Sudret. Polynomial chaos expansion models for the monitoring of structures under operational variability. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 2(3):B4016003, 2016.