Mechanical submodels driven by machine learning : application to structural dynamics

Hamza Boukraichi

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES Paris

Sous-modèles mécaniques pilotés par machine learning : application à la dynamique des structures.
Mechanical submodels driven by machine learning: application to structural dynamics.

Soutenance par

Hamza BOUKRAICHI

20 Mars 2023

École doctorale n°621

Ingénierie des Systèmes, Matériaux, Mécanique, Energétique

Spécialité

Mécanique

Composition du jury :

Antony GRAVOUIL Professeur, INSA Lyon	*Président*
Icíar ALFARO Professeure, Universidad Zaragoza	*Rapporteuse*
Olga MULA Professeure associée , TU Eindhoven	*Rapporteuse*
Daniel RIXEN Professeur, TUM	*Examinateur*
Nissrine AKKARI Ingénieure de recherche, SAFRAN SA	*Examinatrice*
David RYCKELYNCK Professeur, Mines ParisTech	*Directeur de thèse*

MINES PARIS | PSL

Acknowledgement

I would like to express my sincere gratitude to the members of the jury for their time, attention, and valuable feedback. I appreciate the effort that Prof. Alfaro and Prof. Mula have taken to review my manuscript and provide me with their insightful comments. I would also like to thank Prof. Gravouil, Prof. Rixen, and Prof. Rabin for accepting to be part of the jury of my thesis defense. Your presence here today and your invested time and effort in evaluating my work means a lot to me.

I am deeply grateful to my supervisors, David, Nissrine, and Fabien, for their guidance, support, and encouragement throughout my research journey. Their expertise and wisdom have greatly influenced the direction and outcomes of my work. I appreciated working together on this thesis, and I am grateful for the productive and healthy environment they created and maintained for the fulfillment of my work. I could not have found better advisors, thank you for trusting me with this research opportunity.

I would like to extend my thanks to my colleagues in the office and the lab for their support, encouragement, and camaraderie, which made this journey more enjoyable and rewarding. Their willingness to share their knowledge and expertise has been invaluable.

To my friends, thank you for your encouragement, support, and understanding. Your inspiration has been a source of motivation for my work. I would like to express my heartfelt gratitude to Zakaria, who has been a constant source of support and encouragement throughout my research journey. Your unwavering belief in my abilities and your willingness to lend a listening ear have been invaluable to me. I cannot thank you enough for your friendship and guidance during this challenging but rewarding period of my life.

To my family, my parents, and my sister, I am grateful for your unwavering support and patience, and for pushing me to persevere. Without your support, this work would not have been possible.

I would like to express my heartfelt appreciation to a wonderful person who will recognize herself. Thank you for always being there for me and for encouraging me to go as far as possible. I would not have been able to complete this thesis without your unwavering support and belief in me. Your encouragement, guidance, and motivation have been invaluable to me throughout this journey. I am grateful for the countless hours you have spent listening to my concerns and helping me find solutions to the challenges I faced. Thank you for being my rock, my confidante, and my partner.

Finally, I would like to express my gratitude to anyone who has contributed to the accomplishment of this work.

Abstract

The primary goal of this thesis is to develop efficient and reliable numerical methods and deep learning methods for the reduction of parametric and/or non-parametric contact models in structural dynamics, including impact zone scenarios that can evolve over time on cabin aeronautical equipment. The approach is to determine a zone of interest in the physical model and construct models capable of generating boundary conditions to the physical model around the zone of interest. This modelisation will allow to explore the parametric space using the generative model while keeping the high-fidelity caracteristics of the physical solutions by solving the physical problem in the area of interest, and then use it to test out a variety of impact scenarios. Thus reducing the computational cost of the physical model. Our source code for Europlexus will be used to create the program. There will be more Python development for deep learning methods.

Keywords

Structural dynamics, Deep Learning, Machine Learning, Reduced Order Models, Generative Models, Uncertainty Quantification, Submodeling, Structural Zoom, Supervised Learning, Regression Models.

Résumé

L'Objectif principal de la thèse est le développement de méthodes numériques et des méthodes de deep learning efficaces et robustes pour la réduction des modèles paramétriques et/ou non-paramétriques de contact en dynamique des structures, avec des scénarios de zones d'impact qui peuvent évoluer au niveau de l'équipement aéronautique en cabine. L'approche consiste à déterminer une zone d'intérêt dans le modèle physique et à construire des modèles capables de générer des conditions aux limites autour de la zone d'intérêt pour le modèle physique. Cette modélisation permettra d'explorer l'espace paramétrique à l'aide du modèle génératif tout en conservant les caractéristiques de haute fidélité des solutions physiques en résolvant le problème physique dans la zone d'intérêt, puis de l'utiliser pour tester une variété de scénarios d'impact. Réduisant ainsi le coût de calcul du modèle physique. Notre code source pour Europlexus sera utilisé pour créer le programme. Il y aura plus de développement Python pour les méthodes d'apprentissage automatique.

Mots clés

Dynamique des structure, Apprentissage profond, Apprentissage automatique, Modèles d'ordre réduit, Modèles génératifs, Quantification d'incertitude, Sous-modélisation, Zoom structurel, Apprentissage supervisé, Modèles de régression.

Contents

1 Introduction **5**
- 1.1 Industrial context . . . 5
- 1.2 Related work . . . 5
- 1.3 Thesis objective . . . 8
- 1.4 Thesis organization . . . 9
- 1.5 Development environment . . . 11

2 State of The art **13**
- 2.1 Introduction to Machine Learning . . . 13
  - 2.1.1 Supervised learning paradigm . . . 15
  - 2.1.2 Bias-Variance tradeoff . . . 15
  - 2.1.3 Unsupervised Learning . . . 17
  - 2.1.4 Ensemble learning . . . 17
  - 2.1.5 Data scaling . . . 17
- 2.2 Dimension data reduction . . . 18
  - 2.2.1 Singular value decomposition . . . 18
  - 2.2.2 Proper Orthogonal Decomposition . . . 19
  - 2.2.3 Gappy-POD . . . 20
  - 2.2.4 Kernel Principal Component Analysis . . . 21
  - 2.2.5 Multi dimensional scaling . . . 22
  - 2.2.6 Proper generalized decomposition . . . 23
- 2.3 Introduction to Deep Learning . . . 23
  - 2.3.1 Perceptrons . . . 24
  - 2.3.2 Convolutional neural network . . . 25
  - 2.3.3 Activation function . . . 26
  - 2.3.4 Optimization step . . . 27
  - 2.3.5 Autoencoder . . . 27
  - 2.3.6 Neural nets are universal approximators . . . 28
  - 2.3.7 Physics Informed Neural Networks . . . 29
- 2.4 Generative adversarial networks . . . 29
  - 2.4.1 Standard Generative Adversarial Networks . . . 29
  - 2.4.2 Conditional GAN . . . 33
  - 2.4.3 IPM-based GANs . . . 34
  - 2.4.4 Relativistic Discriminator . . . 37
  - 2.4.5 Possible tasks with generative adversarial networks . . . 39
- 2.5 Uncertainty quantification for physical models . . . 40
- 2.6 Bayesian techniques . . . 47

3 Comparison between POD and CNN for regression in explicit dynamic 49
3.1 Introduction . . . 50
3.2 Problem Definition . . . 51
3.3 Methodology . . . 52
3.3.1 parameterized POD . . . 52
3.3.2 Parameterized Space-Time POD . . . 53
3.3.3 Parameterized Space-Time POD: Interpolation on Grassmann Manifolds . . . 53
3.3.4 Deep convolutional neural regressor . . . 56
3.4 Numerical Examples . . . 57
3.4.1 Data Sampling . . . 57
3.4.2 Data preprocessing . . . 58
3.4.3 Trained metamodels . . . 59
3.4.4 Results . . . 59
3.5 Conclusion . . . 66
3.6 Supplementary results . . . 66

4 Uncertainty quantification in a mechanical submodel driven by a Wasserstein Generative Adversarial Network 71
4.1 Introduction . . . 72
4.1.1 Related work . . . 73
4.1.2 Contribution . . . 73
4.2 Models . . . 74
4.2.1 Proper Orthogonal Decomposition (POD) . . . 74
4.2.2 Deep Convolutional Neural Regressor . . . 75
4.2.3 Wasserstein Generative Adversarial Network . . . 75
4.3 Use Case . . . 76
4.3.1 Domain definition . . . 76
4.3.2 Finite element models . . . 76
4.3.3 Dataset generation . . . 77
4.4 Numerical Results . . . 78
4.4.1 Data Sampling . . . 78
4.4.2 Trained submodels . . . 78
4.4.3 Parametric approach results . . . 79
4.4.4 Non-parametric approach results . . . 80
4.5 Conclusion . . . 84

5 Physics oriented data preprocessing for deep learning and optimization of deep learning architecture for physical data 85
5.1 Introduction . . . 86
5.2 Proposed Approach . . . 87
5.2.1 Background . . . 87
5.2.2 Canonical Polyadic Decomposition of convolutional filters . . . 87
5.2.3 Approximation of the CPD of convolutional layers . . . 89
5.2.4 A priori decomposed convolutional layers . . . 89
5.3 Weight sharing . . . 91
5.4 Time regularization . . . 92
5.5 Developped models . . . 92
5.5.1 Models annotations . . . 93

5.6 Physical Data preprocessing . . . 94
5.7 Numerical results . . . 97
5.7.1 2D Wave propagation with one source point and early stopping . . . 97
5.7.2 2D wave propagation with one source point . . . 102
5.7.3 2D wave propagation with four source points . . . 105
5.8 Conclusion . . . 110

6 Uncertainty quantification in impact simulation via conditional Wasserstein Generative Adversarial Network 113
6.1 Introduction . . . 114
6.2 Problem Definition . . . 115
6.2.1 Data processing . . . 117
6.2.2 Data Sampling . . . 120
6.3 Models . . . 120
6.3.1 Deep Convolutional Neural Regressor . . . 121
6.3.2 Generative Adversarial Network . . . 121
6.3.3 Vanilla GAN . . . 121
6.3.4 Wasserstein GAN . . . 121
6.3.5 Gradient Penalty and Spectral Normalization . . . 121
6.3.6 Relativistic Discriminator . . . 122
6.4 Numerical Results . . . 123
6.4.1 Regression - DcNR and Auto Encoder . . . 123
6.4.2 Adversarial Regression - GANs . . . 127
6.4.3 Uncertainty quantification by a conditional GAN . . . 129
6.5 Conclusion . . . 132

7 Conclusion 133
7.1 Main results and contributions . . . 133
7.2 Publication and valorizations . . . 134
7.3 Perspectives . . . 135

Bibliography 145

Chapter 1

Introduction

1.1 Industrial context

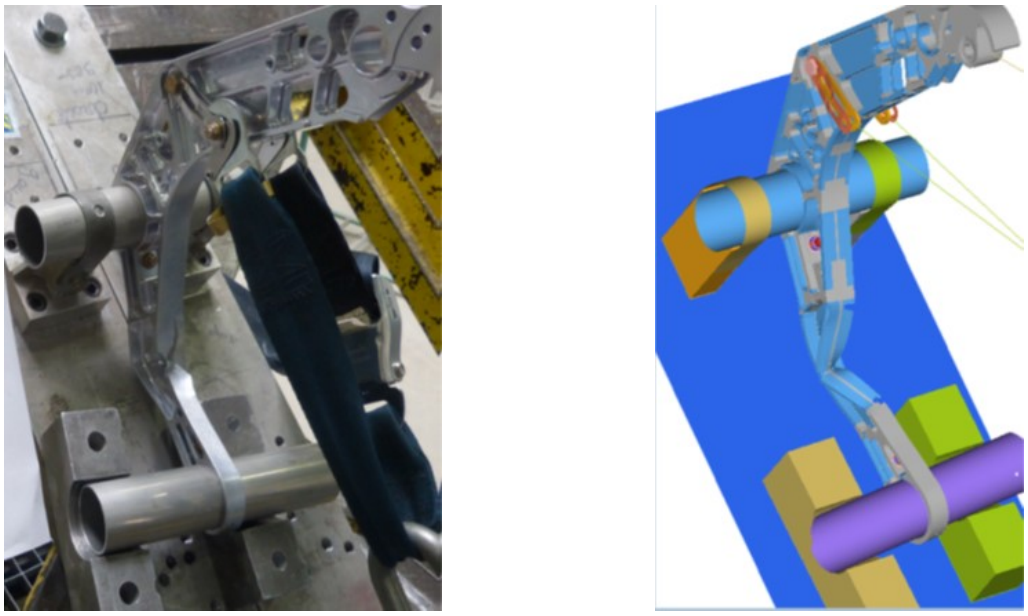
As in many industrial fields, digital simulation is an essential tool, used in all stages of Safran's activities. It implements complex calculation codes, where the computational time of each simulation can reach several hours, even several days. The context of this thesis is impact simulation which is transient, stochastic in nature and parameterized by random variables. Optimization or uncertainty propagation studies require predictions for a large number of realizations of these variables and parametric values, which increases drastically the simulation time. Also, in order to make intensive computing possible for applications such as parametric studies, optimization and design control, finite element models require acceleration techniques, for instance by using a reduction of the number of unknowns.

We are particularly interested in the design of aeronautical components, where a large number of operating regimes are to be expected on the basis of crash simulations involving fast dynamics. Also, it becomes essential to take into account the variability and the different scenarios, through multifidelity approaches which are based on methods of model order reduction of the equations of physics but also on machine learning approaches. Thanks to these multi-fidelity approaches, it would also be possible to integrate test data into the scale models in order to improve their robustness and their predictability with respect to a wide range of scenarios. The main objective of this thesis is to develop group-wide predictive, stochastic and multi-fidelity approaches to aeronautical structures under impact.

For instance, during strong turbulence or during an event impacting an aircraft, the cabin equipment, in particular the seats, undergo significant stress. In order to improve passenger safety, studies and simulations are carried out in order to model the forces and tensions exerted on the passenger seat belts and in particular on an area of interest of a mechanical part, called a spreader (see figure 1.1), which connects a seat belt to a passenger seat. In addition to the forces exerted on the area of interest of the spreader, uncertainties on the parameters of the spreader (see figure 1.2) must be taken into account to dimension the mechanical part.

1.2 Related work

Highly nonlinear and large finite element models require very large resources, due to the fine discretization of the domain in order to be able to take into account a wide range of spatial scales associated with the underlying complex physics. Moreover, when using an explicit solver in fast dynamics, a large number of time steps is required for the stability of the system of equations. Model order reduction solves this problem, but its application in the context we described is com-



(a) Spreader stress experiment (b) Spreader stress simulation

Figure 1.1: Spreader Experiment vs Simulation

plicated, and not necessarily suitable in the form of linear dimension reduction. It is mainly based on the projection of the underlying equations on a subspace of reduced dimension compared to the space of the finite element base. This subspace is generally constructed using the method of proper orthogonal decomposition (POD) Sirovich [1987] it results in spatial modes that describe the most probable structures encountered in a solution of the equations of the physical model. For highly nonlinear problems in structural dynamics, it is often necessary to carry out a hyper-reduction step in order to make the nonlinear reduced order model resulting from the projection of the equations efficient, thanks to anumerical approach which is based on a reduced integration domain. Two hyper-reduction methods have been widely used in the literature for nonlinear finite element models in structural dynamics: A priori hyper-reduction Ryckelynck [2005] and the Energy Conserving Sampling and Weighting (ECSW) method see An et al. [2008], Farhat et al. [2014] & Casenave et al. [2020].

However, contact problems in structural dynamics remain a problem for the reduction of nonlinear models, because of the inequality constraints that describe the contact forces, see Simo and Laursen [1992] & Wriggers [2006]. Solutions have been proposed to be able to build a robust nonlinear scale model to describe the solutions of contact problems. In Balajewicz et al. [2016a] the authors propose to build a reduced basis of the Lagrange multipliers which is positive in order to guarantee the positivity of the Lagrange multipliers intrinsic to the reduced order model. This is achieved by applying the non-negative matrix factorization (NNMF) algorithm as a positive version to the Singular Value Decomposition (SVD) in the POD method. An extension of hyper-reduction to contact problem can be found in Fauque et al. [2018]; Le Berre et al. [2022], by using a reduced integration domain and linear dimensionality reduction for primal variables only. In these works, the reduced integration domain acts as a submodel that reduces the computational complexity of contact problems. Other works in the literature have also been proposed in the context of contact model reduction: in Gastaldi et al. [2018], a linear and nonlinear model reduction technique is proposed. The degrees of freedom associated with the forces of contact are reduced thanks to a reduced basis obtained by the calculation of the Jacobian of the partial derivatives of the forces

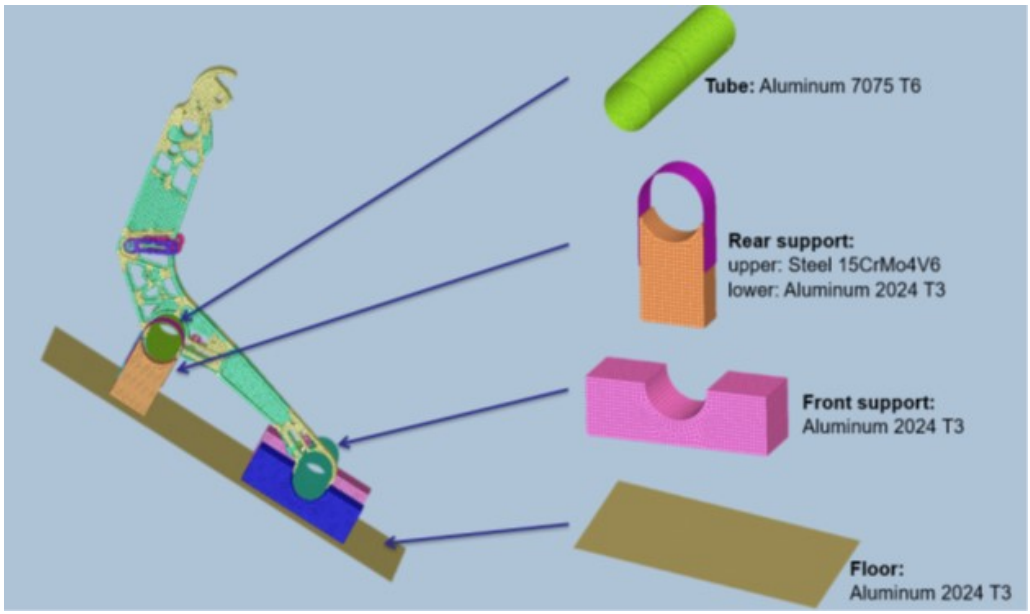


Figure 1.2: Spreader Parameters Variation

of contact compared to displacements. This nonlinear reduced basis is associated with boundary conditions in the contact zone. In Liu et al. [2018], a reduced order model of contact simulations has been proposed, based on the ALE (Arbitrary Lagrangian Eulerian) formulation. The motivation behind the use of this formulation is associated with the different possible contact scenarios for simulations of equipment in contact. The degrees of freedom related to the contact zone are taken into account in the reduced model by the ALE formulation which provides at the same time a precise mesh of the contact simulation. In Le Guennec et al. [2018], a non-intrusive and parametric model reduction approach is proposed to reduce car crash simulations. The reduced-order model is built using a statistical regression model based on a design of experiment that was built from a few high-fidelity simulations. Clustering and linear programming tools are used to make regression analysis more efficient. The parameters of the reduced model (whose number is 3 in this case) correspond to those of the dimensioning of the structure. In Benaceur [2018], developments of the reduced basis method (RB) and the empirical interpolation method (EIM) for nonlinear problems are presented. The works cited above consider for the most part parametrizable variations in the model equations. The non-parametric aspect that an impact zone can have linked to strongly nonlinear effects and even very high-dimensional variables is difficult to take into account by the methods already mentioned.

In Akkari et al. [2018], a model reduction method has been developed in the context of geometric (non-parametric) exploration of explicit unsteady fluid flows. This approach is based on an improvement of the classic Gappy-POD technique proposed by Everson and Sirovich [1995a], in order to update POD modes thanks to a local calculation around an evolving geometry. This method has been tested on semi-industrial cases of fluid flows representative of the complexity of the real cases that we find, typically fluid flows in aeronautical injection systems with variable geometry. The speed up obtained in the geometric exploration study was very interesting when compared to the computation time associated with the phase of the transient in explicit dynamics.

A large number of scenarios are to be expected for fast dynamic crash simulations. To our knowledge, there is no method in the state of the art that allows the consideration of a variable impact zone not known in advance. Preliminary work on the application of convolutional and

adversarial neural networks in numerical simulations in fluid mechanics, shows the first interesting leads on the representativeness of parametric and transient solutions belonging to a variety of very high dimensions, by auto-encoders whose dimension of the latent space is reduced to the number of parameters involved. The training of the auto-encoders could be done on data of reasonable size. Generative adversarial networks GAN Goodfellow et al. [2014] have also made it possible to generate physical solutions from a parametric and/or random normal distribution of very low dimension. GANs have also been used for super-resolution, such as in imaging (see also Jolicoeur-Martineau [2019], Arjovsky and Bottou [2017], Mescheder et al. [2018], Labatie [2019], Kim et al. [2019] & Xie et al. [2018a]). They could advantageously be coupled with reduced models to increase the spatial resolution in the contact zones and thus make it possible to better predict the quantities of interest of the simulation. The nonlinear effect of the scenarios would then be modeled by the GAN generator.

1.3 Thesis objective

The primary goal of this thesis is to develop efficient and reliable numerical methods and deep learning methods for the reduction of parametric and/or non-parametric contact models in structural dynamics, including impact zone scenarios that can evolve over time on cabin aeronautical equipment. The approach is to determine a zone of interest in the physical model and construct models capable of generating boundary conditions to the physical model around the zone of interest. This modelisation will allow to explore the parametric space using the generative model while keeping the high-fidelity caracteristics of the physical solutions by solving the physical problem in the area of interest, and then use it to test out a variety of impact scenarios. Thus reducing the computational cost of the physical model. Our source code for Europlexus will be used to create the program. There will be more Python development for deep learning methods.

The solution of partial differential equations modelling stationary or non-stationary physical phenomena is a widely studied topic. The classical methods of discretization used for their solution are the methods of finite differences, finite elements or finite volumes. Typically, these methods require the resolution of large linear systems, either because a fine discretization is necessary (necessary for example for the approximation of highly nonlinear solutions), or where a large-scale problem is targeted, since the uniform sampling of a volume requires a number of samples which grows exponentially with its dimension, an issue usually referred to as the Curse of Dimensionality.

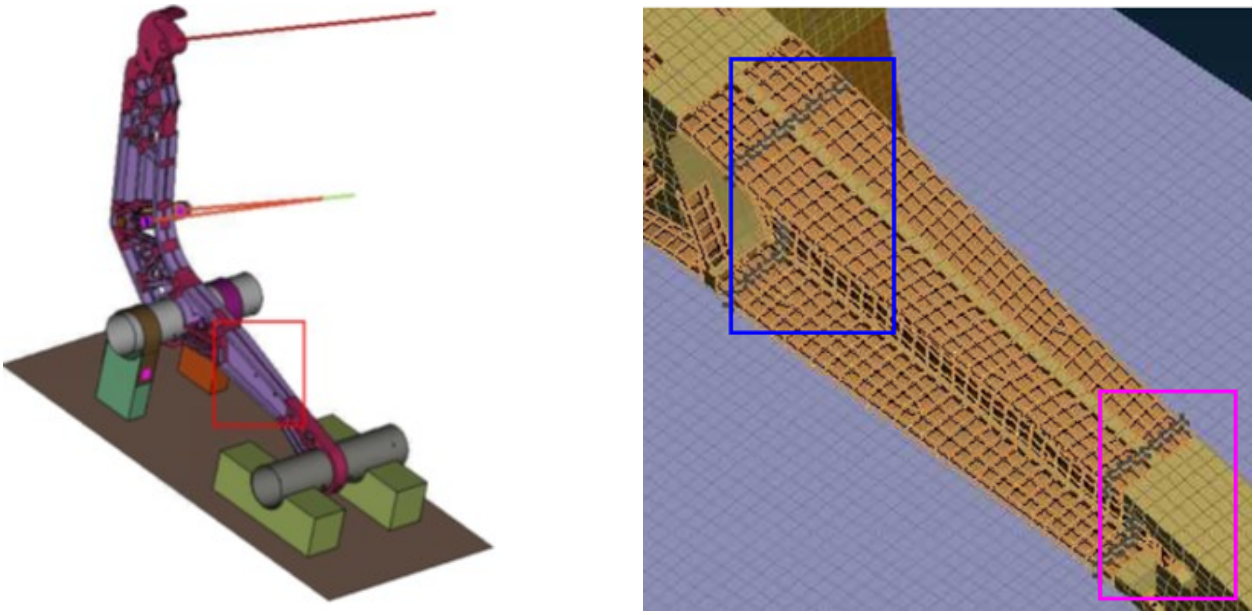
On the other hand, artificial neural networks have been recently successfully employed in different fields, such as classification and regression, image or pattern recognition, to name a few. Their promising performance encouraged also the spread of special neural network's hardware implementations, to decrease the computational training times and to provide a platform for efficient adaptive systems. Especially, artificial neural networks are well known for their excellent flexibility in approximating complex high-dimensional nonlinear functions.

The interest in using a neural network can be indeed motivated by different factors. First, the differential operator does not need to be discretized, and the approximate solution obtained possesses an analytic expression. This allows not only to have an approximation of the solution at all the points of the training interval but also to perform extrapolation outside the interval with an accuracy usually comparable to that obtained at the training points. This approach is useful especially in case of high dimensional equations and high dimensional parameters space, as it allows to alleviate the effect of the curse of dimensionality. More importantly, this approach provides a natural way to solve problems with nonlinear operators, with no need of linearisation.

One of the main issues with the use of artificial neural networks is to be able to successfully

train them. The training is based on the solution of an optimization problem that can be large-scale, like in the case of traditional techniques. This is the case of highly nonlinear solutions, as a network with a large number of weights may be necessary to approximate the solution with a sufficient accuracy. Gradient-based methods may exhibit a slow convergence rate and it may be difficult to properly tune the learning rate. Several studies were already carried on for the approximation of PDE's using neural networks (see Han et al. [2018], Long et al. [2018], Raissi et al. [2018], Raissi and Karniadakis [2018], Schaeffer [2017] & Mishra [2019]. In Calandra et al. [2019] the solution of PDEs is approximated by a feedforward network, which is then trained by a multilevel optimization method.

In this work, the proposed approach is to determine a zone of interest in the physical model (see figure 1.3) and learn models capable of generating boundary conditions to the physical model around the zone of interest (see figure 1.4). The zone of interest equiped with its boundary conditions and related PDEs is termed submodel. This modelisation will allow the user to explore the parametric space using the generative model while keeping the high-fidelity caracteristics of the physical solutions by solving the physical problem in the area of interest. Thus reducing the computational cost of the physical model. Similar work to this approach can be found in Launay [2021] for contact-free static mechanical problems.



(a) Spreader zone of interest

(b) Zone of interest boundaries

Figure 1.3: Spreader Zone of Interest

1.4 Thesis organization

In chapter 2 general state of the art method of machine learning, deep learning, linear model reduction and bayesian methods will be discussed, as well as state of the art method for uncertainty quantification in a physical configuration and in the presence of data generated from experiments of Finite Element Method (FEM) solvers. Chapter 3 will be dedicated to presenting results of comparison of linear dimensional reduction methods with deep learning methods in the case of a 3D impact simulation. We compared linear methods to deep convolutional neural regressor (DcNR)

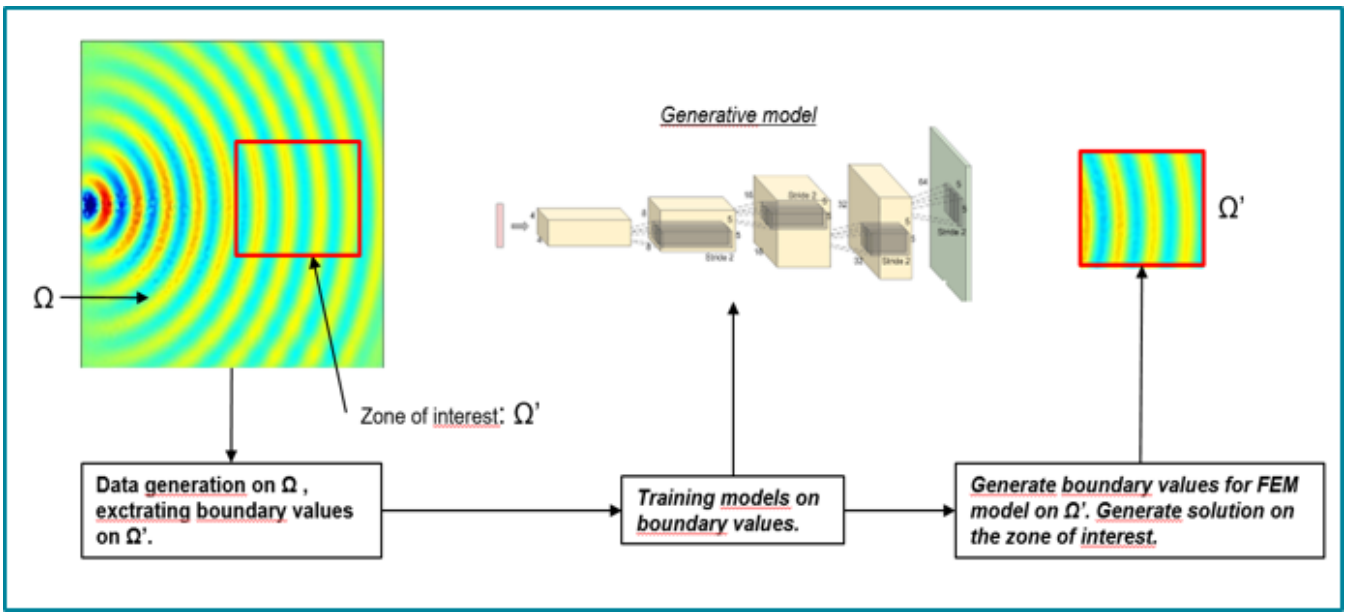


Figure 1.4: Zoom approach vizualization

and we illustrate that for contact cases such as the one investigated, linear methods behave very poorly and it is recommended to use non-linear data driven methods such as DcNR. We also illustrate that having the time step as a parameter of the DcNR helps reducing the dependency of the error with respect to the time. In chapter 4 an initial version of a nonparametric approach for uncertainty quantification in a FEM submodel using generative models will be presented and validated on a 2D use case. We have empirically shown that our methods obtain comparable and slightly better estimation of physical fields than classical neural networks approaches, while reducing the dimensionality of the learning problem and thus reducing the training cost of our models by restricting our attention to the boundary of a submodel, thus offering better generative behavior in the exploration of density tails. Chapter 5 will be dedicated to presenting proposed methods for adapting FEM models data to deep learning models as well as the optimization of neural nets architecture for physical data and the stabilization of time informed neural network loss function, the presented methods will be validated on a 2D use case. The compression approach proposed can also be applied to learning data in higher dimensions since the complexity of the models is linearly dependent of the dimension and actual deep learning code library only allow up to 3D data learning. We demonstrate that compressed convolutional networks achieve better performance on regression problems with fewer learnable parameters. Finally, in chapter 6 an optimized version of the approach presented in the fourth chapter will be proposed and then validated on a 3D impact simulation case. In this chapter we presente a novel method to assess for parametric and non-parametric uncertainty quantification using the same model, a Conditional Wasserstein GAN, relying on physical submodels over an area of interest. We used for training decomposition and regularization methods for training deep learning models for physical data which were discussed in length in this manuscript. We have empirically shown that our methods obtain good generative abilities.

1.5 Development environment

Convention: use of numpy array indices notations. Throughout this study, codes and experiments will be done on the machine learning Python library: Pytorch Paszke et al. [2019]. FEM models developments will be carried on the Python library Fenics Alnæs et al. [2015] for 2D cases and Europlexus Beccantini et al. [2022] for 3D impact cases.

Chapter 2

State of The art

Abstract

In this thesis chapter, we provide an in-depth review of the latest state-of-the-art methods of machine learning, including deep learning, linear model reduction, and Bayesian methods. The chapter also presents the latest methods for quantifying uncertainty in physical configurations, with a focus on dealing with uncertainty in data generated from Finite Element Method (FEM) solvers. Additionally, the chapter provides a detailed discussion on the use of Generative Adversarial Networks (GANs) for analyzing and generating physical data, including applications in areas such as image processing and signal processing. The presented techniques and methods have potential applications in various fields of research, including engineering, physics, and materials science.

Résumé

Dans ce chapitre de thèse, on propose une revue détaillée des dernières méthodes de pointe en apprentissage automatique, y compris l'apprentissage profond, la réduction de modèle linéaire et les méthodes bayésiennes. Le chapitre présente également les dernières méthodes pour quantifier l'incertitude dans les configurations physiques, en mettant l'accent sur le traitement de l'incertitude dans les données générées à partir de solveurs de méthode des éléments finis (MEF). De plus, le chapitre offre une discussion approfondie sur l'utilisation des réseaux antagonistes génératifs (GAN) pour analyser et générer des données physiques, y compris les applications dans des domaines tels que le traitement d'images et de signaux. Les techniques et les méthodes présentées ont des applications potentielles dans divers domaines de recherche, y compris l'ingénierie, la physique et les sciences des matériaux.

2.1 Introduction to Machine Learning

Machine learning (ML) is an area of study dedicated to understanding and constructing techniques that "learn," which are ways that exploit data to enhance performance on some set of tasks. In other words, ML seeks to understand and create methods that "learn." It is considered a component of artificial intelligence (AI). To be able to generate predictions or choices without being specifically programmed to do so, a model is created by machine learning algorithms by building it based on sample data. This kind of data is referred to as training data. Machine learning algorithms are used in a broad number of applications, including medical, email filtering, voice recognition, agriculture, and computer vision. These applications are utilized in situations when it would be difficult or impossible to design traditional algorithms to fulfill the necessary tasks. There is a

subset of machine learning that is closely connected to computational statistics, which focuses on generating predictions using computers. However, statistical learning is not the same as machine learning in its entirety. The study of mathematical optimization contributes to the science of machine learning by providing new methodologies, theoretical frameworks, and application fields. Data mining is a related area of research that focuses on unsupervised learning as a method for exploratory data analysis. Some applications of machine learning make use of data and neural networks in a manner that is intended to simulate the operation of a human brain. Predictive analytics is another name for machine learning when used to the context of solving business challenges. The primary objective of statistics is to draw conclusions about an entire population based on information gleaned from a sample, whereas the objective of machine learning is to identify generalizable predictive patterns. Although the two fields share a close relationship in terms of the methods they use, they are fundamentally different. In certain circles, the use of machine learning techniques has resulted in a new subfield of statistics known as statistical learning.

Common machine learning approaches may often be categorized into one of three classes, which correspond to distinct learning paradigms. These classes are characterized by the features of the signal available to the learning system, and they include the following:

- Supervised learning: is the approach used when the available data is comprised of labeled instances, whereby each data point has both features (covariates) and a label. Feature vectors (inputs) are used by supervised learning algorithms to develop a function that translates to labels (outputs). It does this by inferring a function from a collection of training samples that have been tagged. Each instance in supervised learning is a pair made up of an input object and an expected output value. An inferred function that may be used to map fresh instances is generated by a supervised learning algorithm, which examines the training data. In a perfect world, the algorithm would be able to accurately assign labels to cases it has never encountered. This necessitates reasonable generalization from the training data to novel circumstances by the learning algorithm. The generalization error is a statistic used to evaluate an algorithm's statistical quality.

- Unsupervised learning: is an algorithm that uses untagged data to discover patterns. The goal is that the model will be driven to construct a succinct representation of the data and will subsequently be able to develop inventive material based on that representation. In contrast to supervised learning, which requires labeling data, unsupervised approaches demonstrate self-organization to capture patterns as probability densities or as a collection of neural feature preferences stored in the model weights and activations. Semi-supervised learning is another degree of supervision in which a small fraction of the data is labeled.

- Reinforcement learning: is the study of how machines should behave in environments to maximize some abstract concept called "cumulative reward". In order to achieve its objective, the learner engages with a changing environment (such as driving a vehicle or playing a game against an opponent). The software receives feedback in the form of prizes as it solves its challenge and strives to maximize these benefits. Unlike supervised learning, which requires labeled input/output pairs to be given, reinforcement learning may function without such pairings and can learn from examples where less-than-optimal acts are not explicitly corrected. It is more important to strike a balance between exploring new territory and exploiting it (of current knowledge). Many reinforcement learning methods for this setting include dynamic programming approaches, therefore the environment is generally represented as a Markov Decision Process (MDP). Reinforcement learning techniques aim to solve huge

MDPs, which are beyond the scope of standard dynamic programming approaches since they do not need prior knowledge of a precise mathematical model of the MDP.

In this thesis, approaches will mainly focus on supervised, unsupervised and self supervised learning paradigms which is an intermediate form of unsupervised and supervised learning for processing unlabelled data to obtain useful representations that can help with downstream learning tasks.

2.1.1 Supervised learning paradigm

Given a set of N training examples of the form $[(x_i, y_i)]_{i \in [\![1\ ,\ N]\!]}$, a family of parametric models $M_\lambda = [f_\lambda : X \to Y]$ where X is the input space and Y the output or target space : $(\forall i \in [|1, N|])(x_i, y_i) \in X \times Y$ and an error function $r : X \times Y \to \mathbb{R}$, a supervised learning algorithm objective can be written as :

$$f_{obj} = \arg\min_{f \in M_\lambda} \frac{1}{N} \sum_{i=1}^{n} r(f(x_i), y_i) \tag{2.1}$$

The space M_λ is usually called the hypothesis space, and the function $R : M_\lambda \to \mathbb{R}$ is called the empirical risk defined as the empirical loss of the function f : $R(f) = \frac{1}{N} \sum_{i=1}^{n} r(f(x_i), y_i)$.

For the choice of the error function r, differentiating between classification and regression situations is crucial, where as an example for a regression objective the error function can the squared difference of the targets and the predictions, and for a classification objective the error function can be a binary response of predicting the right class by the model for each data sample.

2.1.2 Bias-Variance tradeoff

In Geman et al. [1992]; James [2003]; Goodfellow et al. [2016] it is noted that the balance that must be achieved between bias and variation is the primary concern. Imagine that we have access to a number of distinct training data sets, all of which are of the same high quality. A learning algorithm is said to have a bias towards a certain input x if, after being trained on each of these data sets, it consistently makes inaccurate predictions about what the appropriate output should be for x. A learning algorithm is said to have a large variance for a certain input x if it generates varied predictions for the value of the output variable after being trained on distinct sets of training data. The sum of the bias and the variance of the learning process is connected to the amount of mistake that is introduced by a learnt classifier's predictions. In most situations, there is a give-and-take relationship between bias and variation. In order to get good results with the data, a learning algorithm with minimal bias has to have flexibility. However, if the learning algorithm is too adaptable, it will match the parameters of each training data set uniquely and will thus have a large variance. The ability of many supervised learning techniques to modify the balance between bias and variance is an important feature that many systems share. This balancing act may be done either automatically or by giving a bias/variance parameter that the user can alter.

Let $\tilde{R}$ be the the generalization risk associated to the supervised learning objective : $\tilde{R}(f) = \mathop{\mathbb{E}}_{(x,y) \in X \times Y}[r(f(x), y)]$ and let $\tilde{f} = \arg\min_{f \in M_\lambda} \tilde{R}(f)$ and $\hat{f} = \arg\min_{f \in M_\lambda} R(f)$, then the generalization error of the empirically optimal model $\hat{f}$:

$$\tilde{R}(\tilde{f}) \leq \tilde{R}(\hat{f}) \leq \tilde{R}(\tilde{f}) + 2 \max_{f \in M_\lambda} |\tilde{R}(f) - \tilde{R}(\hat{f})| \tag{2.2}$$

Thus, the generalization error is bounded by a bias term $\tilde{R}(\tilde{f})$ and a fluctuation term $\max_{f \in M_\lambda} |\tilde{R}(f) - \tilde{R}(\hat{f})|$ associated to the variability of f in M_λ. However, the two values cannot be reduced at the same time. When the class M_λ is large, the bias diminishes but the fluctuations grow. The objective is to identify classes of models with small sizes to limit the fluctuations (entropy) and to minimize the risk associated to the problem while keeping the bias to a minimum. The Probably Approximately Correct (PAC) theorem states that we seek uniform outcomes regardless of the distribution of (X, Y) since it is unknown a priori, and that the fluctuation component goes to zero for N approaching ∞.

PAC Theorem : if $(\forall f \in M_\lambda)\tilde{R}(f)$ is bounded and $card(M_\lambda) < \infty$ then:

$$\mathbb{P}(\max_{f \in M_\lambda} |\tilde{R}(f) - \tilde{R}(\hat{f})| \leq \varepsilon) \geq 1 - \delta \tag{2.3}$$

Where $N \geq \frac{log(card(M_\lambda)) + log(\frac{2}{\delta})}{2\varepsilon^2}$.
This means that to achieve a low error ε in the learning process with a high enough probability 1-δ and a fixed number of samples N, the complexity and dimension of the space M_λ have to relatively low but this will lead to higher values to the bias term.

The choice of the optimal model from the hypothesis space may be made using either an empirical risk minimization strategy or a structural risk reduction strategy. When minimizing risk empirically, it is preferable to choose a function that best matches the available data. The penalty function in structural risk reduction regulates the bias/variance tradeoff.

The supervised learning method finds the optimal model that minimizes the empirical risk. As a result, a supervised learning algorithm may be built by using an optimization technique to determine the optimal model. Empirical risk minimization is identical to maximum likelihood estimation where the model is a conditional probability distribution and the loss function is the negative log likelihood.

When M_λ comprises a large number of candidate functions or the training set is too small, empirical risk reduction results in high variance and poor generalization. The learning algorithm may retain the training samples while not generalizing effectively. This is known as overfitting. By introducing a regularization penalty into the algorithm, structural risk minimization attempts to minimize overfitting. The regularization penalty may be thought of as a practical application of Occam's razor, which rewards less complicated functions over more elaborate ones. A broad range of penalties have been used to correlate to various definitions of complexity. The squared Euclidean norm of the weights, often known as the L_2 norm, is a popular regularization penalty. The L_1 norm is one of the others. Let's denote the penalty norm as $C(f_\lambda)$, the supervised learning problem can be rewritten as :

$$f_{obj} = \arg\min_{f \in M_\lambda} R(f_\lambda) + \gamma C(f_\lambda) \tag{2.4}$$

Where γ is a parameter that regulates the bias-variance tradeoff. When $\gamma = 0$, empirical risk minimization with minimal bias and large variance is obtained. When γ is of higher values, the learning algorithm will be biased and have a low variance. Cross validation may be used to determine the value of lambda.

2.1.3 Unsupervised Learning

Principal component and cluster analysis Roman [2019] are two of the most common approaches used in unsupervised learning. In unsupervised learning, cluster analysis is used to organize or divide datasets with similar properties in order to deduce algorithmic links. Cluster analysis is a subset of machine learning that organizes unlabeled, classified, or categorized data. Cluster analysis, rather of reacting to input, discovers similarities in data and responds to the existence or lack of such similarities in each new piece of data. This method aids in the detection of aberrant data points that do not fall into either category.

The topic of density estimation in statistics is a fundamental application of unsupervised learning, however unsupervised learning spans many other disciplines requiring summarizing and interpreting data aspects. It differs from supervised learning in that supervised learning seeks to infer a conditional probability distribution based on the label of incoming data, while unsupervised learning seeks to infer an a priori probability distribution.

Method of moments is one of the statistical methodologies for unsupervised learning. In the technique of moments, the unknown parameters (of interest) in the model are connected to the moments of one or more random variables, and may therefore be estimated using the moments. Moments are often approximated by empirical sampling. The fundamental moments consist of first and second order moments. In particular, it is shown that the technique of moments is excellent for learning the parameters of models with latent variables. Latent variable models are statistical models that include, in addition to the observable variables, a collection of unobserved latent variables. In contrast of supervised learning, unsupervised learning approaches cannot be written as a unique minimization problem since the objective function depends on the objective of the application, in this work unsupervised learning will mainly be used for dimensionality reduction of data and/or the construction of generative models. In the following examples of dimensionality reduction methods and generative models construction will be presented.

2.1.4 Ensemble learning

Ensemble techniques Opitz and Maclin [1999] in statistics and machine learning combine many learning algorithms to achieve greater prediction performance than each of the component learning algorithms alone. In contrast to a statistical ensemble in statistical mechanics, which is normally unlimited, a machine learning ensemble consists of just a specific finite number of different models, but allows for far more flexible structure to exist within those possibilities. Bootstrapping involves randomly selecting subsets of a dataset over a certain number of repetitions and variables. To provide a more potent outcome, these findings are then averaged. An example of an applied ensemble model is bootstrapping. Some applications of this work contain ensemble learning algorithms and bootstrapping such as Random Forest Regression (RFR) Breiman [2001]. The bootstrapping Random Forest approach combines ensemble learning methods with the decision tree framework to generate numerous randomly generated decision trees from the data, then averages the results to get a new result that often leads to good predictions/classifications.

2.1.5 Data scaling

Many machine learning models or estimators are built on the premise that all characteristics have values near to zero or, more crucially, that they all change on similar scales. Metric-based and gradient-based estimators, in particular, often depend on data that is roughly standard (centered features with unit variances). Estimators based on decision trees, however, are resistant to

such changes in data size. Scalers vary in how they estimate the parameters needed to translate and enlarge each feature, despite all being linear (or more properly affine) transformers. Scaling or Feature Scaling is the process of changing the scale of certain features to a common one. This is typically achieved through normalization and standardization. Normalization is the is converting a set of numbers to a range between zero and one. This method is more prevalent and practical for work involving regression analysis. Standardization refers to the process of transforming data into a scale with a mean of 0 and a standard deviation of 1. It is more popular and beneficial for classification jobs. Both scalers are feature-affine transformation of the training data, for standardization the weight and the bias are respectively the empirical mean and empirical standard deviation of data. In the presence of outliers in data, affine scalers are not efficient to eliminate outliers from scaled data, non-linear scaling approaches have been proposed called Power Transformers Yeo and Johnson [2000]; Box and Cox [1964], even though this approaches show good results in classification tasks, they show poor performance in regression tasks since for regressive models inverting the scaling in the output is necessary and for the non-linear approaches inverse function are not always defined.

2.2 Dimension data reduction

In the manifold hypothesis Fefferman et al. [2016], it is proposed that many real-world high-dimensional data sets really reside along low-dimensional manifolds within the corresponding high-dimensional space. Many data sets that at first seem to need many variables to represent may really be described by a very limited number of variables, analogous to the local coordinate system of the underlying manifold, as a result of the manifold hypothesis. It is hypothesized that this approach is what allows machine learning algorithms to be so successful in characterizing high-dimensional data sets by looking at only a few of shared characteristics. Using nonlinear dimensionality reduction methods in machine learning is connected to the manifold hypothesis, on the basis of the premise that data exists on a low-dimensional submanifold. The next section discussing linear dimensional reduction is partly inspired of Launay [2021], which provides a state-of-the-art of linear reduction approaches adopting reduced basis method strategy Almroth et al. [1978]. The reduced basis method (RBM) is a methodology for projecting data onto a subspace defined by a set of functions chosen to represent the data for a specific research instance. Each technique employs an efficient criteria for data selection in order to construct the reduced basis that generates the approximation subspace. The greedy approach is utilized, which limits the amount of Full Order Model (FOM) simulation to calculate while simultaneously limiting the size of the RB used to shrink the model. Most methods are divided into two parts: an offline part, also known as the training/learning phase, in which the reduced basis is also constructed using parametric training data for various values in the parametric space, and an online part in which the reduced problem that depends on the parameters is solved. Because the issue is projected on the approximation subspace, its dimensionality is substantially lower than it was for the original data.

2.2.1 Singular value decomposition

The singular value decomposition (SVD) Eckart and Young [1936] is a factorization of a real or complex matrix in linear algebra. It generalizes to any $n \times m$ matrix the eigendecomposition of a square normal matrix with an orthonormal eigenbasis. Singular value decomposition is discussed in the next section since it is often utilized in model order reduction. It does, in fact, provide a compressed Reduces Basis by using certain snapshots. A rectangular matrix may be decomposed

into three matrices with distinct characteristics using SVD.

Considering $S \in \mathbb{R}^{n\times m}$ a matrix of rank $d \leq \min(n,m)$, by applying the SVD theorem it exists a decomposition of S such that

$$S = U\,\Sigma\,V^T \tag{2.5}$$

Where $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times m}$ are orthogoanl matrices, $U^TU = UU^T = I_n$ and $V^TV = VV^T = I_m$ and $\Sigma \in \mathbb{R}^{n\times m}$ is written as :

$\Sigma = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{n\times m}$ with $D = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_d)$ such that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_d \geq 0$.

The column vectors $U[:,i]$ and $V[:,i]$ are respectively called left singular vectors and right singular vectors of S. The σ_d are sorted in descending order and are called singular values. The left singular values of S correspond to the eigenvalues of SS^T, the right singular values of S correspond to the eigenvalues of S^TS and the squared value of each singular value of S is equal to the corresponding eigenvalue of SS^T and S^TS. This is directly obtained through the following relations :

$$SS^T = U\Sigma\Sigma^TU^T \tag{2.6}$$

$$S^TS = V\Sigma^T\Sigma V^T \tag{2.7}$$

Since Σ is a block defined matrix with 3 null blocks, the equation 2.5 can be simplified as :

$$S = U[:,1:d]\,\Sigma\,V^T[1:d,:] \tag{2.8}$$

Σ is unique in the decomposition, U and V can be determined in an unique manner by pre-assigning both matrices signs. SVD finds the optimal Frobenius norm approximation of a matrix S for a given number of vectors. Furthermore, the singular values are closely related to the approximation error. The singular vectors with the highest singular values are the ones that holds the most descriptive information of the matrix S. The following is the relationship between the approximation error of a matrix S of rank d by its truncated SVD $\tilde{S}_l$ of rank $l \leq d$ and the singular values σ_i of S:

$$||S - \tilde{S}_l||_F = \left(\sum_{k=l+1}^{d} \sigma_k^2\right)^{\frac{1}{2}} \qquad \text{where} \qquad \tilde{S}_l = U[:,1:l]\,\Sigma[1:l,1:l]\,V^T[1:l,1:l] \tag{2.9}$$

2.2.2 Proper Orthogonal Decomposition

In order to isolate meaningful components in a chaotic flow, the notion of proper orthogonal decomposition (POD) has been established Lumley [1967]. This technique has its origins in the study of data. The purpose was to get down to a few key variables that captured the essence of the original data. Many disciplines examined this overarching concept of data extraction. The SVD described in 2.2.1 shares this core concept. One key distinction between Lumley's and s Sirovich [1987] POD snapshot technique is that the former takes use of a temporal average and a spatial correlation, while the latter does the contrary.

By definition, a POD basis associated with g, is an orthonormal basis $(\phi_n)_{n>=1}$ of the space (denoted X) determined so as to maximize the average respect to time of the kinetic energy contained in the projections of the subspace generated by the $((\phi_n))$. The vector ϕ of this base is

solution of the problem:

$$\frac{<(g,\phi)_X^2>}{||\phi||_X^2} = max_{\Phi \in X} \frac{<(g,\Phi)_X^2>}{||\Phi||_X^2}.$$

Where $(,)$ and $<,>$ respectively designate the scalar product on X and the temporal mean over a certain interval denoted $[0,T]$.

The maximization problem is equivalent to an eigenvalue problem, find $(\phi_i, \lambda_i)_{i \in [\![1\ ;\ n]\!]}$ such as :

$$\mathbb{R}\phi = \lambda\phi$$

where :

$$\mathbb{R} : X \to X$$

$$\Phi \ \to \ <(g,\Phi)>$$

$\mathbb{R}$ is a linear, compact, self-supporting and positive operator on X. Then according to the spectral theory the eigenvalues and the eigenvectors solutions of the problem form an orthonormal basis of X. The problem is then equivalent to a least square problem thanks to the Singular Value Decomposition Eckart and Young [1936].

2.2.3 Gappy-POD

In Everson and Sirovich [1995b] the authors proposed a variant of the classic POD approach where missing data on a matrix is recovered using a POD basis already calculated on data previously collected.

A non complete matrix is expressed as

$$g'(X) = m(X)g(X). \tag{2.10}$$

Where m is a mask and X being the coordinate of the pixels of the matrix. The mask m takes 2 values : 0 for masked data and 1 elsewhere. The challenge is then to approach the real matrix g using the POD basis $(\phi_n)_{[1..N]}$ ie. find $(a'_n)_{[1..N]}$ as :

$$g(x) = \sum_{n=1}^{N} a_n \phi_n. \tag{2.11}$$

$$g'(x) = \sum_{n=1}^{N} a'_n \phi_n. \tag{2.12}$$

Using a least-squares criterion to achieve a best fit to determine $(a'_n)_{[1..N]}$ we obtain :

$$(\forall k) \qquad (m(x)g(X) - \sum_{n=1}^{N} a'_n \phi_n, \phi_k) = 0. \tag{2.13}$$

Thus, using matrix form :

$$Ma' = f \tag{2.14}$$

where $M_{i,j} = (\phi_i, \phi_j)$ and $f_i = (m(x)a_i, \phi_i) = m(x)a_i$.

2.2.4 Kernel Principal Component Analysis

Principal component analysis (PCA) Pearson [1901] is a prominent approach for analyzing huge datasets with a high number of dimensions/features per observation, boosting data interpretability while maintaining the most information, and allowing multidimensional data visualization. PCA is a statistical approach used to reduce the dimensionality of a dataset. This is achieved by linearly converting the data into a new coordinate system in which the variance in the data can be expressed with lower dimensionality than the original data. PCA is analogous to fitting a p-dimensional ellipsoid to the data, with each axis representing a principle component. If the variance along an ellipsoid axis is small, then the variance along that axis is likewise small.

Global nonlinear approaches aim to capture the full geometry of the data space in a mapping to lower- dimensional space, which may be non-Euclidean. Kernel principal component analysis (kPCA) is an extension of the principal component analysis (PCA) technique that allows nonlinear characteristics or bases to be captured inside the PCA framework. Fundamental algorithms begin with a transformation of the data into a new, often nonlinear, domain. Then, principal component analysis is done on this new space (a kernel Hilbert space with well defined geometric features), and the resulting bases are used. Gaussian kernels, sigmoid kernels, kernels of radial basis functions, and linear kernels are all examples of popular mapping functions. Take into account n data points in $\mathbb{R}^d$. The data almost always can be represented in $d > N$ dimensions, but they cannot be linearly separated in $d < N$ dimensions. The function $\Phi(x_i)$ is introduced in such a way that:

$$\Phi \; : \; \mathbb{R}^d \rightarrow \mathbb{R}^N \tag{2.15}$$

A hyperplane may be readily defined here to allow for linear data partitioning. Since Φ generates linearly independent vectors for each data point, an eigen decomposition, as would be performed with conventional PCA, is unnecessary. Working in the feature space ϕ-space is usually avoided. In the feature space, the covariance matrix is defined as:

$$\bar{C} = \frac{1}{n} \sum_{j=1}^{n} \Phi(x_j) \Phi(x_j)^T \tag{2.16}$$

The eigenvalues λ and eigenvector V can be computed :

$$\lambda V = \bar{C} V \tag{2.17}$$

The solution of the eigenvalues problem V lie in the span of $\Phi(x_i)$ for $i = 1, ..., n$. Hence eq 2.18 can be written as :

$$\lambda \left(\Phi(x_i) \cdot V \right) = \left(\Phi(x_i) \cdot \bar{C} V \right) \qquad \text{for all } i = 1, ..., n \tag{2.18}$$

Moreover since V lie in the span of $\Phi(x_i)$ for $i = 1, ..., n$, V can be written as a linear combination of these vectors. It follows :

$$V = \sum_{i=1}^{n} \alpha_i \Phi(x_i) \tag{2.19}$$

Hence eq. 2.18 can be written differently :

$$\lambda \sum_{i=1}^{n} \alpha_i(\Phi(x_k) \cdot \Phi(x_i)) = \frac{1}{n} \sum_{i=1}^{n} \alpha_i(\Phi(x_k) \cdot \sum_{j=1}^{n} \Phi(x_j)) \cdot \left(\Phi(x_j) \cdot \Phi(x_i)\right), \quad (\forall k = 1, \cdots, n) \tag{2.20}$$

The $n \times n$ kernel matrix is then introduced :

$$K_{ij} = \Phi(x_i) \cdot \Phi(x_j) \tag{2.21}$$

It depicts the usually unsolvable feature space's inner product space. The dual form that develops during the formation of a kernel enables us to formally describe a version of PCA in which the eigenvectors and eigenvalues of the covariance matrix in ϕ-space are never solved. The N-elements in each column of K reflect the dot product of one converted data point with respect to all transformed data points (N points).

So eq. 2.20 can then be written as :

$$N\lambda K\alpha = K^2\alpha \tag{2.22}$$

As K is symmetric, it has a set of eigenvectors which spans the whole space, thus :

$$N\lambda\alpha = K\alpha \tag{2.23}$$

The kernel formulation of PCA is limited in that it computes not the principal components themselves, but rather the projections of our data onto those components, since we never operate directly in the feature space.

2.2.5 Multi dimensional scaling

MDS Borg and Groenen [2005] is a technique for showing the degree of similarity between individual examples in a collection. MDS is a technique for converting information about the pairwise distances between a collection of n items or persons into a configuration of n points mapped onto an abstract Cartesian space, commonly known as Principal Coordinates Analysis (PCoA). The technique accepts the dissimilarity matrix D between the data as input and returns a coordinate matrix in the MDS space X as output. This coordinate matrix seeks to minimize a loss function known as strain. The strain is provided by:

$$S = \left(\frac{\sum_{i,j}(B_{ij} - X_i^T X_j)^2}{\sum_{i,j} B_{ij}^2} \right)^{\frac{1}{2}} \tag{2.24}$$

B is obtained by double centering of the proximity matrix $P = D_{ij}^2$ such as :

$$B = -\frac{1}{2}CPC \tag{2.25}$$

Where C is the centering matrix defined as :

$$C = I_n - \frac{1}{n}J_n \tag{2.26}$$

Here n is the number of samples, I_n is the $n \times n$ identity matrix and J_n is a $n \times n$ matrix containing only ones. The eigenvalues Λ_{ii} and eigenvectors E_i are obtained by diagonalizing the B matrix.

Finally, in the MDS space X, the coordinate matrix may be derived as follows:

$$X = E[:,:m]\Lambda[:,:m]^{\frac{1}{2}} \tag{2.27}$$

In this case, m denotes the number of dimensions desired in MDS space. PCA and MDS are comparable in the Euclidean situation. MDS is interested in obtaining a euclidean distance representation based on a non Euclidean distance matrix. In this work, MDS will be mainly used in adapting physical data to the euclidean standard necessary for deep learning models.

2.2.6 Proper generalized decomposition

Proper Generalized Decomposition (PGD) Chinesta et al. [2010] is a data dimensionality reduction strategy that allows you to avoid the difficulties associated by consecutive enrichment, the PGD algorithm computes an approximation of the objective model. This implies that a new component (or mode) is calculated and added to the approximation with each repetition. In general, the more modes found, the closer the approach to the theoretical answer is. PGD modes, unlike POD main components, are not always orthogonal to one another. A reduced order model of the solution is constructed by choosing just the most relevant PGD modes. As a result, PGD is classified as a model order reduction method. For a solution $u \in \Omega_1 \times ... \times \Omega_D$, the approximation of rank l of u is represented with the equation 2.28 where l and the ϕ_i^j, with $i \in [1,l]$ and $j \in [1,D]$, are already known.

$$u(i_1,..,i_D) \approx \sum_{i=1}^{l} \phi_i^1 \times \cdots \times \phi_i^D \tag{2.28}$$

The number of terms l needed to obtain a good reference solution approximation depends on how well it can be split. Functions ϕ_i^j are found by an iterative enrichment. At the enrichment step $n+1$, the functions ϕ_i^j from the previous steps $i \leq n$ are already known and the new product of the D functions ϕ_i^j has to be found. To do so, an approximation of rank $n+1$ of u in the weak formulation of the problem is used. Hence a non linear system is obtained and needs to be solved with an iterative algorithm such as a fixed point algorithm. The resolution of this system is similar to solving a one dimensional problem for each of the D functions ϕ_i^j defined in Ω_j. The phenomenon, in high-dimensional models based on standard discretization approaches, where the number of degrees of freedom (dof) rises exponentially with the size D of the problem, is known as the curse of dimensionality. On the contrary, the number of dof in a PGD-based model rises linearly with the approximation problem complexity. The PGD's capacity to tackle high-dimensional issues is its major benefit, its main disadvantage is that it is a somewhat invasive procedure.

2.3 Introduction to Deep Learning

Artificial neural networks (ANN) or connectionist systems are computing systems that are inspired by, but not identical to, biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with task-specific rules Goodfellow et al. [2016]. An ANN is composed of a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. In ANN implementations, the "signal" at a connection is a real number, and the output of

each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. A single neuron is identified by its weight w and its bias b, for an input signal $x \in \mathbb{R}$ a neuron computes a response $y \in \mathbb{R}$ as:

$$y = wx + b \tag{2.29}$$

2.3.1 Perceptrons

The perceptron can be seen as the simplest type of neural network. It is a linear classifier or regressor. This type of neural network contains no cycle, it is a network of forward propagation neurons usually called feedforward network. In this network, information travels only in one direction, forward, from the input nodes, through the hidden layers (if any), and to the output nodes. Each layer of the network contains several weighted neurons connected to the different inputs. The output of the network a is then the weighted sum of the inputs of the graph and a bias passed in parameter of a function called the function of activation noted σ. The output of the network can then be written :

$$y = \sigma(w^T x + b) \tag{2.30}$$

where w and b $\in \mathbb{R}^n$. This is then generalized for multi-layer networks by composition of function considering the outputs of each layer as the inputs of the following layers.

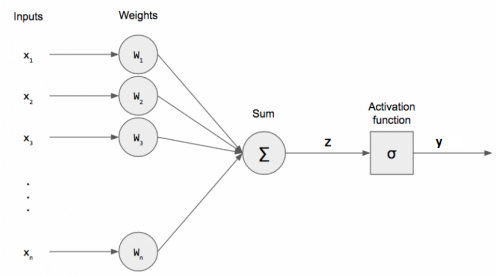


Figure 2.1: Perceptron

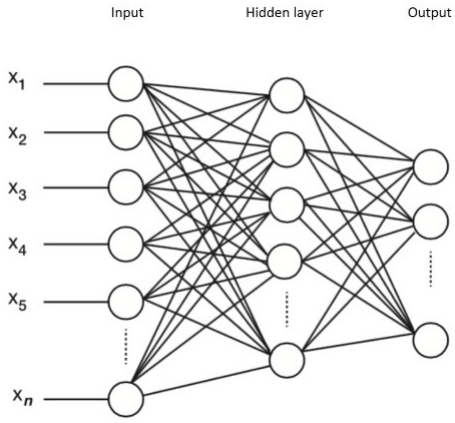


Figure 2.2: Multilayer Perceptron

The objective of the training is to adapt the different weights and biases of the network by seeking to minimize the empirical risk as any supervised learning problem, typically for a regression problem, the mean squared error is used as an error function.

2.3.2 Convolutional neural network

Convolutional networks LeCun et al. [1989], often known as convolutional neural networks (CNNs), are a specific kind of neural network with a grid-like structure used to analyze input. Time-series data, which can be seen as a 1-D grid sampling at regular intervals, and picture data, which can be viewed as a 2-D grid of pixels, are two examples. In practical applications, convolutional networks have proved quite effective. The term "convolutional neural network" suggests that the network performs a mathematical procedure known as convolution. Convolution is a kind of linear operation that is specialized. Convolutional networks are neural networks that substitute convolution for ordinary matrix multiplication in at least one layer. A convolutional neural network or convolutional neural network (CNN) is a type of acyclic artificial neural network, in which the connection pattern between neurons is inspired by the visual cortex of animals. They are often used in image processing problems. The CNN compares images fragment by fragment. The fragments it looks for are called features. By finding approximate features that look roughly alike in 2 different images, CNN is much better at detecting similarities than by an entire image-to-image comparison. In Mallat [2016] convolutional 2D layers are defined as follows, given an input $x \in \mathbb{R}^{n \times m}$ the output of the convolution of x with a kernel K:

$$y(i,j) = (x \star K)(i,j) = \sum_{l=1}^{n} \sum_{h=1}^{m} x(l+i-1, h+j-1) K(i,j) \tag{2.31}$$

This approach can be generalized for higher dimension inputs with higher dimension kernels. Thus, since a convolutional layer consists of a linear combination of the outputs of multiple convolution kernels passed in parameter for a non linear activation function, we can write for an input $x \in \mathbb{R}^{c \times n \times m}$, c being the numbers of channels in the input, and a collection of kernels $K \in \mathbb{R}^{c_k \times n_k \times m_k}$ where $n_k \leq n$ and $m_k \leq m$ and c_k the number of Kernels defined the number of channels for the output, which is usually called the feature map:

$$(\forall j \in [|1, c_k|]) \;\; y[j,:,:] = \sigma(b + \sum_{i=1}^{c} x[i,:,:] \star K[j,:,:]) \tag{2.32}$$

Where $b \in \mathbb{R}^{c_k \times n_{out} \times m_{out}}$ is the layer bias and $y \in \mathbb{R}^{c_k \times n_{out} \times m_{out}}$.

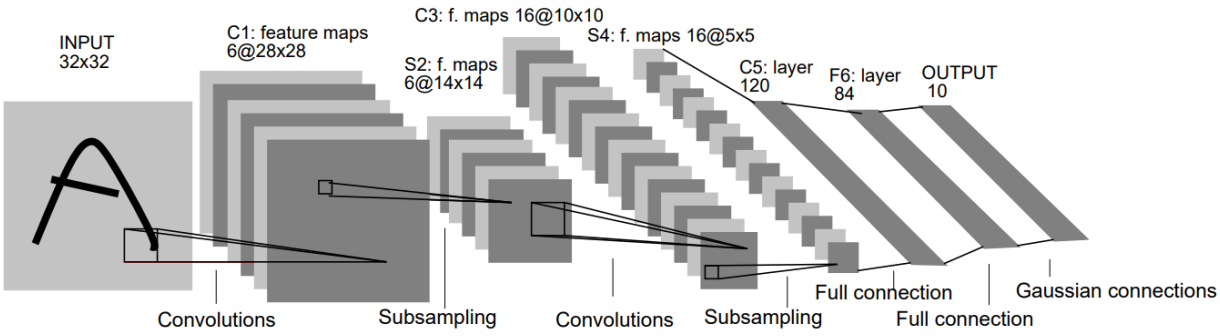


Figure 2.3: LeNet architecture a convolutional neural network from LeCun et al. [1998]

In Mallat [2016] and LeCun et al. [2015] more details about other layers used in convolutional networks adapted to image processing and computer vision such as max pooling. Since the objective of the thesis is to train generative models, we rely on deconvolution layers Zeiler et al. [2010a] or transposed convolution. A deconvolutional layer is defined as the solution of a convolutional equation. Given an input x and a kernel k, a deconvolution is solving the following equation for y:

$$x = k \star y \tag{2.33}$$

But since this operation is costly computationaly, in practice Dumoulin and Visin [2016] we achieve inverse convolution via swapping the forward and backward passes of a convolution. If the input and output were to be unrolled into vectors from left to right, top to bottom, the convolution could be represented as a sparse matrix C where the non-zero element are the elements of the Kernel repeated across the sparse matrix to simulate the convolution. The kernel defines a convolution, but how the forward and backward passes are performed determines whether it is a direct convolution or a transposed convolution. Even while the kernel k provides a convolution whose forward and backward passes are calculated by multiplying with C and C^T, it also defines a transposed convolution whose forward and backward passes are computed by multiplying with C^T and $(C^T)^T = C$. Thus one can use the same representation of the convolutional layer for the transposed convolution. Inverse convolution terminology is no longer used since the real inverse is not computed. With the representation of the convolution as a sparse matrix one can consider that convolutional layers are sparse linear layers which only learn data similarities locally.

2.3.3 Activation function

The primary objective of any neural network is to convert non-linearly separable input data into more linearly separable abstract characteristics by using a hierarchical structure of layers. Combinations of linear and nonlinear functions compose these layers. The choice of activation function depends on the architecture and the application or structure of the problem to solve using neural networks. The chosen activation function must verify some regularity conditions, since neural networks are optimized using gradient based methods. In Dubey et al. [2022] one can find a very detailed survey on different activation functions used in neural networks training. Usually the Rectified Linear Unit or ReLU $x \rightarrow max(0,x)$ is used for image processing applications, in this work, the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and hyperbolic tangent $Tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ are mainly used, their respective output range are : $[0,1]$ and $[-1,1]$.

2.3.4 Optimization step

As mentioned before, neural networks weights and bias are updated by minimizing the empirical risk. Nevertheless, two primary concerns emerge, considering neural networks with a large number of layers and weights, computing gradients become computationaly expensive or infeasible, in addition considering a training data set of important size makes the gradient computation even more expensive.

To tackle the first issue, the backpropagation algorithm has been proposed Rumelhart et al. [1995] which relies on the chain rule to compute the gradient for each neural networks layer with the value of the gradient of the previous layer. Feedforward neural networks training is then performed through backpropagation of the error. Hence the quantities, for neural network f with weights $w_i \in \mathbb{R}^m$, for the layer i, $\frac{\partial R(f)}{\partial w_i}$ are evaluated and used to update the weights. Considering the second issue, stochastic gradient descent methods Bottou and Bousquet [2007] have been proposed. Often abbreviated SGD, stochastic gradient descent is an iterative technique for maximizing an objective function with sufficient smoothness and regularity as differentiability or subdifferentiability. It might be considered a stochastic approximation of gradient descent optimization since it substitutes an estimate for the real gradient, which is derived from the whole data set, as a gradient calculated from a randomly selected subset of the data. This minimizes the very high computational cost, particularly in high-dimensional optimization problems, resulting in quicker iterations at the expense of a slower convergence rate. So combining both methods to compute $\frac{\partial R(f)}{\partial w_i}$ using the chain rule and a random subset of the training data, we can write the update rule of the weights of a layer in a feedforward neural network as:

$$w_i \leftarrow w_i - \eta \frac{\partial R(f)}{\partial w_i} \tag{2.34}$$

Where η is the learning rate considered.

Many optimization methods relying on stochastic gradient descent were proposed, Sun [2020] offers a global overview on stochastic optimizers, in this work, the stochastic optimizer Adam Kingma and Ba [2014] will mainly be used. Several studies have been carried on for the acceleration of the convergence of neural networks training with SGD, as an example Batch Normalization Ioffe and Szegedy [2015a] which is founded on the idea that covariate shift, which is known to complicate the training of machine learning systems, also applies to sub-networks and layers, and that eliminating it from the network's internal activations may benefit in training. The suggested solution derives from standardizing activations and embedding this standardization into the network design itself. This guarantees that any optimization technique used to train the network handles normalization in an acceptable manner. To allow stochastic optimization techniques typically used in deep network training, each mini-batch is normalized and backpropagates gradients via the normalization parameters. Batch Normalization adds just two more parameters per activation, hence preserving the network's capacity to represent data. Variants of this method have been proposed as Weight Normalization Salimans and Kingma [2016] or Layer Normalization Ba et al. [2016] which perform the normalization operation respectively on the weights or the layer. Layer Normalization use statistics summed over all the minibatches for a unique normalization step that doesn't depend on the batch size, in this work Batch Normalization will mainly be used.

2.3.5 Autoencoder

AutoEncoders (AE) are a kind of unsupervised learning also termed self-supervised learning that use neural networks for the purpose of representation learning. This unique neural network

architecture is designed in such a manner as to induce a bottleneck in the neuronal network's hidden layer. Since the desired output and input are identical, the bottleneck imposes a compressed version of the original input. The two primary components of an autoencoder are the encoder, which maps the input into a compressed latent space, and the decoder, which maps the compressed data from the latent space to the original space. Due to the fact that the latent space is smaller than the original space, autoencoders allow dimensionality reduction. Autoencoders are designed to reduce the discrepancy between the input and output in the context of a reconstruction error. The encoder and decoder can be defined as transition functions ϕ and ψ. if the L^2 norm is taken for the loss function it follows :

$$\begin{aligned} \Phi &: X \to F, \\ \Psi &: F \to X, \\ \Phi, \Psi &= \underset{\Phi,\Psi}{\operatorname{argmin}} ||x - (\Psi \circ \Phi)x||^2, \\ \dim(F) &\ll \dim(X). \end{aligned} \tag{2.35}$$

Here $\mathbb{X}$ and $\mathbb{F}$ denote the input space and the latent space respectively. For example, a very simple autoencoder is considered with one single hidden layer. The autoencoder takes as input $x \in \mathbb{R}^d = X$ and maps it into $h \in \mathbb{R}^p = F$.
In this case the hidden layer h is referred to as latent variables or latent representation. The functions Φ, Ψ can be any of the types of neural networks, their respective weights being updated by minimizing the reconstruction L_2 error.

2.3.6 Neural nets are universal approximators

The main approach used in Calandra et al. [2019] consists in using a one hidden layer perceptron for approaching the solution of a partial differential equation. This approach is justified by the universal approximation theorem Hornik et al. [1989] & Csáji et al. [2001], which states that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of $\mathbf{R}^m$, under mild assumptions on the activation function.

Universal Approximation Theorem Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a nonconstant, bounded, and continuous function (called the activation function). Let I_m denote the n-dimensional unit hypercube $[0,1]^m$. The space of real-valued continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\varepsilon > 0$ and any function $f \in C(I_m)$, there exist an integer N, real constants $v_i, b_i \in \mathbb{R}$ and real vectors $w_i \in \mathbb{R}^m$ for $i = 1, \ldots, N$, such that we may define:

$$F(x) = \sum_{i=1}^{N} v_i \sigma \left(w_i^T x + b_i \right) \tag{2.36}$$

as an approximate realization of the function f that is,

$$(\forall x \in I_m) \ |F(x) - f(x)| < \varepsilon \tag{2.37}$$

In other words, functions of the form $F(x)$ are dense in $C(I_m)$. This still holds when replacing I_m with any compact subset of $\mathbb{R}^m$.

Thus, physical numerical data being solutions of stable PDE's verify the conditions of being approached by multi layer perceptrons. In this work, we don't consider fracture mechanics that

may produced discontinuous numerical data.

2.3.7 Physics Informed Neural Networks

As a subset of universal function approximators, physics-informed neural networks (PINNs) Raissi et al. [2017a] are characterized by partial differential equations and may be trained with prior knowledge of any physical rules that apply to a particular data-set (PDEs). The lack of resilience in most state-of-the-art machine learning algorithms renders them unsuccessful in various biological and engineering systems with limited data availability. To improve the accuracy of the function approximation, previous knowledge of general physical principles is used as a regularization agent during the training of neural networks. By doing so, the information value of the given data is increased, making it easier for the learning algorithm to capture the proper answer and generalize well even with a small number of training samples.

Considering a nonlinear PDE of general form:

$$u(t,x) + \mathcal{N}[u,\lambda] = 0, (\forall t, x \in [0,T] \times \Omega) \tag{2.38}$$

Where u is the solution and $\mathcal{N}$ is a non linear operator parameterized by λ and ω is a subset of $\mathbb{R}$. First step being to compute data as solution of the equation for multiple parameter values to train a classical data-driven neural network, then by approximating u as a neural network $f(x,t) \approx u(x,t)$ we can compute the residual of the PDE on the neural approximation and use it as an additional loss function as follows for multiple parametric values called collocation points.

$$r(f,\lambda) = |f(x,t) + \mathcal{N}[f,\lambda]|^2 \tag{2.39}$$

Thus we obtain a new class of universal function approximators that is capable of encoding any underlying physical laws that govern a given data-set and can be described by partial differential equation with few training data. The main drawback of this method is that it is quite intrusive and it is not feasible for learning data from a FEM solver since for most case accessing the real PDE solved is computationaly expensive and intrusive.

In Jia [2021] the authors presents a brand-new strategy called Physics-guided Machine Learning for utilizing the complimentary characteristics of modern machine learning models and process-based models by integrating them. To better capture the dynamism of scientific systems and increase our understanding of underlying physical processes, novel process-guided deep learning models are presented.

2.4 Generative adversarial networks

2.4.1 Standard Generative Adversarial Networks

GAN (Generative Adversarial Networks) technology, introduced in Goodfellow et al. [2014], is an innovative programming approach for the development of generative models, that is to say capable of producing data themselves, especially images. Generative adversarial networks are an artificial intelligence method created to handle the challenge of generative modeling.
The objective of a generative model is to analyze a set of training samples and learn the probability distribution that created them. The computed probability distribution may then be used by GANs to produce new instances. Deep learning-based generative models are widespread, but GANs have

shown to be particularly effective (especially in terms of their ability to generate realistic high-resolution images). GANs have been successfully applied to a broad range of problems (mainly in research settings), but they continue to bring distinct difficulties and research possibilities since they are based on game theory, while the majority of other methods to generative modeling are focused on optimization. This type of network, as its name suggests, puts into competition 2 neural networks, a generator and a discriminator which can be perceptrons or convolutional networks, depending on the objective of the user. The discriminator's mission is to classify the data or images it receives as input into 2 classes, real or generated. While the aim of the generator is to generate from noise input data or images very close to the real data in order to deceive the discriminator. The goal of this type of network is then to learn how to generate images or data very close to reality. Figure 2.4 shows a visualization of this architecture.

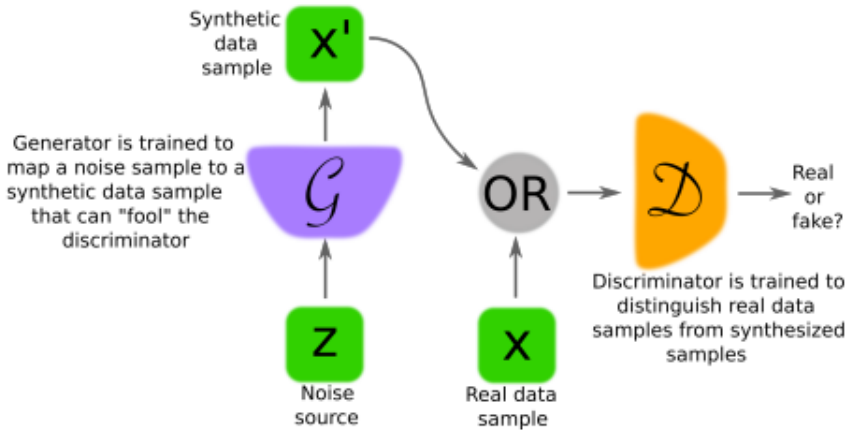


Figure 2.4: GAN architecture taken from Creswell et al. [2018]

Training

We can then model the training of a network of this type as a classic **min max** problem as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim\mathbb{P}_r}[Log(D(x))] + \mathbb{E}_{z\sim\mathbb{P}_z}[Log(1-D(G(z)))] \tag{2.40}$$

Where G and D are the outputs of the generator and the discriminator, and $\mathbb{P}_r$ and $\mathbb{P}_z$ are the distributions of the real data and of the noise at the input of the generator. Let θ_d and θ_g be the sets of trainable parameters of respectively the discriminator and the generator, we propose the training algorithm of a GAN Algorithm 1.

Algorithm 1: GAN Training

for i in $[1, itermax]$ **do**

Sample minibatch of m noise sample $(Z_1, ..., Z_m)$
Sample minibatch of m real data sample $(X_1, ..., X_m)$
Update Discriminator by ascending its stochastic gradient :

$$\nabla_{\theta_d}(\frac{1}{m}\sum_{i=1}^{m} Log(D(X_i)) + Log(1 - D(G(Z_i))))$$

Sample minibatch of m noise sample $(Z_1, ..., Z_m)$
Update Generator by descending its stochastic gradient :

$$\nabla_{\theta_g}(\frac{1}{m}\sum_{i=1}^{m} Log(1 - D(G(Z_i))))$$

end for

Properties

The theoretical analysis of the GAN was made on the basis of articles Goodfellow et al. [2014] and Arjovsky and Bottou [2017], this section will be dedicated to synthesizing this analysis. For the sake of clarity and conciseness, demonstrations of certain properties will not be proposed, for more details please refer to these documents.

Global Optimality For a fixed generator G, the optimal discriminator which minimizes (2.40) is:

$$D^*_G(x) = \frac{P_r(x)}{P_r(x) + P_g(x)} \tag{2.41}$$

Where P_r is the probability density of real data and P_g is the probability density of fake data generated by the Generator.

Proof Let G be the fixed generator. We have then :

$$D = \arg\max V(G, D) \tag{2.42}$$

$$\begin{aligned} V(G,D) &= \int_{\chi} P_r(x) log(D(x)) \partial x + \int_{Z} P_g(z) log(1 - D(G(z))) \partial z \\ &= \int_{\chi} (P_r(x) log(D(x)) + P_g(x) log(1 - D(x))) \partial x \end{aligned} \tag{2.43}$$

And the function $x \to a\ log(x) + b\ log(1-x)$ restricted to $[0,1]$ reaches its maximum on $\frac{a}{a+b}$. Thus :

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)} \tag{2.44}$$

Theorem 1 : If P_r and P_g have supports contained in 2 disjoint subsets $\mathscr{R}$ and $\mathscr{G}$ then it exists a discriminator $D^* : \chi \to [0,1]$ smooth and optimal such as :

$$P_r[D^*(x) = 1] = 1, \quad P_g[D^*(x) = 0] = 1, \quad (\forall x \in \mathbb{M} \cup \mathbb{P}) \ \ \nabla_x D^*(x) = 0 \tag{2.45}$$

Definition We define the following norm which will be used later :

$$||D|| = sup_{x\in\chi}|D(x)| + ||\nabla_x D(x)||_2$$

Theorem 2 Let $g_\theta : Z \to \chi$ be the generator output function, which induces a distribution P_g, P_r the distribution of real data and D a differentiable discriminator.

If the conditions of Theorem 1 are satisfied and
$\exists\ \varepsilon\ et\ M$ such as $||D - D^*|| < \varepsilon$ and $\mathbb{E}_{z\sim P_z}[||J_\theta g_\theta(z)||_2^2] < M^2$ then

$$||\nabla_\theta \mathbb{E}_{z\sim P_z}[Log(1 - D(g_\theta(z)))]||_2 < \frac{M\varepsilon}{1-\varepsilon}$$

Where J is the Jacobian matrix.

Corollary

$$\lim_{\varepsilon\to 0} ||\nabla_\theta \mathbb{E}_{z\sim P_z}[Log(1 - D(g_\theta(z)))]||_2 = 0$$

We conclude that the more the discriminant is driven towards optimality the more the generator gradient tends towards 0 and updating its weights becomes impossible.

To correct the fact that the generator gradient tends towards 0, an alternative to the generator gradient is proposed in Arjovsky and Bottou [2017], by modifying the cost function Equation 6.8 we obtain for the gradient:

$$\nabla_\theta(\mathbb{E}_{z\sim P_z}[-log(D(g_\theta(z)))])$$

Theorem 3 Let $g_\theta : Z \to \chi$ be the generator output function, which induces a distribution P_g, P_r distribution of actual data. Let us suppose that the hypotheses of Theorem 1 are verified and that D is a discriminator such that $D^* - D = \varepsilon$ is a Gaussian distribution indexed by x and independent for each x. And $\nabla_x D^* - \nabla_x D = r$.
Then each coordinates of $\mathbb{E}_{z\sim P_z}[\nabla_\theta - log(D(g_\theta(z)))]$ is a Cauchy's distribution centered on infinite variance.

Proof Having $D^*(g_\theta(z)) = 0$ and $\nabla_x D^*(g_\theta(z)) = 0$ thus :

$$\mathbb{E}_{z\sim P_z}[\nabla_\theta(-log(D(g_\theta(z))))] = \mathbb{E}_{z\sim P_z}[-\frac{J_\theta g_\theta(z)\nabla_x D(g_\theta(z))}{D(g_\theta(z))}]$$

$$= \mathbb{E}_{z\sim P_z}[-\frac{J_\theta g_\theta(z) r(z)}{\varepsilon(z)}]$$

We therefore have a ratio of 2 variables which follow normal distribution, we then obtain a Cauchy distribution , and as the mean is calculated on the z, the distribution is centered.

We can conclude that this alternative for the gradient is unstable. This instability can be corrected while training the network by adding noise to the output in the cost function.

2.4.2 Conditional GAN

The principle of GAN networks can be extended to a conditioned model Mirza and Osindero [2014] if the generator and the discriminator are both conditioned by additional information y. In the generator, noise and information are combined, while in the discriminator, noise and information are considered as 2 network inputs. So the problem can be written:

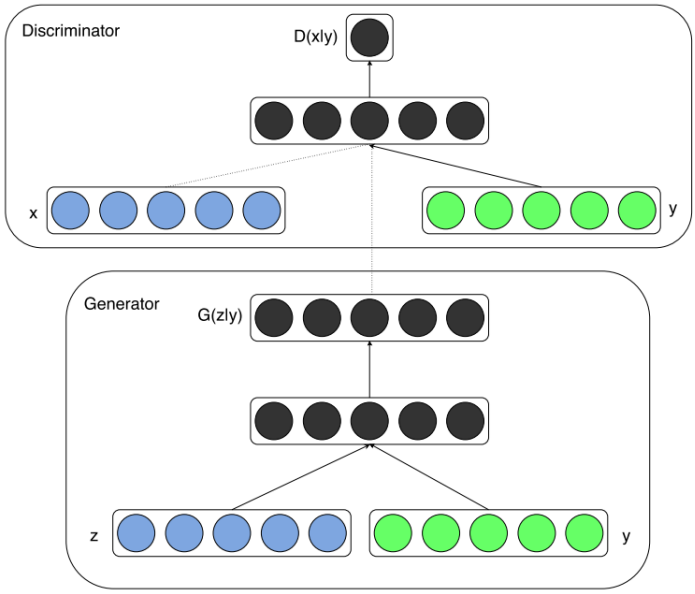


Figure 2.5: Conditional GAN architecture taken from Mirza and Osindero [2014]

$$min_G\ max_D\ \ V(D,G) = \mathbb{E}_{x\sim P_{data}}[Log(D(x|y)] + \mathbb{E}_{z\sim P_z}[Log(1-D(G(z|y)))]$$

Thanks to this modified architecture the generation mode of the generator can be controlled, its output will be dependent of a random noise vector θ and a set of deterministic parameters defined p. A deep convolutional GAN is a variant of the GAN which is simply using Deep convolutional networks for both the generator and the discriminator (sometimes for the Generator Only). In our opinion, this architecture is more suited for physics driven approach, since using spatial aware filters (2D or 3D) is more efficient to capture the physics properties of the data. A C-DCGAN being a conditional and convolutional GAN, using both the previous architectures, we can then consider the Generator (after successful training) as a class of conditional Monte Carlo estimator. Physics informed approach for generative adversarial network Yang and Perdikaris [2019] have been proposed where the generator is constrained with the residual of the PDE, another example in Xie et al. [2018b] where the discriminator is trained on batch of temporal data to obtain temporally coherent generative models for super resolution in a fluid simulation.

In Jolicoeur-Martineau [2019] the authors define generally GANs in terms of the discriminator as follows:

$$\mathscr{L}_D = \mathop{\mathbb{E}}_{x_r\sim\mathbb{P}}[\tilde{f}_1(D(x_r))] + \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z}[\tilde{f}_2(D(G(z))] \tag{2.46}$$

and the loss for the generator is the following:

$$\mathscr{L}_G = \mathop{\mathbb{E}}_{x_r \sim \mathbb{P}}[\tilde{g}_1(D(x_r))] + \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z}[\tilde{g}_2(D(G(z))] \tag{2.47}$$

Where $\tilde{f}_1$, $\tilde{f}_2$, $\tilde{g}_1$, and $\tilde{g}_2$ are scalar-to-scalar functions, $\mathbb{P}$ is the distribution of real data, $\mathbb{P}_z$ is often a multivariate normal distribution centered at 0 with variance 1. As we shown earlier the Standard GAN (or Vanilla GAN) has two different cost functions for the generator saturating Goodfellow et al. [2014] and non-saturating Arjovsky and Bottou [2017]. Non-saturating and saturating loss functions can be used to categorize the majority of GANs. GANs that have saturating losses have $\tilde{g}_1 = -\tilde{f}_1$ and $\tilde{g}_2 = -\tilde{f}_2$, whereas GANs that have non-saturating losses have $\tilde{g}_1 = \tilde{f}_1$ and $\tilde{g}_2 = \tilde{f}_2$. Since they can be seen as alternating between the same loss function's maximization and minimization, saturating GANs are intuitive. In contrast, non-saturating GANs can be viewed as maximizing the same loss function but substituting fake input for real data (and vice-versa). SGAN requires a cross-entropy loss, $\tilde{f}_1(D(x)) = log(D(x))$ and $\tilde{f}_2(D(x)) = log(1 - D(x))$, respectively. where $D(x) = \sigma(C(x))$, σ being the sigmoid activation function and $C(x)$ is the non-transformed discriminator output (which we refer to as the critic in accordance to Arjovsky et al. [2017]). In most GANs, $C(x)$ can be used to determine how realistic the input data is.

2.4.3 IPM-based GANs

Integral Probability Metrics (IPMs) are statistical divergences defined as :

$$IPM_{\mathscr{F}}(\mathbb{P}||\mathbb{Q}) = \sup_{C \in \mathscr{F}} \mathop{\mathbb{E}}_{x \sim \mathbb{P}}[C(x)] - \mathop{\mathbb{E}}_{x \sim \mathbb{Q}}[C(x)] \tag{2.48}$$

Equations 2.46 and 2.47 can be used to build IPM-based GANs. In these equations, $D(x) = C(x)$, and $f_1(D(x)) = g_2(D(x)) = D(x)$, there is no alteration that takes place. It can be seen that the loss functions for the discriminator and generator are both unbounded and, if directly optimized, will diverge to ∞. IPMs, on the other hand, make the assumption that the discriminator belongs to a particular class of function that grows slowly, preventing the loss functions from diverging. For example, WGAN Arjovsky et al. [2017] assumes a Lipschitz D and WGAN-GP Gulrajani et al. [2017] constraint the assumption of the Lipschitz discriminator using a gradient penalty. Each IPM applies a specific condition on the discriminator. For IPM-based GANs the discriminator is taken as having unbounded values in contrast to Standard GAN where the discriminator values are bounded to $[0, 1]$, the discriminator in a IMP-based GANs will be referred to as the critic C and for Standard GANs as the discriminator D where $D(x) = \sigma(C(x))$, σ being the sigmoid activation function.

Wasserstein GANs

Wasserstein GANs (WGANs) introduced by Arjovsky et al. [2017] is an IPM-based GAN using the Earth-Mover (EM) distance or Wasserstein-1, defined as follows:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\eta \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathop{\mathbb{E}}_{(x,y) \sim \eta}[||x - y||] \tag{2.49}$$

Where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distribution whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$.

For GANs training, using the Earth-Mover distance instead of Jensen-Shannon divergence will result in computing the distance between the real data distribution $\mathbb{P}$ and the image of the distribu-

tion of noise input $\mathbb{P}_z$ by the generator function w.r.t the generator parameters ω_G. Thus to optimize such cost function the Wasserstein-1 distance has to fulfill some regularity requirements w.r.t ω_G, we denotes the distribute of fake data generated by the generator as $G(\mathbb{P}_z)$.

The authors Arjovsky et al. [2017] demonstrate that under the assumption that G is locally Lipschitz w.r.t ω_G then $W(\mathbb{P}_r, G(\mathbb{P}_z))$ is continuous everywhere, and differentiable almost everywhere. This assumptions is verified if G is a feedforward neural network and $\mathbb{P}_z$ has finite expectation.

Using the Kantorovich-Rubinstein duality Villani [2009] we can rewrite the Wasserstein-1 distance as follows:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leq 1} \mathop{\mathbb{E}}_{x \sim \mathbb{P}_r}[f(x)] - \mathop{\mathbb{E}}_{x \sim \mathbb{P}_g}[f(x)] \tag{2.50}$$

The authors in Arjovsky et al. [2017] shows that having a generator G and a distribution $\mathbb{P}_z$ under the same assumptions as before (G locally Lipschitz w.r.t to ω_G and $\mathop{\mathbb{E}}_{x \sim \mathbb{P}_z}[x] < \infty$) are sufficient conditions for the existence of f solution of:

$$\max_{\|f\|_L \leq 1} \mathop{\mathbb{E}}_{x_r \sim \mathbb{P}_r}[f(x_r)] - \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z}[f(G(z))] \tag{2.51}$$

and

$$\nabla_{\omega_G} W(\mathbb{P}_r, G(\mathbb{P}_z))) = - \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} \nabla_{\omega_G}[f(G(z))] \tag{2.52}$$

Finding the function f to solve the maximizing problem in equation 2.50 is the next task. We may train a neural network parameterized with weights ω_C lying in a compact space $\mathbf{W}$ and then backpropagate via $\mathop{\mathbb{E}}_{z \sim \mathbb{P}_z}[f_{\omega_C}(G(z))]$, as we would with a normal GAN. $\mathbf{W}$'s compactness means that every f_{ω_C} will be K-Lipschitz for any K that solely depends on $\mathbf{W}$ rather than the weights. Thus approaching 2.50 up to a scaling factor and the capacity of the critic f_{ω_c}. After each gradient update, we can easily clamp the weights to a box (lets us say $\mathbf{W} = [0.01, 0.01]^l$) to ensure that the parameters ω_C resides in a compact space.

Because the EM distance is continuous and differentiable, we may (and should) train the critic until it is optimal. The reasoning is straightforward: the more we train the critic, the more accurate the Wasserstein gradient becomes, which is important due to the fact that Wasserstein is almost always differentiable. As the discriminator improves, the JS gradients get more accurate, but the real gradient is 0 since the JS is locally saturated and we observe vanishing gradients, as shown in 2.4.1.

It is a big advantage to be able to use WGAN in order to fine-tune the training that the critic receives. Once the critic has been completely trained, it will only provide a loss to the generator, this will enable us to train the generator in the same manner that we would train any other neural network. This is a promising indicator that we no longer need to stress about adjusting the capacity of the generator and the discriminator. If the critic is of high quality, then the gradients that we employ to train the generator will also be of high quality.

Results in Arjovsky et al. [2017] shows that WGAN improves the stability of GAN's training and solve the issue of mode collapse, nevertheless the clipping methods to ensure the Lipschitz regularity of the critic isn't robust and can lead to undesired behavior as shown in Gulrajani et al. [2017].

Wasserstein GAN with gradient penalty

The authors in Gulrajani et al. [2017] propose and alternative way to ensure the Lipschitz constraint on the critic, taking into consideration directly restricting the gradient norm of the critic's output with regard to its input since a differentiable function is said to be 1-Lipschtiz if and only if it has gradients with norms of at most 1 everywhere. A a soft form of the constraint was implemented in order to get around tractability concerns, and we do this by imposing a penalty on the gradient norm for both real and fake samples. Thus the algorithm to train a Wasserstein GAN with gradient penalty is presented in Algorithm 2. Although this method solves instability issues and propose a better method to fulfill the Lipschitz constraint it is computationaly costly.

Algorithm 2: WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{critic} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$

Require: The gradient penalty coefficient λ, the number of critic iteration per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2
Require: Initial critic parameters ω_0, initial generator parameters θ_0
while θ has not converged **do**
  for t = 1,...,n_{critic} **do**
    for i = 1,...,m **do**
      Sample real data x =(U,V) from training sets, latent variable $z \sim \mathcal{N}(0,1)$, a random number $\varepsilon \sim \mathcal{U}[0,1]$
      $\tilde{x} \leftarrow G_\theta(z)$
      $\hat{x} \leftarrow \varepsilon\tilde{x} + (1-\varepsilon)x$
      $L^{(i)} \leftarrow D_\omega(\tilde{x}) - D_\omega(x) + \lambda(||\nabla_{\hat{x}} D_\omega(\hat{x})||_2 - 1)^2$
    end for
    $\omega \leftarrow Adam(\nabla_\omega \frac{1}{m}\sum_{i=1}^m L^{(i)}, \omega, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{z^{(i)}\}_{i=1}^m \sim \mathcal{N}(0,1)$
  $\theta \leftarrow Adam(\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_\omega(G_\theta(z_i)), \theta, \alpha, \beta_1, \beta_2)$
end while

Spectral Normalization

In Miyato et al. [2018] the authors suggest an innovative weight normalization approach known as spectral normalization in order to make the training of discriminator networks more consistent and thus enforce the Lipschitz constraint without computing any additional gradient, reducing the computational cost of discriminator training. The Lipschitz constant of the discriminator function f is controlled via spectral normalization, which is achieved by restricting the spectral norm of each layer. The spectral norm of a matrix A is defined as follows:

$$\gamma(A) = \max_{h,h\neq 0} \frac{\|Ah\|_2}{\|h\|_2} = \max_{\|h\|_2 \leq 1} \|Ah\|_2 \tag{2.53}$$

Thus, for a linear layer $f(h) = W^T h$ the norm is given by $\|f\|_{Lip} = \gamma(W)$, then using the inequality $\|f_1 \circ f_2\|_{Lip} \leq \|f_1\|_{Lip}\|f_2\|_{Lip}$ considering a feedforward neural network with only 1-Lipschitz activation functions, we can constraint the Lipschitz condition on the neural network by applying Spectral Normalization to each layer. Spectral normalization normalizes the spectral

norm of the weight matrix W so that it satisfies the Lipschitz constraint :

$$\tilde{W}_{SN} = \frac{W}{\gamma(W)} \tag{2.54}$$

Power iteration method Golub and Van der Vorst [2000] is used to estimate the Spectral Norm at each optimization step of the Critic in a GANs optimization Setup, thus fulfilling the 1-Lipschitz condition with less computational cost than gradient penalty. Nonetheless this regularization technique is also useful to stabilize training in Standard GAN, since it provides regularity in the Discriminator.

2.4.4 Relativistic Discriminator

The intuition that the likelihood that real data are real $D(x_r)$ should decline as the likelihood that fake data are real $(D(G(z))$ increases, is the main argument for the relativistic Discriminator approach. This is the main missing attribute of SGAN. The authors (Jolicoeur-Martineau [2019]) offer three justifications for why Standard GAN ought to possess this quality. IPM-based GANs measure how realistic real data is compared to fake data, which implicitly accounts for the fact that some of the samples must be fake. This is the desirable behavior for Standard GAN.

For more understanding, we analyze the gradients of standard GANs with GANs built using IPM. In a non-saturating SGAN, it can be demonstrated that the discriminator and generator gradients are, respectively:

$$\nabla_{\omega_D} \mathscr{L}_D^{SGAN} = - \mathop{\mathbb{E}}_{x_r \sim \mathbb{P}} [(1 - D(x_r)) \nabla_{\omega_C} C(x_r)] + \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} [D(G(z)) \nabla_{\omega_C} C(G(z))] \tag{2.55}$$

$$\nabla_{\omega_G} \mathscr{L}_G^{SGAN} = - \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} [(1 - D(G(z))) \nabla_{\omega_G} C(G(z)) \mathbf{J}_{\omega_G} G(z)] \tag{2.56}$$

Where $C(x) \in \mathscr{F}$ the class of functions to determine the critic for the IPM and $\mathbf{J}$ is the Jacobian. In an IPM-based GAN, it can be demonstrated that the discriminator and generator gradients are, respectively:

$$\nabla_{\omega_D} \mathscr{L}_D^{IPM} = - \mathop{\mathbb{E}}_{x_r \sim \mathbb{P}} [\nabla_{\omega_C} C(x_r)] + \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} [\nabla_{\omega_C} C(G(z))] \tag{2.57}$$

$$\nabla_{\omega_G} \mathscr{L}_G^{IPM} = - \mathop{\mathbb{E}}_{z \sim \mathbb{P}_z} [\nabla_{\omega_G} C(G(z)) \mathbf{J}_{\omega_G} G(z)] \tag{2.58}$$

Based on these equations, we can see that SGAN has the same dynamics as IPM-based GANs when:

1. $D(x_r) = 0$ and $D(G(z)) = 1$ in the discriminator step of SGAN.
2. $D(G(z)) = 0$ in the generator step of SGAN.
3. $C(x) \in \mathscr{F}$.

Assuming that the generator and discriminator are trained to optimality in each step and that it is possible to distinguish perfectly between the real and fake data (a strong assumption, but typically true since the first steps training), we would have that $D(x_r) = 1$, $D(G(z)) = 0$ in the generator step and that $D(x_r) = 1$, $D(G(z)) = 1$ in the discriminator step for most x_r and z. Thus, $D(x_r) = 0$ in the discriminator phase would be the lone unmet assumption.

If all of the assumptions were maintained and the generator could indirectly impact $D(x_r)$, we would have that $D(x_r) = 0$ and $D(G(z)) = 1$, despite the fact that the scenario described above does not reflect reality (due to the fact that we do not ever train G to optimality). Therefore, SGANs would have gradients identical to those of IPM-based GANs. The authors Jolicoeur-Martineau [2019] hypothesize that making SGAN more analogous to IPM-based GANs might perhaps result in an increase in the system's degree of stability.

$D(x) = \sigma(C(x))$ may be used to define the discriminator in a typical GAN in terms of the untransformed layer $C(x)$. A straightforward method for making the discriminator relativistic (i.e., having the output of D depend on both real and false data) is to sample from real/fake data pairs $x = (x_r, G(z))$ and define $D(x) = \sigma(C(x_r) - C(G(z))$.

This alteration may be interpreted as follows: The discriminator evaluates the likelihood that the provided real data is more realistic than a randomly picked set of fake data. In a similar fashion, we may define $D_{rev}(x) = \sigma(C(G(z)) - C(x_r))$ as the probability that the provided fake data is more realistic than randomly picked real data. Another aspect of this discriminator is that we do not need to incorporate D_{rev} in the loss function through $log(1 - D_{rev}(x))$ since $D_{rev}(x) = 1 - \sigma(C(G(z))_C(x_r)) = \sigma(C(x_f) - C(G(z))) = D(x)$, hence, $log(D(x)) = log(1 - D_{rev}(x))$. The discriminator and generator (non-saturating) loss functions of the Relativistic Standard GAN (RS-GAN) may be stated as:

$$\mathscr{L}_D^{RGAN} = \mathop{\mathbb{E}}_{(x_r,z)\sim(\mathbb{P},\mathbb{P}_z)}[log(\sigma(C(x_r) - C(G(z))))] \tag{2.59}$$

and

$$\mathscr{L}_G^{RGAN} = \mathop{\mathbb{E}}_{(x_r,z)\sim(\mathbb{P},\mathbb{P}_z)}[log(\sigma(C(G(z)) - C(x_r)))] \tag{2.60}$$

In general, we consider relativistic any discriminator defined as $(C(x_r) - C(G(z)))$, where a is the activation function. This implies that nearly every GAN is capable of having a relativistic discriminator. This creates a new class of models that we refer to as Relativistic GANs (RGANs). Using a relativistic discriminator, these GANs take on the following structure:

$$\mathscr{L}_D^{RGAN} = \mathop{\mathbb{E}}_{(x_r,z)\sim(\mathbb{P},\mathbb{P}_z)}[\tilde{f}_1(C(x_r) - C(G(z)) + \tilde{f}_2(C(G(z)) - C(x_r))] \tag{2.61}$$

and

$$\mathscr{L}_G^{RGAN} = \mathop{\mathbb{E}}_{(x_r,z)\sim(\mathbb{P},\mathbb{P}_z)}[\tilde{g}_1(C(G(z)) - C(x_r)) + \tilde{g}_2(C(x_r) - C(G(z))] \tag{2.62}$$

Using the identity function ($\tilde{f}_1(y) = \tilde{g}_2(y) = y$, $\tilde{f}_2(y) = \tilde{g}_1(y) = y$) leads to a degenerate instance since there is no supremum or maximum. However, if one adds a constraint such that $C(xr) - C(G(z))$ is constrained, there is a supremum and one obtains IPM-based GANs. Thus, although being distinct, IPM-based GANs share a very similar loss function centered on the distinction between critics. Importantly, $\tilde{g}_1$ is typically disregarded in GANs since its gradient is 0 and the generator has no effect on it. In RGANs, however, g_1 is affected by fake data, and therefore the generator.

Relativistic Average GANs

In SGAN, the discriminator is interpreted substantially differently than in RSGAN. $D(x)$ in SGAN evaluates the chance that x is real, whereas $D(x_r - G(z))$ in RGANs assesses the likelihood that x_r is more realistic than $G(z)$. As a compromise, the authors proposed an alternative to the Relativistic Discriminator that has about the same meaning as the discriminator in SGAN while

remaining relativistic. The Relativistic average Discriminator (RaD) works by contrasting the critic of the input data with the average critic of samples of the opposite kind. It is possible to define the discriminator loss function for this strategy as follows:

$$\mathscr{L}_D^{RaSGAN} = -\mathop{\mathbb{E}}_{x_r\sim\mathbb{P}}[log(D(x_r))] + \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z}[log(1-D(G(z)))] \tag{2.63}$$

Where

$$D(x) = \begin{cases} \sigma(C(x) - \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z}[C(G(z))]) & \text{if x is real.} \\ \sigma(C(x) - \mathop{\mathbb{E}}_{x_r\sim\mathbb{P}}[C(x_r)]) & \text{if x is fake.} \end{cases} \tag{2.64}$$

RaD is more comparable to the conventional discriminator in its interpretation than the relativistic discriminator. RaD assesses the chance that the provided actual data is more realistic than the false data on average. Using the same formula as previously, we can expand this technique to work with any GAN loss function:

$$\mathscr{L}_D^{RaGAN} = \mathop{\mathbb{E}}_{x_r\sim\mathbb{P}}[\tilde{f}_1(C(x_r) - \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z}[C(G(z))])] + \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z}[\tilde{f}_2(C(G(z)) - \mathop{\mathbb{E}}_{x_r\sim\mathbb{P}}[C(x_r)])] \tag{2.65}$$

and

$$\mathscr{L}_G^{RaGAN} = \mathop{\mathbb{E}}_{x_r\sim\mathbb{P}}[\tilde{g}_1(C(G(z)) - \mathop{\mathbb{E}}_{x_r\sim\mathbb{P}}[C(x_r)])] + \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z}[\tilde{g}_2(C(x_r) - \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z}[C(G(z))])] \tag{2.66}$$

According to the findings of Jolicoeur-Martineau [2019], relativism has the potential to drastically improve the data quality and stability of GANs with no additional computational expense. In addition, combining a relativistic discriminator with other tools of the trade (such as spectral norm, gradient penalty, and so on) may result in a more refined state of the art.

2.4.5 Possible tasks with generative adversarial networks

Generative adversarial networks are widely used for the approximation of high dimension probability densities Abbasnejad et al. [2019a]; Haas and Richter [2020] and also uncertainty quantification for physical data using physics informed approach Yang and Perdikaris [2019], indeed considering a GAN training paradigm, with partially conditional or unconditional GANs, the Generator (after successful training) can be seen as a class of non-parametric Monte Carlo estimator for physical data for non-conditioned parameters and a parametric estimator for conditioned ones, both approaches can be combined. In the other hand if the conditional approach is complete (all parameters are conditioned), the GAN is then used in a regression purpose, this regression training approach is called adversarial regression Bigolin Lanfredi et al. [2019]; Boget [2019]; Haas and Richter [2020], and then more statistics could be computed with respect to the physical parameters of the model. So as to gain precision in the Generator prediction, a mean over a certain number of random vectors selections can be considered :

$$\frac{\sum_{i=1}^{n} G(Z_i, p)}{n} \tag{2.67}$$

In this approach there is no nonparametric uncertainy, one can also view this approach as training an uncertainty propagation tool around multiple parametric values, in the exploitation phase, the generator can be viewed as a randomized and parameterized simulation generator trained

on deterministic data. Then one can use many predictions of the generator while varying the random vector z as a tool to propagate uncertainties of simulations with respect to fixed parameters p.

The Discriminator as a proxy

After training is complete, most systems relying on GAN training discard the Discriminator and only retain the Generator to use in future data generation. As such, Turner et al. [2019] propose the Metropolis-Hastings GAN, a GAN that combine the generator with information from the discriminator to create an improved generator. Both networks are combining using Markov Chain Monte Carlo methods. Azadi et al. [2018] is another work that correct the generator sampling using the discriminator, where for each sample generated by the generator a rejection probability is computed using the discriminator where only the samples in the adequate threshold are accepted.

Fréchet Inception Distance

Images generated by a generative model, such as a generative adversarial network, may be evaluated using the Fréchet inception distance (FID) Heusel et al. [2017]. In contrast to the inception score (IS) Barratt and Sharma [2018], which merely looks at how the generated data are distributed, the FID compares your generated data to a real-world dataset, the training data set. This score relies on the Fréchet Distance between two multidimensional Gaussian distribution p_1, p_2 where $p_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, defined as follow:

$$d_F(p_1, p_2) = ||\mu_1 - \mu_2||_2^2 + Tr(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{\frac{1}{2}}) \tag{2.68}$$

Inception score is then defined as the Fréchet distance between the output of the last convolutional layer of the Inception Model Szegedy et al. [2015] trained on image classification for the generated data with the generator and also the training data. The Gaussian is the maximum entropy distribution for given mean and covariance, therefore we assume the coding units of Inception Model for unseen data to follow a multidimensional Gaussian distribution. Let f be the Inception model stripped from his last activation layer, the Fréchet Inception Score of a generative model G trained on a data set $(x_i)_{1\leq i\leq n}$ is defined as:

$$F_{score}(G) = d_F(f(x), f(G(z))) \tag{2.69}$$

Where $(z_j)_{1\leq j\leq n_s}$ is a set of random variable to sample data from the generator, n_s to be determined empirically. Although this score is adequate to assess generative capabilities of GAN models trained on image generation, it is not suited for physical data, in Preuer et al. [2018] the authors propose a method to adapt the Fréchet Inception Score for chemical data by swapping the Inception model by a Chemb1Net model which was trained on molecular data. In Obukhov and Krasnyanskiy [2020] authors propose methods to adapt FID for generic Generative Adversarial Network by swapping Inception model with an adequate classifier depending on the distribution being learnt. To our knowledge, no prior work has been carried on for adapting the FID for generative models training on physical data.

2.5 Uncertainty quantification for physical models

The aim of this section is to present the state of the art methods on uncertainty propagation for physical models. In order to present a synthesis and clearly determine non-parametric approaches,

one has to define the differences between aleatory and epistemic uncertainty. Which are stated in Batou et al. [2015] & You et al. [2020] :

- Aleatory uncertainties : the uncertainties relative to some model parameters induced by the lack of knowledge related to those parameters. To process these uncertainties, parametric approaches are used as the modelization of the uncertainty of the parameters by random variables and fields in order to construct stiffness and mass matrices with respect to those parameters.

- Epistemic uncertainties : also arises from lack of knowledge on parameters but based on subjective perception, imperfect modeling, and limited data availability, such as interval analysis, possibility theory, and fuzzy set theory. Parametric approaches are not suited for this application.

Both these types of uncertainty are listed as model parameter uncertainty in Batou et al. [2015] and also introduce a new type of uncertainty :

- Modeling Error : the uncertainties induced by the modeling errors within the choice of the physical model.

Epistemic and modeling errors cannot be processed by fully parametric approaches. Non-parametric and mixed approaches (see Batou et al. [2015]) are necessary such as :

- Probabilistic approach : random matrix theory (see Adhikari and Chowdhury [2010], Adhikari and Chowdhury [2010], Guedri et al. [2012], Reynders et al. [2014], Murthy et al. [2012], Wang et al. [2020] & Legault et al. [2012]).

- Possibilistic approach : Fuzzy variables & interval analysis (see You et al. [2020]).

In the following a detailled presentation of some of the state of the art in uncertainty quantification for physical data is proposed. In Adhikari and Chowdhury [2010] a new reduced-order non-parametric computational approach for damped stochastic linear dynamical systems for the analysis of uncertainty of very large dynamical systems over a wide range of frequency is proposed using Wishart random matrix distribution (as introduced in Soize [2005], relies on direct Monte Carlo Simulation). The equation of a damped n-degree-of-freedom linear dynamic system can be written as :

$$M\ddot{q}(t) + C\dot{q}(t) + Kq(t) = f(t) \tag{2.70}$$

The purpose of the work in Adhikari and Chowdhury [2010] is to look for a fast simulation approach for obtaining frequency response function (FRF) statistics using Wishart matrices. The method is based on stochastic system transformation and reduction in the modal domain.

The dispersion parameter proposed by Soize [2005] for the construction of the random substitues of M and K :

$$\delta_K = \frac{\mathbb{E}[\|K - K_0\|_F^2]}{\|K_0\|_F^2} \tag{2.71}$$

Where δ_k, δ_M, K_0 and M_0 are the parameters (mean and dispersion) for random matrices M, K.

The dynamic response of a proportionally damped stochastic system is characterized by the eigensolutions of :

$$H = M^{-\frac{1}{2}} K M^{-\frac{1}{2}} \tag{2.72}$$

where $H \sim \mathbb{W}_n(p,\Sigma)$. The major goal of the study in Adhikari and Chowdhury [2010] is to design a Reduced Wishart technique, which is an effective reduced computing approach for large dynamical systems. Considering the Laplace transformation of the equation of motion :

$$[s^2M + sC + K]\bar{q}(s) = \bar{f}(s) \tag{2.73}$$

the aim is to obtain statistical properties of $\bar{q}(s) \in \mathbb{C}^n$ when the system is formed with random matrices.

We obtain the undamped eigenvalues problem :

$$j \in [|1,m|] \ , \ K\Phi_j = \omega_j M\Phi_j \tag{2.74}$$

Where $\Omega = diag([\omega_1,...,\omega_m]) \in \mathbb{R}^{m\times m}$ and $\Phi = [\Phi_1,...,\Phi_m] \in \mathbb{R}^{n\times m}$ are the truncated modal matrices. The modes are then only computed by solving the reduced order eigenvalue problem for each realization of K and M. Since $\Phi^T K\Phi = \Omega^2$, $\Phi^T M\Phi = I_m$ and Ω^2 being symmetric define positive : $\Omega_r^2 = \Psi^T\Omega^2\Psi$ and the damping is supposed proportionally modal i.e : $C = \Phi^T C\Phi = 2\varepsilon\Omega$

One can deduce :

$$\bar{q}(s) = \Phi\Psi[s^2I_m + 2s\varepsilon\Omega + \Omega^2]^{-1}\Psi^T\Phi^T\bar{f}(s) \tag{2.75}$$

$$\bar{q}(s) = \sum_{j=1}^{m} \frac{X_j^T\bar{f}(s)}{s^2 + 2s\varepsilon_j\omega_j + \omega_j^2}X_j \tag{2.76}$$

where $X = \Phi\Psi$.

Monte Carlo simulation is used to calculate physical statistical information after selecting the number of modes to analyze and the degree of freedom of interest. The suggested method is integrated with commercial finite element software by taking use of its non-intrusive nature.

Otherwise, epistemic uncertainties are due to a lack of accurate knowledge concerning the physical laws governing the behavior of a component or interface and can generally be reduced with a combination of more detailed modeling and experimental investigations.

In Guedri et al. [2012], is proposed to examine the robustness of a classical reliability analysis with respect to both aleatory and epistemic uncertainty.

For reliability analysis the aim is to approach the failure probability or its density :

$$p_f = p[g(x) \leq 0] = \int_{g(x)\leq 0} f_x(X)\delta x \tag{2.77}$$

where g is the state function. Since epistemic uncertainty is limited to parametric models, a completely parameterized modelization would be required in the presence of epistemic uncertainty. Parameters are considered random variables : $[X_1,...,X_n]$ to be sampled as well as the matrices (or within) that form the stochastic model.

Considering the transfer function of a dynamic system :

$$H(\omega) = [-\omega^2M + i\omega C + K]^{-1} \tag{2.78}$$

Random matrices are defined as substitutes of M, C and K as symmetric define positive random matrices as shown before : $G = L_n^T L_n$ where L_n is an upper triangular matrix formed by positive random variables following positive valued gamma law.

$$K = \bar{L}_k^T Gk\bar{L}_K. \tag{2.79}$$

Where L_k being the upper triangle matrix obtained by the Cholesky factorization of $\bar{K} = \mathbb{E}[K]$

Multiple random samples of these matrices are created and then used to calculate the system's response using:

- Direct Monte Carlo Simulation techniques :

$$p_f = \frac{N_f}{N} \tag{2.80}$$

  where N_f is the number of failure samples and number of accumulated samples. To estimate the precision of the technique, coefficient of variation :

$$COV(p_f) = \sqrt{1 - \frac{p_f}{N p_f}} \tag{2.81}$$

- Importance sampling : parameters that have more impact are sampled more.

$$p_f = \frac{1}{N} \sum_{i=1}^{n} I_f \frac{f_x[X_{1,i}, ..., X_{n,i}]}{h_x[X_{1,i}, ..., X_{n,i}]} \tag{2.82}$$

  where f_x is the original density, h_x the importance density and I_f a failure indication where $I_f = 0$ if failure, and $I_f = 1$ for survival.

While the uncertainty quantification of physical numerical models is quite straightforward, applying same methods to experimental data is less intuitive. Batou et al. [2015] deals with the identification of a stochastic computational model using experimental eigenfrequencies and mode shapes.

This relationship is established in this study by applying a modified transformation for the calculated modal values. The converted computed modal values may then be compared with the experimental data to determine the stochastic computational model's parameters.

The purpose of the work in Batou et al. [2015] is to determine the hyperparameters of an Stochastic Computational Model (SCM) using the observed natural frequencies and mass-normalized mode shapes of a family of structures. In order to match the computational observations (computational eigenfrequencies and computational mode shapes) with the experimental observation of each measured structure, the proposed method employs a random transformation of the computational observations (computational eigenfrequencies and computational mode shapes). In this study, uncertainty in model parameters and uncertainty in the model itself are addressed. For model parameter uncertainty, random variables are substituted for parameter values.

Let $M(h)$, $K(h)$ the mass and stiffness matrices depending on a set of parameters h. h is replaced by a random vector H. Then random matrices stubstitues of M(H) and K(H) are constructed with respect to Soize's methodology on the probabilistic space of H. Then to account for model uncertainties, new random matrices are built to substitute those for parameter uncertainty, on the joint probabilistic space of the model and of H.

$$\forall \theta = (\theta^{par}, \theta^{mod}) \in (\Theta^{par} \times \Theta^{mod}), \quad M^{tot}(\theta) = L_M(\theta^{par})^T G_M(\theta^{mod}) L_M(\theta^{par}) \tag{2.83}$$

Those matrices being built, and an identification process with the different experimentations is presented: For each experiment (configuration of structure) modes are identified Φ^{exp_j}. Projection operator from DOF of statistical model to the DOF of the experimentations : $\bar{\Phi}^j = P^j \Phi$. The experimental modes cannot directly be compared to the computational mode (no correspondence).

Then the ROM is built as:

$$\phi'^{j} = \bar{\phi}^{j} Q^{j}_{opt} \tag{2.84}$$

with

$$Q^{j}_{opt}(\theta) = argmin_{Q \in \mathscr{O}ST(n,n_j)} \|\bar{\Phi}^{j}(\theta)Q - \Phi^{exp_j}\|. \tag{2.85}$$

Then parameters are fitted using maximum likelihood method.

For structures with both random and fuzzy uncertainty, You et al. [2020] presents a novel method for determining the membership function in fuzzy reliability with the Automatic Updating Extreme Response Surface (AUERS) method, this method is then suitable for sizing optimization problems since it consists in optimization in extremums. Analysis of reliability includes establishing aleatory uncertainty with a limit state function, which is evaluated as the likelihood that a structural response exceeds a threshold limit. Automatic Updating Extreme Response Surface (AUERS) is suggested in this study as a double loop system that combines the advantages of PSO and NERS. Aleatory uncertainties are modeled using random variables, while epistemic uncertainties are modeled with membership functions or intervals for possible outcomes.

Let $G(x,\bar{x})$ be the unit state function, where $x = [x_1,...,x_m]$, $\bar{x} = [\bar{x}_1,...,\bar{x}_m]$ respectively being independent random variables and independent fuzzy variables. Considering a cut of size α of the interval $x_\alpha \in [x^L_\alpha, x^U_\alpha]$ then the failure probability $p_{r\alpha} = p[G(x_\alpha > 0] \in [p^L_{r\alpha}, p^U_{r\alpha}]$.

Those values are then determined using

$$\mu_{p_r}(p_r) = \mu_x(g^{-1}(p_r)) \tag{2.86}$$

under each cut of the membership intervals, where μ are the respective membership functions.

Having :

$$p^L_{r\alpha} = p[G(x, \theta^L_{extreme})] \tag{2.87}$$

where $\theta^L_{extreme}$, $\theta^U_{extreme}$ are extreme points on the membership interval where the structure exhibit minimum and maximum response. Those points are determined using a PSO algorithm to optimize the limit state function.

A kriging model is built :

$$\theta_{\alpha extreme}(x) = f^T(x)\gamma + Q(x). \tag{2.88}$$

Where γ is the regression coefficient and $Q(x) \sim \mathscr{N}(0,\sigma^2)$

$Cov(Q(x_i),Q(x_j) = \sigma^2 R(x_i,x_j)$ with $R(x_i,x_j) = e^{[-\Sigma^N_{p=1}\omega_p(x^i_p - x^j_p)^2]}$, ω being the parameter of the random variables x. These parameters are determined with a maximum likelihood method.

After identifying the most extreme point in each sample group, the kriging extreme response surface is created at the current cut level. In addition, the Root Mean Squared Percentage Error (RMSPE) is employed in each inner loop to determine if it might fall into the outer loop, which then enters the FORM-based reliability calculation procedure. If the RMSPE is not met, the random sample points will be added and the kriging extreme response surface will be re-fitted, so completing the automated updating cycle.

In addition, work for uncertainty propagation in a wave propagation setup has been carried on in Reynders et al. [2014] where a stochastic method for sound transmission loss prediction through a wall in between two rooms, consisting of a hybridization of displacement-based and energy-based modeling, has been presented. It is particularly appropriate for mid-frequency analysis, and the condition in which the wall or floor is modally sparse has been studied in detail. Comparatively to deterministic approaches such as finite element analysis or the wave-based method, this method is computationally inexpensive since rooms are represented in a highly efficient, non-parametric

stochastic manner, as in statistical energy analysis. In the SEA subsystems, the approach automatically adjusts for spatial differences in geometry, material qualities, and boundary conditions that have an influence on wave scattering. It results in transmission loss estimates that are resistant to such variations: not only are mean values and variances obtained, but also standard deviations. The uncertainty created by random wave scatters is significant unless the modal overlap across rooms is quite large. As was shown for a perforated brick wall with an unknown damping loss factor, additional parametric uncertainty may be compensated for in a straightforward manner if required. The approach was thoroughly tested against a comprehensive modal model of a transmission suite with random acoustic mass distribution as well as laboratory data for progressively complicated wall systems. Within its range of validity, the hybrid technique demonstrated outstanding predictive performance in every instance.

Equation of motion:

$$Dq = f \tag{2.89}$$

with q being the amplitude vector for response and f load amplitude vector, and D dynamic stiffness matrix, that can be decomposed as:

$$D = D_d + \sum_{k=1}^{n} D_k \tag{2.90}$$

with D_d stiffness matrix associated with master system and D_k the matrix associated with the k^{th} subsystem. Each of this matrices can be decomposed to a deterministic component and a random component: $D_k = D_d^k + D_{ran}^k$ where $D_d^k = \mathbb{E}[D_k]$. Thus the equation of motion for the subsystem: $D_d^k q = f_k + f_{ran}^k$ where the reverberant force are defined as $f_{ran}^k = -D_{ran}^k q$. Then for the whole system :

$$D_{tot} q = f + \sum_{k=1}^{n} f_{ran}^k \quad , \quad D_{tot} = \mathbb{E}[D]. \tag{2.91}$$

Thus the mean of energy for every subsystem :

$$\mathbb{E}[P_j] = \omega(\eta_j + \eta_{dj})E_j + \sum \omega \eta_{jk} \eta_j (\frac{E_j}{\eta_j} - \frac{E_k}{\eta_k}). \tag{2.92}$$

And then variance of response is computed the same so as to compute the mean of transmission loss and its variance to compare it to values obtained from experiences.

Whereas in Murthy et al. [2012] authors presents the initial findings of a combined experimental and computational study focused on the validation of a reduced order model of geometrically nonlinear structures in the presence of uncertainty. This technique to model validation is predicated on the notion that the model is valid if the experimental data can be interpreted as random sample responses of the stochastic system whose mean is the reduced order model. So, random matrices are generated for stiffness and mass matrices constructed by maximization of the statistical entropy :

$$A = \bar{L} H H^T \bar{L}^T \tag{2.93}$$

where $\bar{L}$ is the results of Cholesky decomposition of the mean $\bar{A}$. And H is a lower triangular matrix whose below diagonal element are all independent zero mean Gaussian variables. And $H_{ii} = \sqrt{\frac{Y_{ii}}{\nu}}$ where Y_{ii} is Gamma distributed $\frac{(p(i)-1)}{2}$ and $p(i) = n - i + 2\lambda - 1$, $\nu = \frac{n+2\lambda-1}{2}$, λ being the hyperparameter to fit. The research proposes a way to extract stiffness matrix from reduced model then construct the stochastic model with regard to the extracted matrix then utilize the stochastic

model in running Monte Carlo simulations. The aim of this study is the direct incorporation of uncertainty into reduced order models of the nonlinear geometric response of structures based on maximum entropy ideas. In addition, in Wang et al. [2020] the authors give more details of the tuning performed to turn the system matrix extracted from the reduced model of the experiments into a definite positive matrix. Tuning that may introduce additional error to the system since it modifies the physical law. In case this tuning turns to be impossible an LDL^T decomposition based analysis is proposed :

$$\bar{K}_B = \bar{L}_k D \bar{L}_k^T \quad , \quad K_B = \bar{L}_K H D H^T \bar{L}_K^T. \tag{2.94}$$

One of the main claims of the non parametric model of random uncertainty introduced by Soize [2005] is its ability to account for model uncertainty. Legault et al. [2012] investigates this claim by examining the statistics of natural frequencies, total energy and underlying dispersion equation yielded by the non parametric approach.

In general, models of uncertainty will be based on a parametric or nonparametric representation of uncertainty, or perhaps both. A parametric description of uncertainty indicates that the parameters of the dynamical model (material qualities, dimensions, etc.) are seen as uncertain variables that may be defined statistically, by intervals, by fuzzy numbers, or by any other uncertainty model. Various approaches, such as Monte Carlo simulations or the stochastic finite element method, are then used to propagate uncertainty via the equations of motion. A non-parametric account of uncertainty, on the other hand, believes that the system's uncertainties can be characterized using some form of universal uncertainty model, independent of their specific nature. Stiffness K, and mass matrix M are defined as proposed by Soize.

Therefore, if the matrices K and M are translated back into physical coordinates, the random matrices K and M associated with them will be completely filled. The randomization process will couple degrees of freedom that were not initially coupled because they are not spatially adjacent (degrees of freedom for which the associated coupling matrix entry is zero). It will be demonstrated that the presence of this long-range coupling has significant physical consequences. This raises problems regarding the analyst's past expectations for the system, despite the fact that the specifics are obscured by unknown model uncertainty. If it can be shown that the nominal model reflects the mean of the unknown system matrices, and if this is the only available prior knowledge, then it is logical to use the nonparametric method. The concept of maximal entropy suggests that in such a situation, taking such a stance would be the most noncommittal option. Nonetheless, if the available mean information is not considered to be in the matrix space, but rather in the response space, in the sense that the mean modal density and modal norm of the system should be centered on the asymptotic values, then the nonparametric approach, in its current form, leads to results that may be at odds with these expectations (even though the deviations from the asymptotic values may become significant only at high values of uncertainty). If the information that the mean modal density and modal norm of the system should be centered around the asymptotic values was considered to be known in addition to the mean matrices of system, then this knowledge might in theory impose extra restrictions in the maximum entropy computation. Estimation of total energy in the system is computed (as in Soize [2005]) and another useful quantity, the statistical overlap factor :

$$S(\omega) = \frac{2var(\omega_n)^{\frac{1}{2}}}{\mu(\omega)} = \frac{2\mathbb{E}[(\omega_n - \bar{\omega}_n)^2]^{\frac{1}{2}}}{\mu(\omega)}. \tag{2.95}$$

where $\mu(\omega)$ is the average frequency spacing between consecutive modes. These quantities as modal density are computed using Monte Carlo simulation techniques.

In Farhat et al. [2018] is presented a projection based model order reduction method on a

parametric problem denoted :

$$K(\rho)u(\rho) = \omega(\rho)^2 M(\rho)u(\rho) \tag{2.96}$$

Where K is the stiffness matrix and M the mass matrix of the problem. Then a projection based model order reduction is performed by compressing the snapshot matrix collected using an SVD method to construct reduced K_r and M_r. The problem is then projected using the eigenvectors (V):

$$K_r(\rho)u_r(\rho) = \omega(\rho)^2 M_r(\rho)u_r(\rho) \tag{2.97}$$

where $K_r = V^T KV$, $M_r = V^T MV$.

The problem is then solved for u_r then u is deduced using $u = Vu_r$. The limitation of this approach is that it is only expected to be accurate on the parameter point the snapshot matrix was built on and possibly its immediate neighborhood. Then a greedy method which relies on sampling a certain number of parameters point to build a generalized basis is presented, to overcome the previous limitations. The efficiency of this approach is relative to the size of the problem, for a parameter domain relatively small where a uniform sampling of a Latin hyper cube is possible this approach is feasible. Finally a stochastic approach is presented where missing information about the deterministic model are extracted from the data and infusing into a stochastic lower-dimensional counterpart to adapt said model, being a data-driven model adaptation.

2.6 Bayesian techniques

Bayesian inference is the domain of statisitics where we would like to predict and compute the true probability distribution of uncertain parameters or latent representative features associated with observable data. This prediction can then be used in the approximation of uncertainty and the error within the observable fields, which are associated with random parameters of mechanical models or modeling choices or epistemic parameters. In this field we may distinguish also between parametric and non-parametric methods. The empirical Bayesian method Casella [1985] is a parametric one that consists in finding hyper-parameters of statistical models that describe random parameters or latent representative features that might be solutions of a reduced order model. This method is based on the maximization of the logarithm-likelihood of the observation data and the application of the Bayes theorem Adamson [2012]. The variational Bayesian method Blei et al. [2017] is a non-parametric approach for finding the true posterior of random parameters or epistemic parameters or representative latent features associated with observable data. It is based on the maximization of the well known Evidence Lower BOund (ELBO) which is a natural lower bound of the logarithm marginal likelihood Kullback [1997]. It is obvious that for random parameters of known probability distributions, the use of the parametric Bayesian empricial approach is more robust. However, the use of the variational Bayesian technique may be more efficient. In this case, a trade-off between efficiency and sharpness needs to be investigated. However, for epistemic parameters and representative features, the use of the variational Bayesian method appears to be more convenient as the true probability distribution is completely unkown in this case and not only its hyper-parameters. In all cases, the knowledge of the true posterior distribution of the parameters or the latent features of observable data and an approximation of the conditional probability distribution of the observable data allow to perform efficiently a Monte Carlo of the whole function with respect to the parameters or the latent features and, to deduce the probability distribution of this function or quantities of interest from this function with respect to the parameters of interest

or the latent features. Therefore, following this philosophy, GANs can be seen as a non-parametric Bayesian approach as the generator of the GAN is defined on a Gaussian multivariate distribution allowing to perform efficiently a Monte Carlo of the whole function with respect to the parameters of interest.

Chapter 3

Comparison between POD and CNN for regression in explicit dynamic

Abstract

In the following chapter, a benchmark of different non-intrusive model reduction approaches is performed on an explicit dynamic contact $3D-$ problem. The main purpose of this chapter is to evaluate the stability of the reduced model with respect to time along with the precision of these approaches with respect to the true solutions of interest, which are the displacement and velocity fields. The precision of these approaches is also evaluated with respect to the evolution of some materials parameters. Six parameters vary in this study and we would like to predict the whole transient fast dynamic impact response with respect to each parameter. To this end, several models are trained: Proper Orthogonal Decomposition (POD) and Deep convolutional Neural Network (DcNN), in addition, a vectorized version of Interpolation in Grassman Manifolds is proposed. The benchmark performed illustrates that using DcNN's allows achieving the best precision and stability in predicting physical fields.

Résumé

Dans le présent chapitre, un benchmark de différentes approches de réduction de modèle non intrusives est réalisé sur un problème de contact dynamique explicite en 3D. Le principal objectif de ce chapitre est d'évaluer la stabilité du modèle réduit par rapport au temps ainsi que la précision de ces approches par rapport aux véritables solutions d'intérêt, qui sont les champs de déplacement et de vitesse. La précision de ces approches est également évaluée par rapport à l'évolution de certains paramètres de matériaux. Six paramètres varient dans cette étude et nous souhaitons prédire l'ensemble de la réponse d'impact dynamique transitoire rapide par rapport à chaque paramètre. À cette fin, plusieurs modèles sont entraînés : la décomposition en bases orthogonales (POD), des réseaux de neurones convolutifs profond (DcNN), et une version vectorisée de l'interpolation dans les variétés de Grassman est proposée. Le benchmark réalisé illustre que l'utilisation des DcNN permet d'obtenir la meilleure précision et stabilité dans la prédiction des champs physiques.

Contents

3.1 Introduction 50
3.2 Problem Definition 51
3.3 Methodology 52
3.3.1 parameterized POD 52
3.3.2 Parameterized Space-Time POD 53
3.3.3 Parameterized Space-Time POD: Interpolation on Grassmann Manifolds 53
3.3.4 Deep convolutional neural regressor 56
3.4 Numerical Examples 57
3.4.1 Data Sampling 57
3.4.2 Data preprocessing 58
3.4.3 Trained metamodels 59
3.4.4 Results 59
3.5 Conclusion 66
3.6 Supplementary results 66

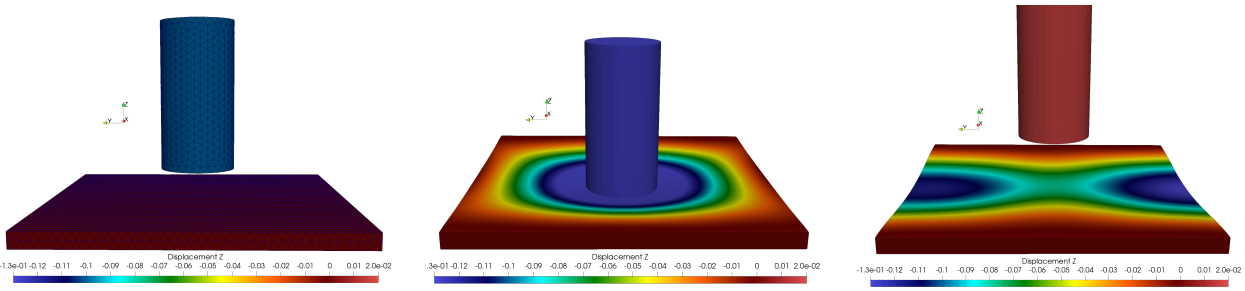
3.1 Introduction

A large number of fast dynamic impact numerical simulations is crucial for optimization or uncertainty quantification and propagation problems, in order to study the influence of different scenarios and regimes in engineering problems. Dimension reduction approaches such as the Singular Values Decomposition Muller et al. [2004] and model reduction techniques such as non-linear regressions based on Gaussian Process Regression Ranftl et al. [2021] or Deep Learning LeCun et al. [2015], offer attractive alternatives to finite element impact problems in order to reduce the computational cost in these multiple resolutions demanding tasks. Model Order Reduction based on the projection of the continuous High-Fidelity Partial Differential Equations (PDEs) on a reduced basis such as the one obtained by the Proper Orthogonal Decomposition Chatterjee [2000], is also a very important complementary option to finite element impact problems. These model order reduction approaches were considered in Balajewicz et al. [2016b] where a Non-Negative Matrix Factorization is used to construct a reduced basis with the inequality constraints describing the contact forces. A nodeless superelement approach was proposed in Géradin and Rixen [2016] for the dual problem of the Lagrange multipliers, where the superlement dynamic is described by a set of orthogonal static modes. In Giacoma et al. [2014], a multi-scale reduced order approach based on the LATIN method Ladevèze et al. [2010] and a mutli-grid solver is proposed for frictional contact problems. In Benaceur et al. [2020], the reduced basis method is applied for parameterized variational inequalities in contact mechanics.

In the following work, we are only interested in non-intrusive and non-projection-based model reduction approaches, i.e.without performing a projection of the continuous PDEs on a reduced basis. Non-intrusive approaches rely only on data driven method of modeling without using any physical information in the construction or the training of the reduced model, whereas intrusive methods use all the physical information available such as the exact PDE solved to construct their reduced model by projecting the equations on lower dimension spaces or constraining models with

the equations in the training phase. The main motivation is the simplicity of the numerical framework in this case. We would like to compare different algorithms for constructing these reduction tasks, by resorting to non-linear regressions by Kriging, interpolation on Grassmann Manifolds, and Deep Learning. More precisely, the Gaussian Process Regression and the Grassmann interpolation are applied to predict the coefficients in the linear combination of Proper Orthognal Decomposition modes as an approximation of the fields of interest. As for the Deep Learning method for the non-linear regression, it is performed thanks to a deconvolutional neural network Zeiler et al. [2010b] as an approximation of the mapping between the parameters of interest and the fields of interest.

3.2 Problem Definition



(a) Before impact (b) Impact (c) After impact

Figure 3.1: Main steps of the simulation

In this chapter we investigate the reducibility of a contact case in structural dynamics. We consider a cylinder of height $H_{cylinder}$ and diameter $D_{cylinder}$ which comes into contact with a plate, a rectangular cuboid of length, height and width $L_{plate}, W_{plate}, H_{plate}$ with an initial velocity v_z along the z-axis. Our objective is to study wave propagation and reflection phenomena on the plate following the shock over a discretized time grid denoted $T = (t_k)_{k\in[|1,n|]}$ with a time step Δt. This analysis requires the constructions of models to generate new parameterized simulations with a lower cost than solving the governing physical equations. Therefore we choose to apply data-driven methods such as metamodeling and deep learning models. Figure 6.1 shows the main steps of the finite element method simulation. Both solids are considered formed by uniform elastic and isotropic materials behaving linearly following the Hooke's law, which is written in the matrix form as

$$\sigma = \frac{E}{1+\nu}\left(\varepsilon + \frac{\nu}{1-2\nu}\mathrm{Tr}(\varepsilon)\right), \tag{3.1}$$

where σ and ε are respectively the second-order stress and strain tensors, E is the Young modulus, ν is the Poisson ratio, and Tr is the trace operator. Additionally, we denote the density of the materials by ρ, which plays a part in the response of the plate in such transient dynamical regimes. We introduce a unique parametrization of each simulation results :
let $\theta = (E_{cylinder}, \rho_{cylinder}, \nu_{cylinder}, E_{plate}, \rho_{plate}, \nu_{plate})$ a parameters vector that defines a unique finite element simulation denoted $S(\theta)$, S being the vector of the fields values on the grid of the plate. The objective here is to construct regression models which function is to map the parameters vector θ to the simulation vector $S(\theta)$. As physical fields, we considered both displacement and

velocity of the elements of the plate on the z-axis (u_z, v_z), both will be referred to as S in what follows.

Since both solids in contact have linear material and kinematic behavior, and we consider only the case of frictionless, adhesive-free normal contact, and considering a space discretization of both solids relying on the Finite Element discretization, the dynamic contact problem can be written as (see Balajewicz et al. [2015]) :

$$\begin{cases} M\ddot{u} + Ku = B^T\lambda \\ Bu - c \leq 0 \\ \lambda(Bu - c) = 0 \\ \lambda \geq 0 \\ u(0) = u_0 \\ \dot{u}(0) = \dot{u}_0. \end{cases} \tag{3.2}$$

Where a dot designates a time derivative, the first equation expresses the dynamic equilibrium of the two solids, the inequality constraint derives from the space discretization of the Hertz-Signorini-Moreau contact conditions which are enforced by introducing dual Lagrange multipliers. u is a space discretized displacement vector of the plaque and the cylinder degrees of liberty, M and K are respectively the block diagonal mass and stiffness matrices for the two solids. B is a signed boolean matrix which extracts from u the pair of dof governed by a contact condition, c is the vector of initial clearances, and λ is the vector of descretized Lagrange multipliers.

Then an explicit time discretization is performed in the dynamic case, finally solving the dynamic problem can be written as:

$$\forall n \geq 2, (u_n, \lambda_n) = \arg\min_{v,\mu} v^T(\frac{1}{\Delta t^2}M + K)v + \frac{1}{\Delta t^2}v^T M(-2u_{n-1} + u_{n-2}) - \mu^T(Bv - c) \tag{3.3}$$

Where the superscript n designates the n-th time-step and Δt designates the difference between two consecutive time-steps.

Both solids are discretized in finite element meshes using 3D tetrahedric elements, 8546 nodes combined between the two solids, around 5000 nodes for the plaque mesh.

3.3 Methodology

The present Section describes the different methodologies used in this work.

3.3.1 parameterized POD

In this Subsection, we recall the classic parameterized POD strategy for field prediction. A dataset of N samples is considered: $\{\theta_i, \mathbf{y}_i\}_{i\in[\![1,N]\!]}$, where $\theta_i \in \mathbb{R}^d$ and $\mathbf{y}_i \in \mathbb{R}^n$ represent respectively a vectorial parameter of moderate dimensionality and a high-dimensional output. The objective is to build a predictive model $\theta \to \tilde{\mathbf{y}}(\theta)$. The methodology is decomposed in the following steps:

- Orthonormal basis generation. The data matrix $M = [\mathbf{y}_1, ..., \mathbf{y}_N] \in \mathbb{R}^{n\times N}$ is considered. In practice, note that the number of samples N is largely below the output dimensionality n. Its

SVD decomposition yields

$$M = U\Sigma V^T, \tag{3.4}$$

with $U = [\boldsymbol{u}_1, ..., \boldsymbol{u}_N] \in \mathbb{R}^{n\times N}$ containing the spatial orthonormal eigenvectors satisfying $\boldsymbol{u}_i^T \boldsymbol{u}_j = \delta_{ij}$, the eigenvalues matrix $\Sigma = \text{Diag}(\sigma_1, ...\sigma_N)$ and $V = [\mathbb{V}v_1, ..., \mathbb{V}v_N] \in \mathbb{R}^{N\times N}$, $\mathbb{V}v_i^T \mathbb{V}v_j = \delta_{ij}$.

Another strategy implies the covariance matrix eigen-decomposition: $C = M^T M = V\Sigma^2 V^T$, with $\boldsymbol{u}_i = \dfrac{M\mathbb{V}v_i}{\sigma_i}$.

- Number of modes selection. Given an accuracy threshold ε, the number of modes r selected is the smallest integer satisfying:

$$\frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{N} \sigma_i^2} > 1 - \varepsilon. \tag{3.5}$$

The prediction is then sought in the span of the reduced orthonormal basis $\mathscr{U}_r = [\boldsymbol{u}_1, ...\boldsymbol{u}_r]$.

- Scalar coefficients in the reduced basis. The prediction is expressed as

$$\tilde{\mathbf{y}}(\theta) = \sum_{i=1}^{r} h_i(\theta)\boldsymbol{u}_i, \tag{3.6}$$

with the scalar functions $\theta \to h_i(\theta)$ satisfying

$$h_i(\theta_k) = < \mathbf{y}_k, \boldsymbol{u}_i >, \tag{3.7}$$

$<,>$ being the canonical scalar product in $\mathbb{R}^n$.
Each of the coefficients h_i is trained using a regression method (Kriging, RBF, ANN, random forest,...) based on the DoE $\{\theta_k, < \mathbf{y}_k, \boldsymbol{u}_i >\}_k$, yielding a predictive function $\theta \to \tilde{h}_i(\theta)$.

- Prediction. The sought predictive model finally reads

$$\tilde{\mathbf{y}}(\theta) = \sum_{i=1}^{r} \tilde{h}_i(\theta)\boldsymbol{u}_i. \tag{3.8}$$

3.3.2 Parameterized Space-Time POD

In the context of space-time POD, the dataset is given as follows: $\{\theta_i, S_i\}_{i\in[\![1,N]\!]}$, where $\theta_i \in \mathbb{R}^d$ and $S_i = [y(\theta_i, t_1), ..., y(\theta_i, t_m)] \in \mathbb{R}^{n\times m}$. Each matrix S_i contains the snapshots for all time steps. For this case, we propose to consider the time as an independent parameter, and recast the dataset to fit the framework of classic parameterized POD (Subsection 3.3.1: $\{\hat{\theta}_k, \hat{\mathbf{y}}_k\}_{k\in[\![1,mN]\!]}$ where for each $k \in [\![1, mN]\!]$ corresponds to a couple of integers $(i,l) \in [[\![1,N]\!]] \times [[\![1,m]\!]]$, with $\hat{\theta}_k = (\theta_i, t_l)$ and $\hat{\mathbf{y}}_k = y(\theta_i, t_l)$. The previous framework is then used for training and prediction.

3.3.3 Parameterized Space-Time POD: Interpolation on Grassmann Manifolds

Friderikos et al. proposed a framework for parameterized space-time POD based on interpolation on Grassmann manifolds Friderikos et al. [2021], suitable for scalar parameters. The

underlying idea is to interpolate POD basis associated with a limited number of training points on Grassmann manifolds, so as to obtain a robust non-intrusive ROM corresponding to a target parameter, separating both reduced spatial and temporal basis. A methodology to extend this algorithm for vectorial input parameters is proposed. The main steps of the algorithm are summarized in Subsubsection 3.3.3. The proposed modifications for the extension to the vectorial case is given in Subsubsection 3.3.3.

Scalar version (from Friderikos et al. [2021])

- *Dataset* : Similarly to Subsection 3.3.2, the dataset is given as follows: $\{\theta_i, S^{(i)}\}_{i\in[\![1,N]\!]}$, where $\theta_i \in \mathbb{R}$ and $S_i = [y(\theta_i,t_1),...,y(\theta_i,t_m)] \in \mathbb{R}^{n\times m}$. This version is coined 'scalar' because it is restricted parameters θ taken as real numbers only.

- *Set a number of modes r* : We propose the following strategy to set the number of modes r. We fix a given threshold accuracy $\varepsilon = 10^{-12}$, then perform a SVD decomposition on the mean output matrix. The number of modes r is then chosen according to the eigenvalues decay, similarly to POD (Subsection 3.3.1)

$$\bar{S} = \frac{1}{N}\sum_{i=1}^{N} S^{(i)} = \bar{U}\bar{\Sigma}\bar{V}^T, \tag{3.9}$$

  with $\bar{\Sigma} = \mathrm{Diag}(\sigma_1,...,\sigma_m)$. r is the smallest integer satisfying:

$$\frac{\sum_{i=1}^{r}\bar{\sigma}_i^2}{\sum_{i=1}^{m}\bar{\sigma}_i^2} > 1-\varepsilon. \tag{3.10}$$

- *Oriented SVD for each output matrix, with mode extraction of order p* : For each output matrix $S^{(i)}$, an oriented SVD Friderikos et al. [2021] decomposition is performed:

$$S^{(k)} = \Phi^{(k)}\Sigma^{(k)}\Psi^{(k)T}. \tag{3.11}$$

  The r first modes are extracted, leading to matrices $\Phi_r^{(k)} \in \mathbb{R}^{n\times r}, \Sigma_r^{(k)} \in \mathbb{R}^{r\times r}, \Psi_r^{(k)} \in \mathbb{R}^{m\times r}$. The following reconstructed matrix output is also evaluated:

$$S_r^{(k)} = \Phi_r^{(k)}\Sigma_r^{(k)}\Psi_r^{(k)T}, \tag{3.12}$$

  with $k \in [\![1,N]\!]$.

- *Interpolation on compact Stiefel manifolds* : Using the datasets $\{\theta_k, \Phi_r^{(k)}\}_{k\in[\![1,N]\!]}$ and $\{\theta_k, \Psi_r^{(k)}\}_{k\in[\![1,N]\!]}$, define a map $\theta \to \tilde{\Phi}(\theta) \in \mathbb{R}^{n\times r}$ and (resp.) $\theta \to \tilde{\Psi}(\theta) \in \mathbb{R}^{m\times r}$. Let us denote by $\{\theta_k, Y_k\}_{k\in[\![1,N]\!]}$ such a generic dataset of orthonormal basis, assuming $Y_k \in \mathbb{R}^{n\times r}$. A set of matrices $\{Z_k\}_k$ is first evaluated:

$$\begin{cases} Z_1 = 0_{\mathbb{R}^{n\times r}}, \\ Z_k = U_k \tan^{-1}(\Sigma_k)V_k^T, \quad k>1, \end{cases} \tag{3.13}$$

  where the orthonormal matrices $U_k \in \mathbb{R}^{n\times r}$, $V_k \in \mathbb{R}^{r\times r}$ and the diagonal matrix $\Sigma_k \in \mathbb{R}^{r\times r}$

are computed by performing the SVD on the following matrix:

$$Y_k(Y_1^T Y_k)^{-1} - Y_1 = U_k \Sigma_k V_k^T.$$

An interpolator $\theta \to Z(\theta)$ using linear Lagrange polynomial basis functions $(L_i)_{i\in[|1,N|]}$, suitable for scalar parameters, is then defined, as well as $\theta \to U(\theta), \Sigma(\theta), V(\theta)$ obtained by applying a SVD decomposition on $Z(\theta)$:

$$Z(\theta) = \sum_{i=1}^{N} L_i(\theta) Z_i = U(\theta)\Sigma(\theta)V^T(\theta), \tag{3.14}$$

The final interpolator $\theta \to \tilde{Y}(\theta)$ finally reads:

$$\tilde{Y}(\theta) = [Y_1 V(\theta)\cos(\Sigma(\theta)) + U(\theta)\sin(\Sigma(\theta))]V(\theta)^T. \tag{3.15}$$

- *Square matrix of the mixed part: Interpolation* : The following matrices are evaluated:

$$M_i = \tilde{\Phi}(\theta_i)^T S_r^{(k)} \tilde{\Psi}(\theta_i) \in \mathbb{R}^{r\times r}, i \in [\![1,N]\!]. \tag{3.16}$$

An interpolation $\theta \to \tilde{M}(\theta)$ suitable for scalar parameters similar to Eq. (3.14) is built:

$$\tilde{M}(\theta) = \sum_{i=1}^{N} L_i(\theta) M_i. \tag{3.17}$$

- *Final Predictor* : The predictor of the entire solution reads:

$$\tilde{S}(\theta) = \tilde{\Phi}(\theta)\tilde{M}(\theta)\tilde{\Psi}(\theta)^T \in \mathbb{R}^{n\times m}. \tag{3.18}$$

Algorithm 3: Interpolation on Grassman manifolds algorithm.

Require: Dataset $\{\theta_i, S^{(i)}\}_{i\in[\![1,N]\!]}$, precision threshold ε

Perform SVD on Dataset $\bar{S} = \frac{1}{N}\sum_{i=1}^{N} S^{(i)} = \bar{U}\bar{\Sigma}\bar{V}^T$, $\bar{\Sigma} = \mathrm{Diag}(\sigma_1, ..., \sigma_m)$.

Determine r the smallest integer satisfying $\dfrac{\sum_{i=1}^{r}\bar{\sigma}_i^2}{\sum_{i=1}^{m}\bar{\sigma}_i^2} > 1-\varepsilon$.

for $k \in [|1,n|]$ **do**

Perform SVD on $S^{(k)} = \Phi^{(k)}\Sigma^{(k)}\Psi^{(k)^T}$

Perform SVD on $Y_k(Y_1^T Y_k)^{-1} - Y_1 = U_k\Sigma_k V_k^T$, Y_k being respectively $\Psi_r^{(k)}$ or $\Phi_r^{(k)}$

end for

Construct the matrices $Z_1 = 0_{\mathbb{R}^{n\times r}}$, $Z_k = U_k \tan^{-1}(\Sigma_k)V_k^T, k > 1$ for respectively $\Psi_r^{(k)}$ or $\Phi_r^{(k)}$

for $k \in [|1,n|]$ **do**

Use Lagrange polynomial basis functions to obtain a mapping for $(\theta_i, Z(\theta_i))_{i\in[\![1,N]\!]}$

$\tilde{Y}(\theta) = [Y_1 V(\theta)\cos(\Sigma(\theta)) + U(\theta)\sin(\Sigma(\theta))]V(\theta)^T$ Y_k being respectively $\Psi_r^{(k)}$ or $\Phi_r^{(k)}$

$M_i = \tilde{\Phi}(\theta_i)^T S_r^{(k)} \tilde{\Psi}(\theta_i) \in \mathbb{R}^{r\times r}, i \in [\![1,N]\!]$

$\tilde{M}(\theta) = \sum_{i=1}^{N} L_i(\theta) M_i$

Prediction for θ $\tilde{S}(\theta) = \tilde{\Phi}(\theta)\tilde{M}(\theta)\tilde{\Psi}(\theta)^T$

end for

Vectorial version (contribution)

In this case, the dataset reads: $\{\theta_i, S_i\}_{i\in[\![1,N]\!]}$, where $\theta_i \in \mathbb{R}^d$ and $S_i = [y(\theta_i,t_1),...,y(\theta_i,t_m)] \in \mathbb{R}^{n\times m}$. This version is coined 'vectorial' because the parameters θ are now taken as real vectors.

The modifications proposed are performed on Equations (3.14) and (3.17): they permit to obtain mappings $\theta \to \tilde{Y}$ and $\theta \to \tilde{M}$ based on datasets $\{\theta_i, Y_i\}_{i\in[\![1,N]\!]}$ and $\{\theta_i, M_i\}_{i\in[\![1,N]\!]}$ respectively. As $M_i \in \mathbb{R}^{r\times r}$ with r relatively small, a regression based on random forests (using the Python package scikit learn) is directly used on the components of the matrix.

Instead, as $Y_i \in \mathbb{R}^{q\times r}$ (with $q = n$ or $q = m$) represents matrices of spatial or temporal eigenvectors, possibly very large, a different strategy is adopted. As $Y_i = [Y_i^{(1)},...,Y_i^{(r)}]$ with $Y_i^{(l)} \in \mathbb{R}^q, l \in [\![1,r]\!]$, parameterized POD regressors as introduced in Subsection 3.3.1 are used to obtain independent regressors for each column of Y_i, denoted as $\tilde{Y}^{(l)}(\theta)$ built using the DoE $\{\theta_i, Y_i^{(l)}\}_{i\in[\![1,N]\!]}$. The regressor on the full eigenvector matrix reads:

$$\tilde{Y}(\theta) = [\tilde{Y}^{(1)}(\theta),...,\tilde{Y}^{(r)}(\theta)]. \tag{3.19}$$

3.3.4 Deep convolutional neural regressor

A Deep convolutional Neural Regressor (*DcNR*) consists in learning to generate the physical data S over a grid with the parameters vector θ as an input. As indicated by its name, the internal structure of this network is formed by a succession of transposed convolutional layers of adequate dimensions in order to obtain a regression model of the physical field in the adequate size. The objective function in this case being:

$$\min_{\gamma} \sqrt{\mathop{\mathbb{E}}_{(t,\theta)} ||S(\theta,t) - N(\gamma,\theta,t)||_2^2}, \tag{3.20}$$

In practice, the empirical mean is used to approximate the expectation:

$$\min_{\gamma} R(\gamma,\Theta,T) = \sqrt{\sum_{(t,\theta)\in(T\times\Theta)} \frac{1}{|T||\Theta|} ||S(\theta,t) - N(\gamma,\theta,t)||_2^2}, \tag{3.21}$$

where N denotes the neural network, γ its trainable weights, Θ the training set of parameters vectors and $|\Theta|$ its cardinal, T the the set of time steps and $|T|$ its cardinal. The use of optimal Latin Hyper Cube sampling with a significantly large data set ensures that the introduced bias due to the use of the average to approximate the expectation is kept reasonably low. We trained two types of DcNR, the first type generating the whole time series of the simulation at each call and the second generating only one time step at each call, thus having the time step t as an additional parameter. Both networks have the same architecture except for the number of filters at each feature map. Both architectures start with a linear upscaling of the parameter vector then a succession of 3D transposed convolution used for physical values in Bode et al. [2019] for the first time, batch normalization Ioffe and Szegedy [2015b] and hyperbolic tangent activation. We used Adam (Kingma and Ba [2014]) as the optimization algorithm for the descent step.

Table 3.1: Network architecture - Transposed convolutions

Layer	Kernel Size	Stride	Padding	n-filters Full	n-filters time parameterized
1	(3,3,3)	(1,1,1)	(0,0,0)	1024	128
2	(3,3,3)	(1,1,1)	(0,0,0)	512	64
3	(3,3,4)	(1,1,1)	(0,0,0)	256	32
4	(5,5,5)	(1,1,1)	(0,0,0)	196	16
5	(4,4,5)	(2,2,1)	(0,0,0)	128	8

Algorithm 4: Training of DcNR. All experiments in this chapter used the default values $\lambda = 0.0001$, $m = 2000$ simulations, $\varepsilon_0 = 10^{16}$, $\varepsilon_{obj} = 10^{-3}$, Epochs=10000.

Require: λ learning rate, m batch size, ε_0 initial error of model, ε_{obj} error objective, Epochs number of maximum epochs authorized, γ DcNr weights, M number of training simulations.

$s \leftarrow \frac{M}{m}$
$i \leftarrow 1$
while $(i \leq Epochs)$ or $(\varepsilon_{obj} \leq \varepsilon_i)$ **do**
  $\varepsilon_i \leftarrow 0$
  for $j \in [|1, s|]$ **do**
    Sample $(t_k, \theta_k)_{k=1}^{m} \in (T_{batch} \times \Theta_{batch})$
    $\gamma \leftarrow \gamma - \lambda Adam(R(\gamma, \Theta_{batch}, T_{batch}))$
    $\varepsilon_i \leftarrow \varepsilon_i + R(\gamma, \Theta, T)$
  end for
  $\varepsilon_i \leftarrow \frac{\varepsilon_i}{s}$
  $i \leftarrow i + 1$
end while

For reproducibility purposes, both Table 3.1 and Algorithm 4 show the architectures and the training algorithm used in all experiments.

3.4 Numerical Examples

3.4.1 Data Sampling

In this section we present the data range used for sampling and generating data for the training and testing phase. Values were chosen as: $L_{plate} = 8m$, $W_{plate} = 4m$, $H_{plate} = 0.5m$, $H_{cylinder} = 4m$, $D_{cylinder} = 1m$, $N_T = 100$, $\Delta t = 4 \times 10^{-2} s$, and $v_z = 100m/s$.

The variable parameters identified in Section 3.2 are sampled following Table 4.1 values with a latin hypercube sampling framework. We sampled $N_{Training} = 100$ values for the training data set and $N_{Testing} = 25$ values for the testing data set.

Table 3.2: Parameters sampling

P	Mean Value	Variation (%)	Min Value	Max Value
Young modulus$E$	2.2×10^{11} MPa	20%	1.7×10^{11} MPa	2.64×10^{11} MPa
Densityν	$7800 kg/m^3$	20%	$6240 kg/m^3$	$9360 kg/m^3$
Poisson's ratio ρ	0.3	20%	0.24	0.35

We define Ω as the discretized space of the plate and Ω_{z^*} the 2D surface resulting from the intersection of the plate grid and the plan of equation $z = z^*$. As a quantity of interest, we choose to train our models to predict displacement and velocity on the z-axis for the plate, and to compare our models, we fix two time steps $t_1 = 0.01s$ and $t_2 = 0.03s$, two parameters vectors $\theta_1 = \left(2.12 \times 10^{11}, 7.87 \times 10^3, 3.13 \times 10^{-1}, 2.20 \times 10^{11}, 6.47 \times 10^3, 2.72 \times 10^{-1}\right)$ and $\theta_2 = \left(1.71 \times 10^{11}, 6.68 \times 10^3, 3.27 \times 10^{-1}, 2.31 \times 10^{11}, 6.86 \times 10^3, 3.41 \times 10^{-1}\right)$, two points $p_1 = (1.66, 0.88, 0.2875)$ and $p_2 = (5, 3.33, 0.2875)$ and $\Omega_{z=0.2875}$ as parameterized points to asses the precision of our models to predict values of interest.

We define a relative error indicator in order to quantify the precision of our metamodels, noted η. For a metamodel M, a parameter vector θ, a space point $p = (x, y, z)$, and a time value t:

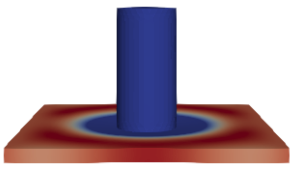
$$\eta_p(M, \theta, p, t) = \frac{|M(\theta, p, t) - S(\theta, p, t)|}{\sqrt{\mathop{\mathbb{E}}_{p \in \Omega} |S(\theta, p, t)|^2}}. \tag{3.22}$$

Then we define the global error indicator to compare models:

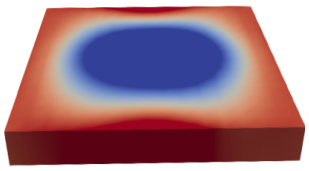
$$\eta(M, \theta, t) = \mathop{\mathbb{E}}_{p \in \Omega} [\eta_p(M, \theta, p, t)]. \tag{3.23}$$

3.4.2 Data preprocessing

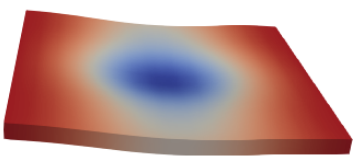
To ensure that our simulation data can be processed by convolutional neural networks, a projection phase of the data on an Euclidean 3D grid is necessary, we choose the smallest box that contains the whole range of displacement of the original grid, using the Finite Element basis functions we construct the projection operator from the initial mesh onto the containing box in addition to the inverse projection operator. Then after training our models and computation of test values, an inverse projection operation is done on the original grid. Figure 3.2 shows a visualization of this framework. Besides, since neural networks are more suited to learn from continuous fields, we used extrapolation, using an extension of the finite element basis functions, over the area were no motion of the grid is detected for each time step, see Figure 3.3. It prevents the neural network from having to learn the sharp discontinuities of the boundaries of the plate. After the projection operation is performed we scale the physical data using a standard affine scaler to standardize data by removing the mean and scaling to the unit variance.



(a) Data generation on full mesh



(b) Projection on containing box



(c) Inverse projection on initial mesh

Figure 3.2: Data processing steps

(a) Extrapolation (b) Zero-Fill

Figure 3.3: Extrapolation vs zero-fill projection

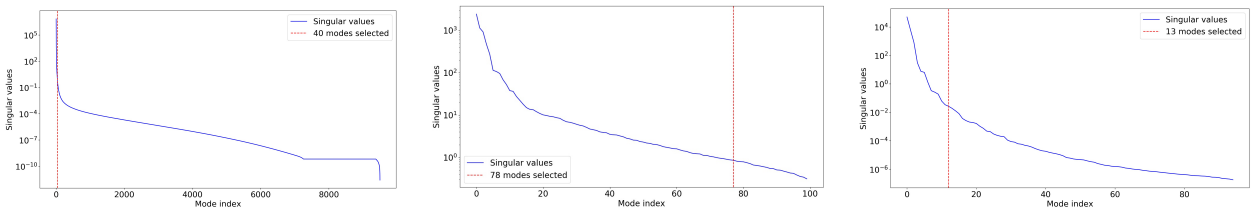
3.4.3 Trained metamodels

To provide extensive comparisons, we train multiple versions of each model described in Section 3.3:

- **POD**: We train different POD models with multiple metamodels over the orthogonal projection coefficients (random forest, Gaussian process and linear). We choose to keep POD models with random forest considering it held the best trade-off between precision and computational cost for our problem:
  - *POD_p*: it takes as input the parameter vector p and outputs the value of S over all the area of interest and for all time steps.
  - *POD_pt*: Time-Space POD, it takes as input the parameter vector p and the time value t and outputs the value of S over all the area of interest at the instant t.
- **Stief** : We trained a generalized model for interpolation on Grassman manifolds using also a random forest model for inference over parameters.
- **DcNR**: We train multiple DcNR :
  - *NN*: it takes as input the parameter vector p and outputs the value of S over all the area of interest and for all time steps.
  - *NN_t*: it takes as input the parameter vector p and the time value t and outputs the value of S over all the area of interest at the instant t.

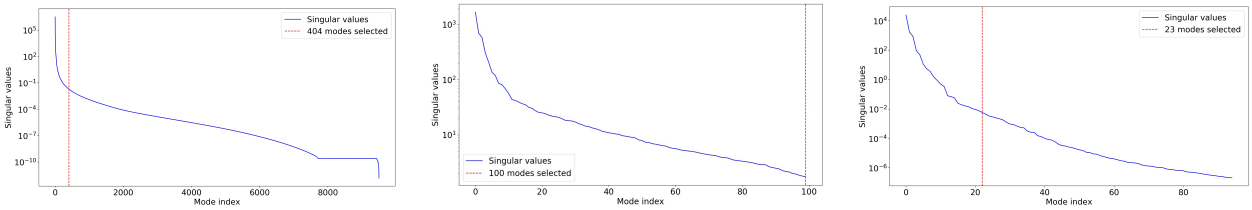
Since the error threshold for the root mean squared error was set as 10^{-3} when training the DcNR, we train our linear models (POD metamodels) with an error threshold of 10^{-6} and with the same projected and scaled data to ensure fair comparison.

3.4.4 Results



(a) POD_pt displacement (b) POD_p displacement (c) Stief displacement

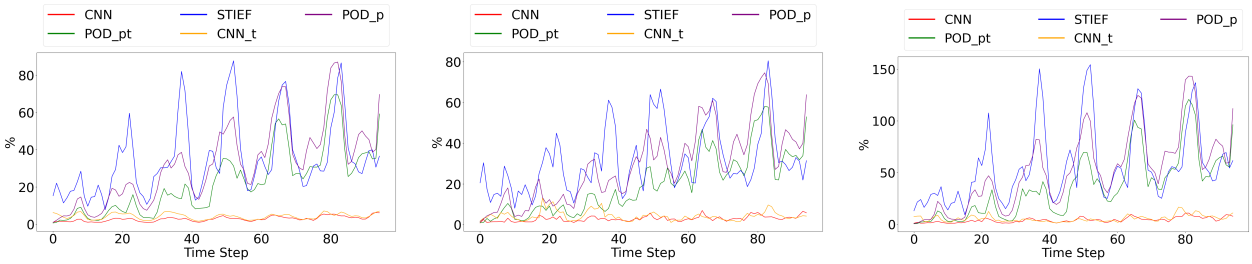
Figure 3.4: Singular values decay and modes selection for linear models - Displacement



(a) POD_pt velocity (b) POD_p velocity (c) Stief velocity

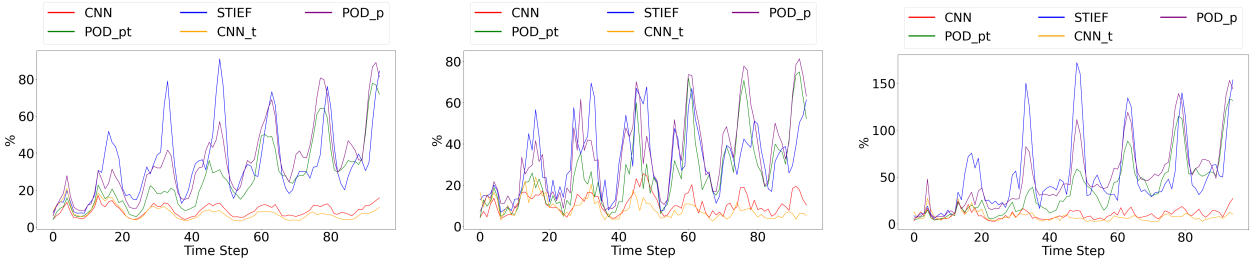
Figure 3.5: Singular values decay and modes selection for linear models - Velocity

Figures 3.4 and 3.5 show the different singular values decay for each linear model, all models require more modes to fit the velocity which can be explained by the fact that velocity data are less linearly reducible than displacement data. The POD_{pt} model selects an inferior ratio of modes than the POD_p model, considering both parametric and time-space information in the construction of the basis gives better approximation properties to the linear model. Nonetheless, the cost of singular value decomposition operation done for the construction of the POD_{pt} model is more important and, the POD_{pt} corresponding metamodel must learn coefficients in a high dimension ($m \times r$, where m is the number of the time steps and r are the retained POD modes) thus being more difficult to train.



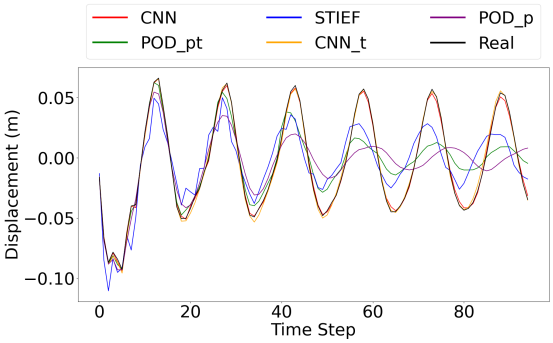
(a) Global error indicator (b) Pointwise error indicator on p_1 (c) Pointwise error indicator on p_2

Figure 3.6: Error indicator η for displacement averaged over test

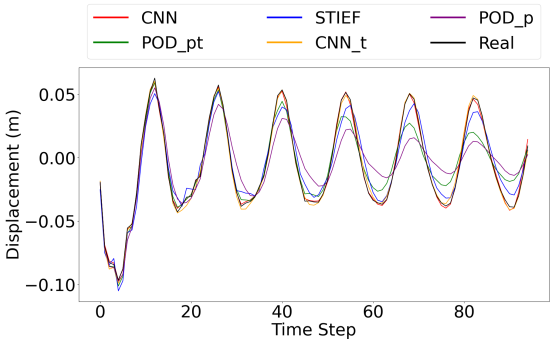


(a) Global error indicator (b) Pointwise error indicator on p_1 (c) Pointwise error indicator on p_2

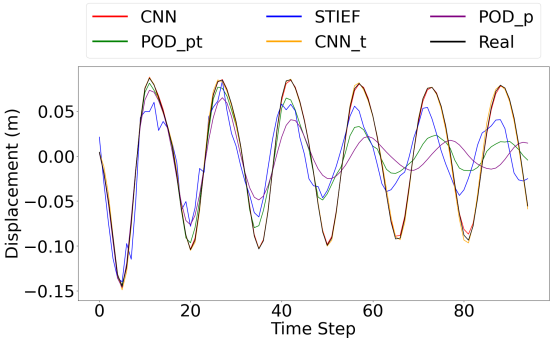
Figure 3.7: Error indicator η for velocity averaged over test



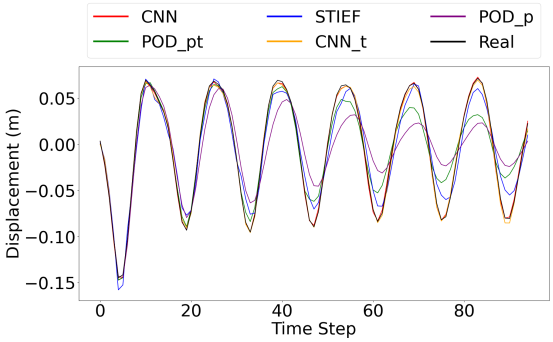
(a) Displacement - (p_1, θ_1)



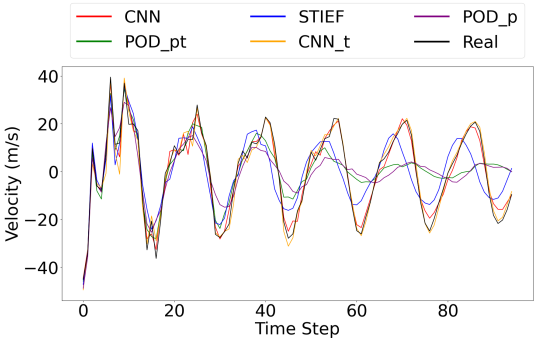
(b) Displacement - (p_1, θ_2)



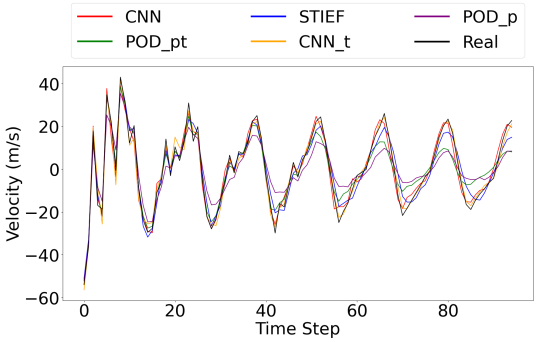
(c) Displacement - (p_2, θ_1)



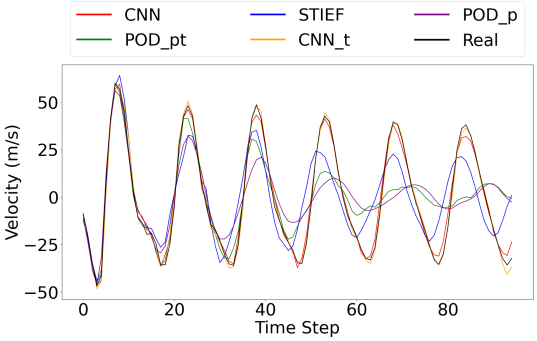
(d) Displacement - (p_2, θ_2)



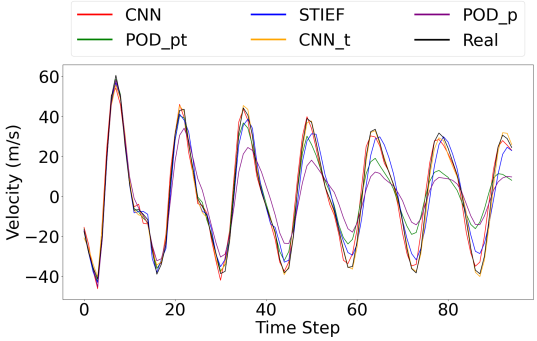
(e) Velocity - (p_1, θ_1)



(f) Velocity - (p_1, θ_2)



(g) Velocity - (p_2, θ_1)

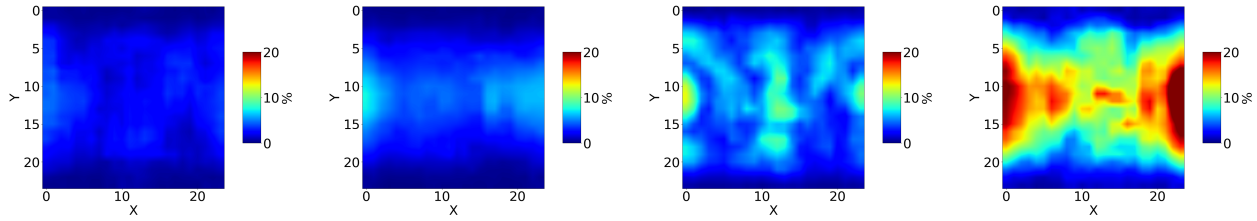


(h) Velocity - (p_2, θ_2)

Figure 3.8: Models prediction onp_1 and p_2

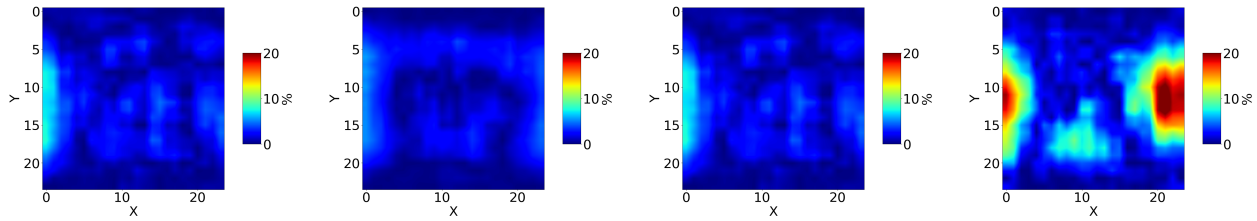
Figures 3.6 and 3.7 show that best precision is achieved by the neural networks, while within the linear models, the POD_{pt} holds the best behavior, which is also shown by the figure 3.8 where

the velocity is, as expected, harder to fit by all models. The intuition of comparing between a model whose objective is to learn the whole physical simulation *CNN* and a model whose objective is to learn a regression scheme over the time step CNN_t is to test the hypothesis that a model who has access to a full time interval of the simulation captures better the background physics. Nonetheless the CNN_t results shows slightly better results since *CNN* model is also more difficult to train as the number of trainable parameters increases with the size of the time interval considered. Within those figures, a time dependency of the accuracy can be seen over most models. This time dependency is reduced when using the neural networks, and especially in the model CNN_t.



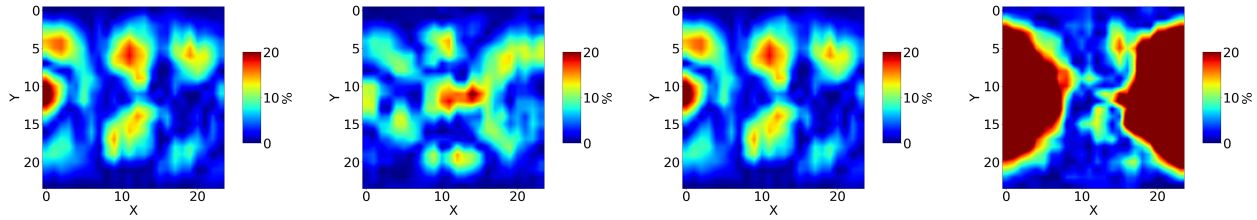
(a) Displacement t_1 (b) Displacement t_2 (c) Velocity t_1 (d) Velocity t_2

Figure 3.9: CNN error on S averaged on test



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

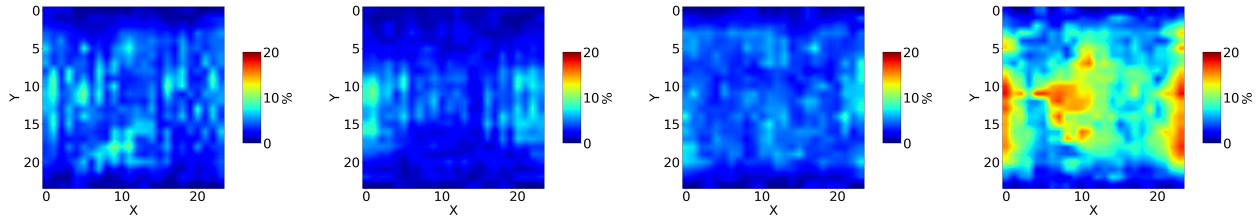
Figure 3.10: CNN error on S for displacement



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

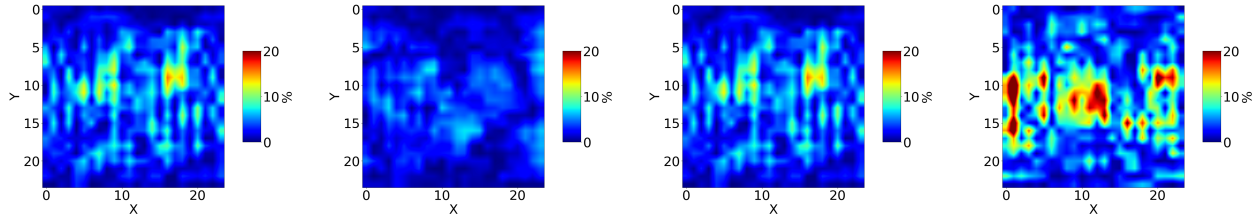
Figure 3.11: CNN error on S for velocity

Figures 3.9, 3.10 and 3.11 shows an introduced noise and error by the neural networks following the structure of the multiple convolution. A way of correcting this noise was developed in Boukraichi et al. [2021] by using a physical submodel over an area of interest that takes as an input boundary conditions learned by a neural network; this method was not used in this chapter. We can also see a time dependency of the error, but globally the model achieves good accuracy.



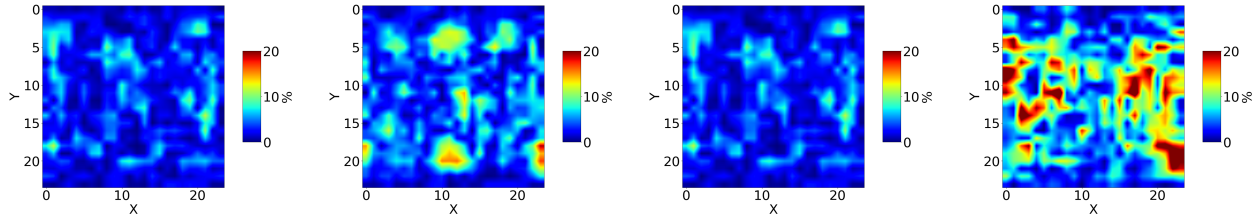
(a) Displacement t_1 (b) Displacement t_2 (c) Velocity t_1 (d) Velocity t_2

Figure 3.12: CNN_t error on S averaged on test



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

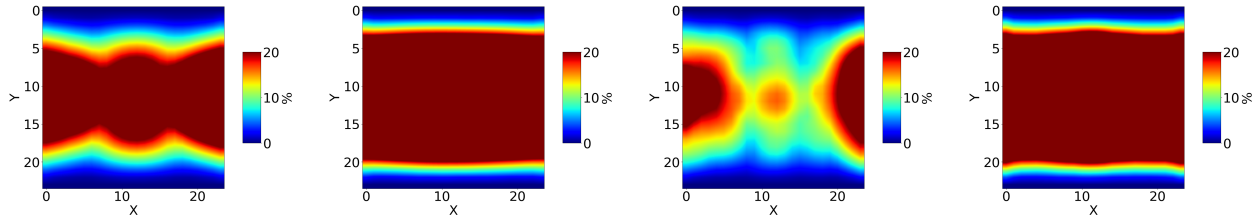
Figure 3.13: CNN_t error on S for displacement



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

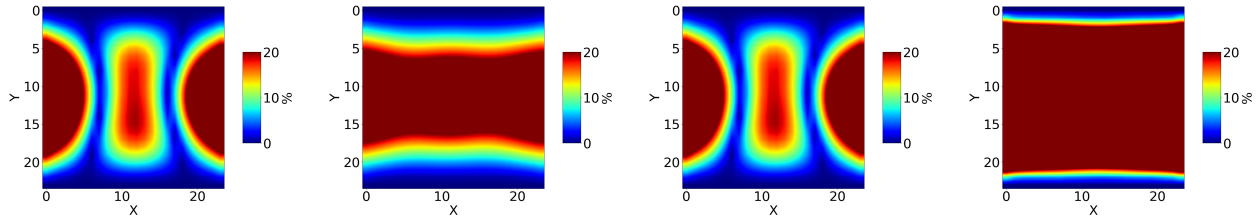
Figure 3.14: CNN_t error on S for velocity

Figures 3.12, 3.13 and 3.14 shows that the introduced noise and error by the neural network architecture is higher in the CNN_t model but the time dependency on the error was completely eliminated, globally the model achieves better accuracy results than the CNN model (1% of error at t_2 for CNN_t vs around 3% for CNN).



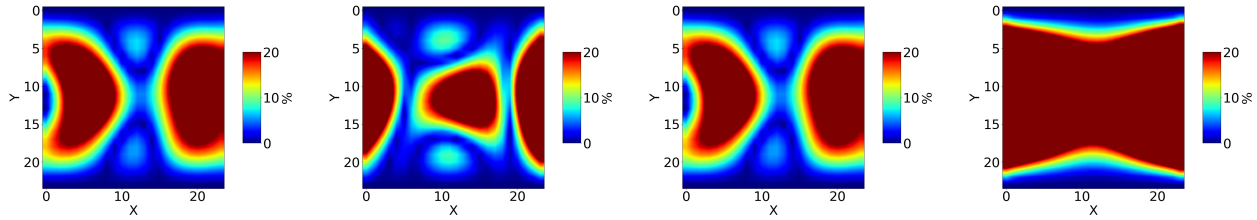
(a) Displacement t_1 (b) Displacement t_2 (c) Velocity t_1 (d) Velocity t_2

Figure 3.15: POD_p error on S averaged on test



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

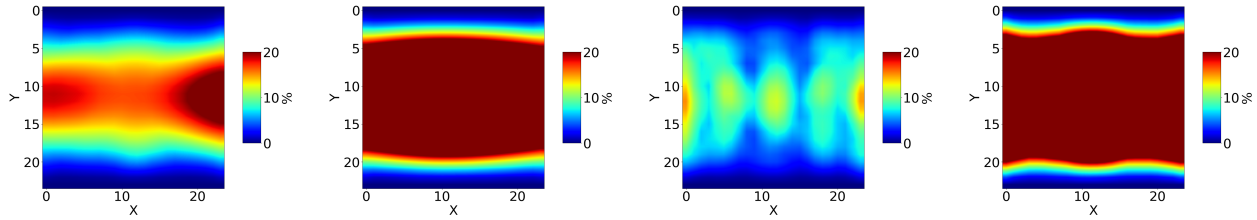
Figure 3.16: POD_p error on S for displacement



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

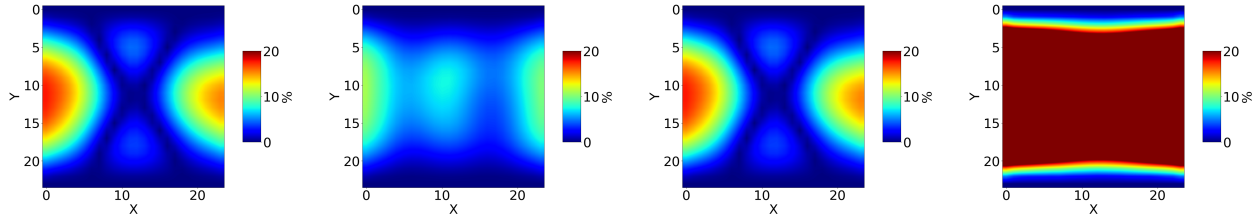
Figure 3.17: POD_p error on S for velocity

Figures 3.15, 3.16 and 3.17 shows that the error of the POD_p model has a physical and structured shape and a poor precision. Time dependency of the error is highly present for this model even though it cannot be seen over these figures but it is present on figures 3.6 and 3.7.



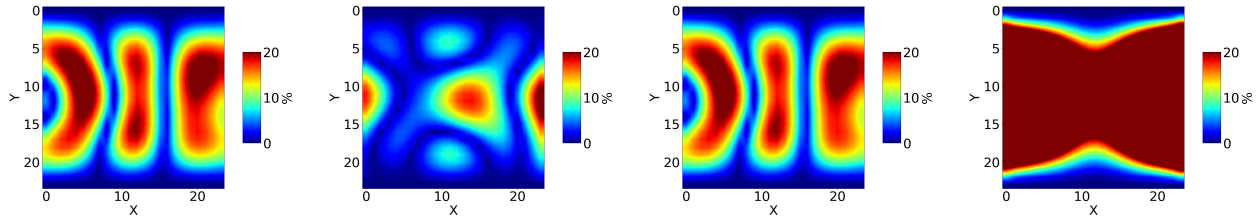
(a) Displacement t_1 (b) Displacement t_2 (c) Velocity t_1 (d) Velocity t_2

Figure 3.18: POD_pt error on S averaged on test



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

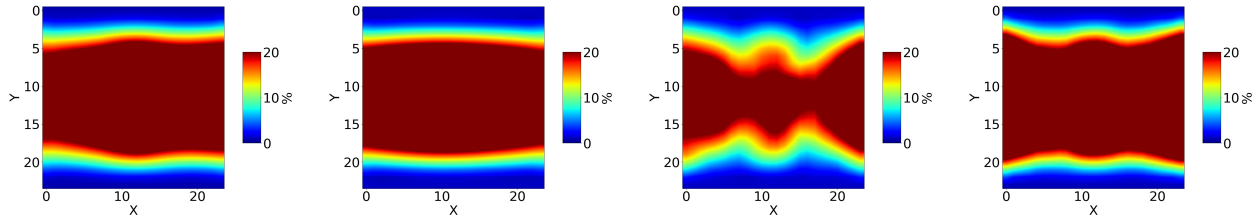
Figure 3.19: POD_pt error on S for displacement



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

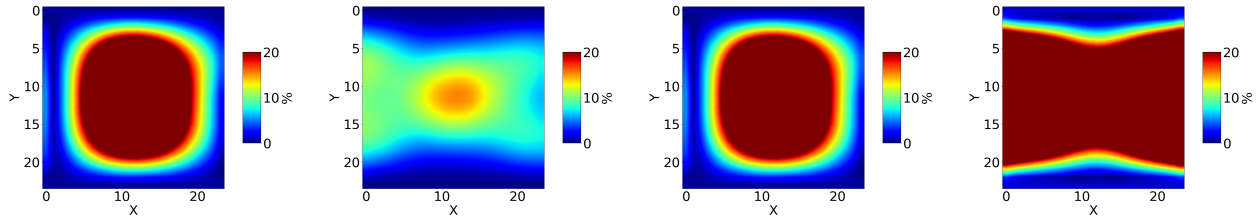
Figure 3.20: POD_pt error on S for velocity

Figures 3.18, 3.19 and 3.20 shows that the error of the POD_{pt} model has also a physical and structured shape, but a better precision over the test set than the POD_p model (4% of error at t_1 for POD_{pt} vs around 8% for POD_p). Time dependency of the error is highly present for this model.



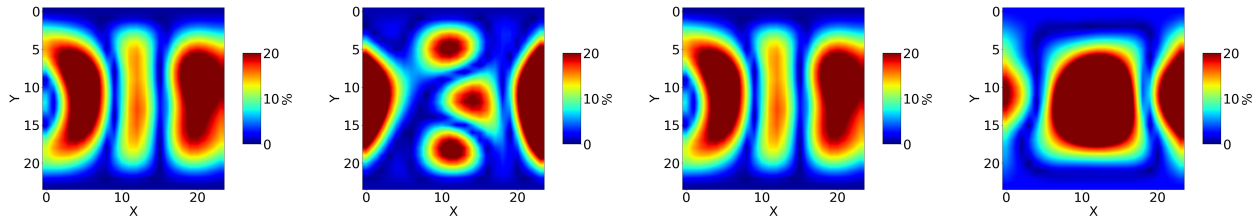
(a) Displacement t_1 (b) Displacement t_2 (c) Velocity t_1 (d) Velocity t_2

Figure 3.21: Stief error on S averaged on test



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.22: Stief error on S for displacement



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.23: Stief error on S for velocity

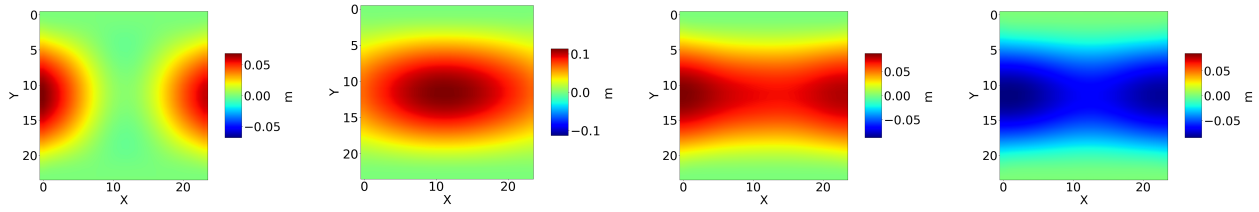
Figures 3.21, 3.22 and 3.23 shows that the error of the Stieff model is the highest within all models and has a structured shape on the error. Nonetheless our generalization approach for Grassman manifolds interpolation shows some good prediction properties for some parametric values and some region of the grid. So, we can conclude that our approach is still not optimal and requires more investigation to generalize the scalar version.

3.5 Conclusion

In this chapter we investigate the capacity of linear models relying on modal analysis to construct a regression model for a non-reducible problem, we also showed the benefits and the costs of considering both parametric and time-space information in the modal analysis step. We also propose an approach to generalize Grassman manifold interpolation from scalar output to vector ones, but it still requires further investigations. We compare all methods to deep convolutional neural regressor and we illustrate that for contact cases such as the one investigated, linear methods behave very poorly and it is recommended to use non-linear data driven methods such as DcNR. We also illustrate that having the time step as a parameter of the DcNR helps reducing the dependency of the error with respect to the time.

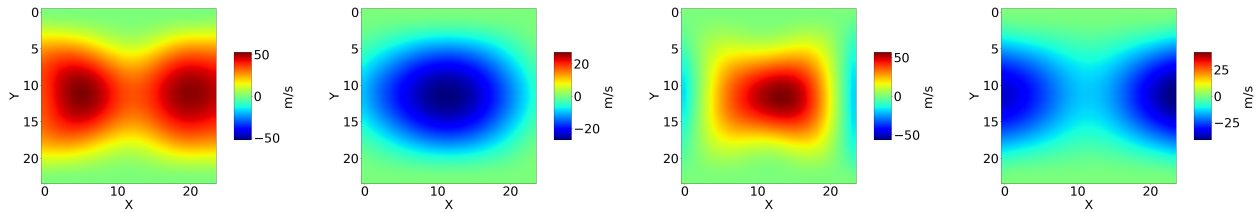
3.6 Supplementary results

In this section we present to the reader additionnal 2D visualization for further comparison between models.



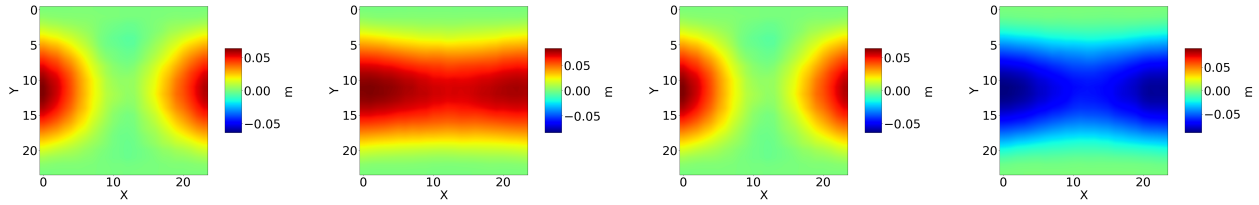
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.24: Real values of displacement



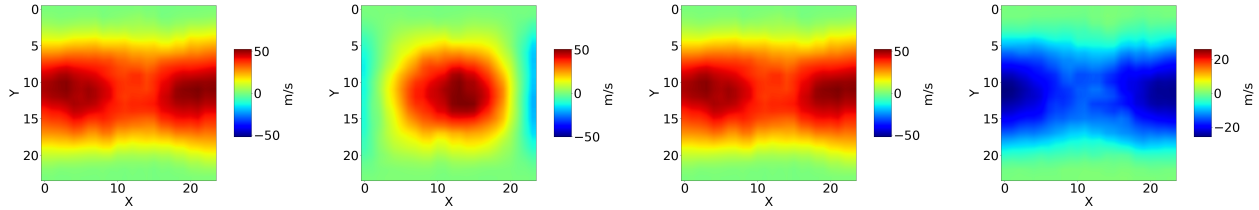
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.25: Real values of velocity



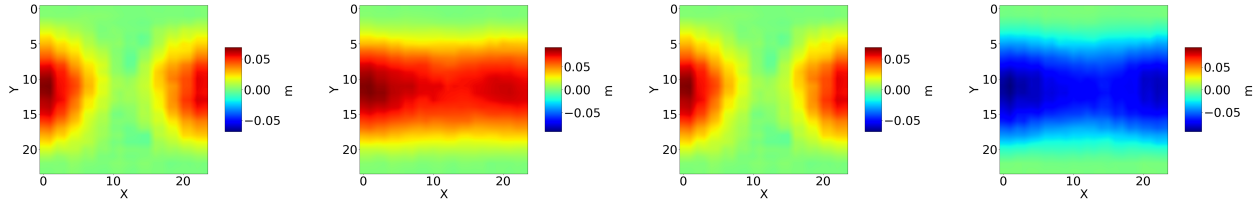
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.26: CNN prediction on S for displacement



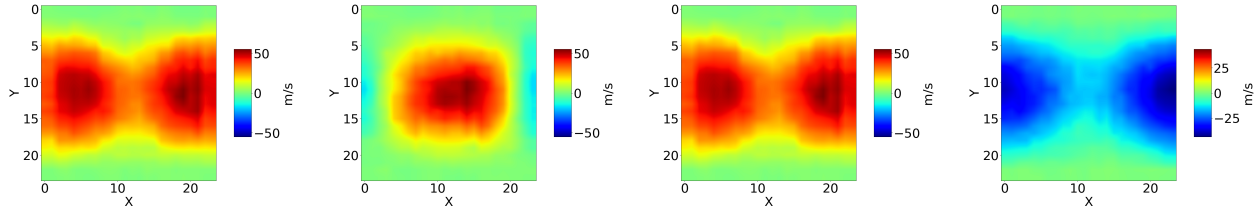
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.27: CNN prediction on S for velocity



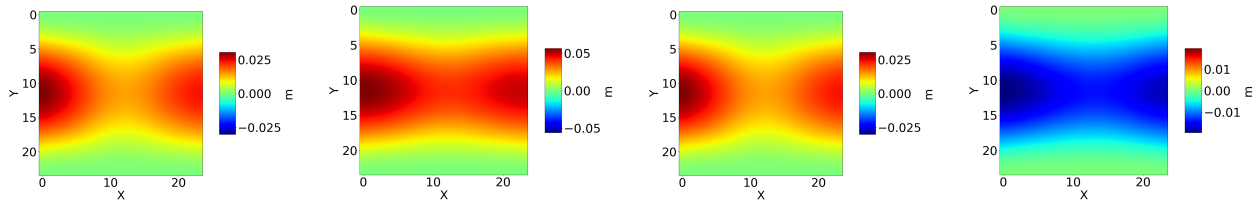
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.28: CNN_t prediction on S for displacement



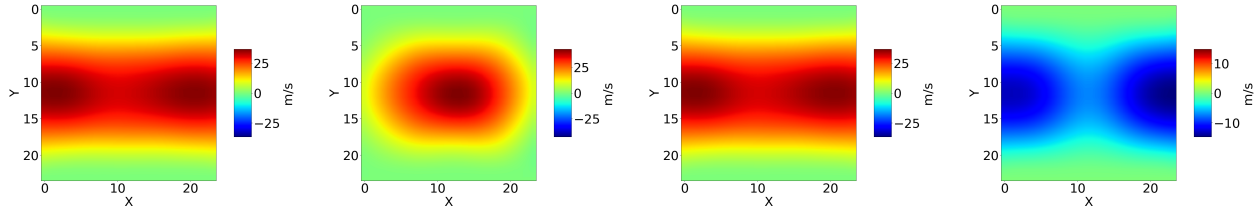
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.29: CNN_t prediction on S for velocity



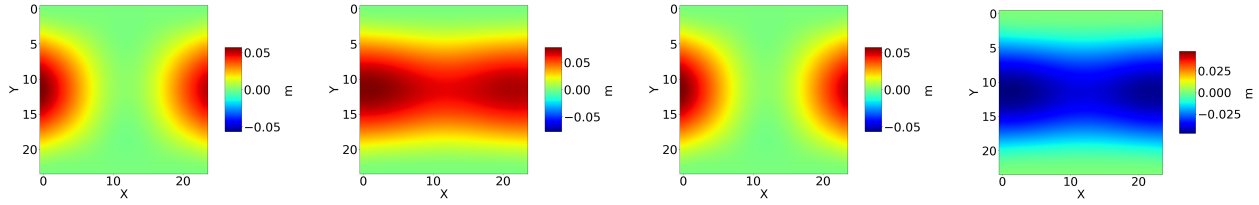
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.30: POD_p prediction on S for displacement



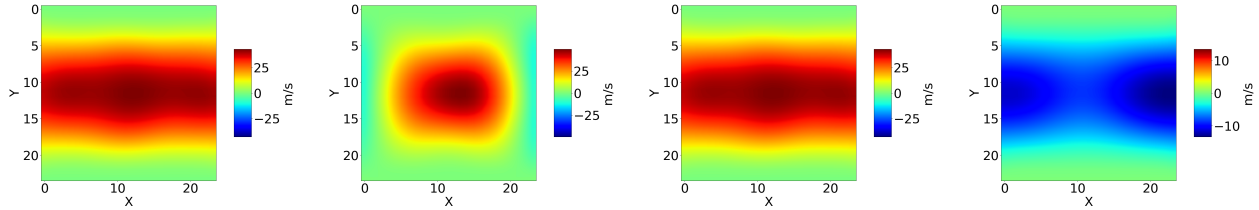
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.31: POD_p prediction on S for velocity



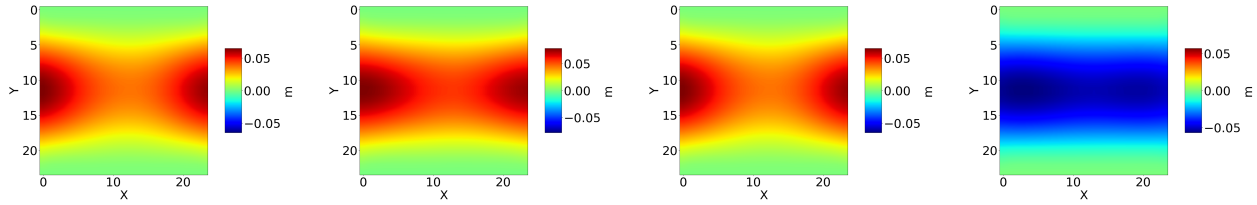
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.32: POD_pt prediction on S for displacement



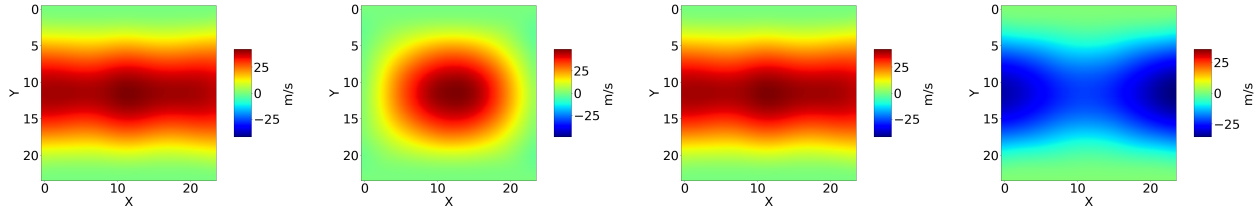
(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.33: POD_pt prediction on S for velocity



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.34: Stief prediction on S for displacement



(a) (t_1, θ_1) (b) (t_1, θ_2) (c) (t_2, θ_1) (d) (t_2, θ_2)

Figure 3.35: Stief prediction on S for velocity

Chapter 4

Uncertainty quantification in a mechanical submodel driven by a Wasserstein Generative Adversarial Network

Abstract

This chapter presents novel methods for submodeling using deep learning models for parametric and non-parametric uncertainty quantification for fast dynamics. In the case of fast dynamics and wave propagation, we investigate a random generator of boundary conditions for submodels by using machine learning. Generative Adversarial Networks (GANs) are used to extract stochastic boundary conditions for faster finite element predictions on a submodel. The framework can be viewed as a randomized and parameterized simulation generator on the submodel, which can be used as a Monte Carlo estimator. The objective of this work is to evaluate the performance of GANs for uncertainty quantification in a physical configuration and in the presence of data generated from experiments of Finite Element Method (FEM) solvers. The results show that GANs are a powerful tool for predicting physical fields with high precision and stability in the presence of parametric and non-parametric uncertainties.

Résumé

Ce chapitre présente des méthodes innovantes pour la modélisation de sous-ensembles à l'aide de modèles d'apprentissage profond pour la quantification d'incertitudes paramétriques et non paramétriques pour la dynamique rapide. Dans le cas de la propagation rapide des ondes, nous étudions un générateur aléatoire de conditions aux limites pour les sous-modèles physiques en utilisant l'apprentissage automatique. Les réseaux antagonistes génératifs (GAN) sont utilisés pour extraire des conditions aux limites stochastiques pour des prédictions d'éléments finis plus rapides sur un sous-modèle. Le système peut être considéré comme un générateur de simulation aléatoire et paramétré sur le sous-modèle, qui peut être utilisé comme estimateur de Monte-Carlo. L'objectif de ce travail est d'évaluer les performances des GAN pour la quantification d'incertitudes dans une configuration physique et en présence de données générées à partir de solutions de méthode des éléments finis (FEM). Les résultats montrent que les GAN sont un outil puissant pour prédire les champs physiques avec une grande précision et stabilité en présence d'incertitudes paramétriques et non paramétriques.

Contents

4.1 Introduction . . . 72
4.1.1 Related work . . . 73
4.1.2 Contribution . . . 73
4.2 Models . . . 74
4.2.1 Proper Orthogonal Decomposition (POD) . . . 74
4.2.2 Deep Convolutional Neural Regressor . . . 75
4.2.3 Wasserstein Generative Adversarial Network . . . 75
4.3 Use Case . . . 76
4.3.1 Domain definition . . . 76
4.3.2 Finite element models . . . 76
4.3.3 Dataset generation . . . 77
4.4 Numerical Results . . . 78
4.4.1 Data Sampling . . . 78
4.4.2 Trained submodels . . . 78
4.4.3 Parametric approach results . . . 79
4.4.4 Non-parametric approach results . . . 80
4.5 Conclusion . . . 84

4.1 Introduction

The analysis of parametric and non-parametric uncertainties of very large dynamical systems requires the construction of a stochastic model of said system. Linear approaches relying on random matrix theory Soize [2000] and principal component analysis can be used when systems undergo low-frequency vibrations. In the case of fast dynamics and wave propagation, we investigate a random generator of boundary conditions for fast submodels by using machine learning. We show that the use of non-linear techniques in machine learning and data-driven methods is highly relevant.

Physics-informed neural networks Raissi et al. [2017b] are a possible choice for a data-driven method to replace linear modal analysis. An architecture that supports a random component is necessary for the construction of the stochastic model of the physical system for non-parametric uncertainties, since the goal is to learn the underlying probabilistic distribution of uncertainty in the data. Generative Adversarial Networks (GANs) are suited for such applications, where the Wasserstein-GAN with gradient penalty variant Gulrajani et al. [2017] offers improved convergence results for our problem.

The objective of our approach is to train a GAN on data from a finite element method code (Fenics) so as to extract stochastic boundary conditions for faster finite element predictions on a submodel. The submodel and the training data have both the same geometrical support. It is a zone of interest for uncertainty quantification and relevant to engineering purposes. In the exploitation phase, the framework can be viewed as a randomized and parameterized simulation generator on the submodel, which can be used as a Monte Carlo estimator. The aim of this chapter is to present

novel methods for submodeling using deep learning models for parametric and non-parametric uncertainty quantification for fast dynamics, where approaches based on linear modal analysis are computationally inefficient and inaccurate.

4.1.1 Related work

In order to determine parametric and non-parametric approaches, one has to define the differences between aleatory and epistemic uncertainty. These laters are stated in Batou et al. [2015] and You et al. [2020]:

- Aleatory uncertainties: the uncertainties relative to some model parameters induced by the lack of knowledge related to those parameters. To process these uncertainties, parametric approaches are used as the modeling of the uncertainty of the parameters by random variables and fields in order, for instance, to construct stiffness and mass matrices with respect to those parameters.

- Epistemic uncertainties: also arise from lack of knowledge on parameters but based on subjective perception, and limited data availability, such as interval analysis, possibility theory, and fuzzy set theory. Parametric approaches are not suited for this application. Batou et al. [2015] introduces a new type of uncertainty the uncertainties induced by the modeling errors within the choice of the physical model, which can also be considered epistemic.

Epistemic uncertainties and modeling errors cannot be processed by fully parametric approaches. Non-parametric and mixed approaches (see Batou et al. [2015]) are necessary such as probabilistic approach: random matrix theory (see Adhikari and Chowdhury [2010], Guedri et al. [2012]) and possibilistic approach: Fuzzy variables and interval analysis (see You et al. [2020]). In this chapter, both parametric and non-parametric approaches are investigated in the situation of both aleatory and epistemic uncertainties. Modeling errors are not investigated.
The use of neural networks to learn solutions of partial differential equations (PDEs) have been recently proposed for physics application (see Raissi et al. [2017b] , Raissi et al. [2019]), using the real or an approximate of the residual from the PDE to enforce a physical constraint on the output of the network. Such application exists for architectures like generative adversarial networks that were introduced in Goodfellow et al. [2014] and optimized in Gulrajani et al. [2017]. More details on these architectures can be found in Section 4.2.
Generative adversarial networks and adversarial training are used for non-parametric density estimation in general cases of random data (Abbasnejad et al. [2019b] and Singh et al. [2018]) and also for physical data that are solutions of certain partial differential equations (Yang and Perdikaris [2019]). Our study consists in using similar approaches to learn non-parametric densities over data from a finite element model, without any information about the underlying partial differential equation solved. The models training is data-driven, no physical property is used in the optimization step, but we enforce physical properties using a submodel in the area of interest in the exploitation phase for uncertainty quantification.

4.1.2 Contribution

The aim of this chapter is to present two novel methods developed for the construction of stochastic submodels for uncertainty quantification using data from a finite element model (FEM). These two methods rely on the same general principle which is a stochastic submodel formed of two components:

- A neural network learning boundary conditions around a predetermined zone of interest.
- A finite element submodel in the zone of interest using boundary conditions generated by the neural network. We assume that there is no modeling error in the zone of interest covered by the proposed submodel.

The objective is to obtain comparable or/and better predictions than a classical learning process of a neural network over physical data, while improving some physical properties. Indeed, during the training of a physics-informed neural network, increasing precision over physical properties is generally obtained using a penalization term given by the residual of the PDE in the cost function, but with FEM models designed for engineering applications, it is quite intrusive to get access to the residual of the PDE.
The development of deep neural networks that are thermodynamically-consistent is a key issue, as explained in Hernandez et al. [2021]. In our approach, the known physical properties and principles are enforced, online, using a submodel over the interest zone, here enforcing the output of the whole reduced model to be a solution to the underlying FEM formulation on the submodel zoom area. Also, the training of the neural networks is facilitated since every network learning problem is one dimension lower. For a 3D problem on a Cartesian mesh, the network has to learn the prediction over a 2D surface representing the boundary conditions instead of learning the data over the whole 3D domain.
So to address both aleatory and epistemic uncertainties, we propose two methods as follows:

- Aleatory uncertainties: a deep convolutional neural regressor is trained to generate parameterized boundary conditions associated with the parameters of the simulation, for a parametric approach.
- Epistemic uncertainties: a Wasserstein GAN is trained to generate stochastic boundary conditions by using the same training data. It aims at learning the underlying probabilistic density binding the simulation data and the parameters of the simulation, for a non-parametric approach.

Both methods are then compared to a linear data reduction, using the Proper Orthogonal Decomposition (POD) method, constructed over the same boundary data.

4.2 Models

4.2.1 Proper Orthogonal Decomposition (POD)

Let us denote by $X = [L^2(\Omega)]$ the functional Hilbert space of the square integrable scalar functions over a bounded $2D-$open set Ω. We denote the $L^2(\Omega)$-inner product by $(.,.)$. Consider $U(p)(t,x,y) \in \mathbb{R}$ the value of physical data over a mesh of Ω and associated to the parameters vector p and to a time t. The mesh has a grid shape, so that U is also a tensor of data. A subspace of the solution space is obtained thanks to the snapshots POD method Sirovich [1987]: if we discretize the time interval to m points, and the parameters domain into n points, then the snapshots set is given as follows: $\mathscr{S} = \{U_{p_j}(t_i)\ ; i = 1,...,m \ \ and \ \ \text{j=1,...,n.}\}$.
We denote by $\mathscr{U}_{\mathscr{I}}$ an element of the previous snapshots set with the subscript $\mathscr{I}$ evolving from 1 to $m \times n$. The POD modes $\Phi_{\mathscr{J}}$, $\mathscr{J} = 1,...,m \times n$, computed via the snapshots POD starts with the solution of the eigenvalues problem with the correlations matrix:

$$C_{\mathscr{I}\mathscr{J}} = (\mathscr{U}_{\mathscr{I}}, \mathscr{U}_{\mathscr{J}}), \tag{4.1}$$

of size $(m \times n) \times (m \times n)$. Let us denote by $(A_{\mathscr{J}})_{1 \leq \mathscr{J} \leq m \times n} = \left(A_{\mathscr{I},\mathscr{J}}\right)_{1 \leq \mathscr{I},\mathscr{J} \leq m \times n}$ and $\left(\lambda_{\mathscr{J}}\right)_{\mathscr{J}=1,\ldots,m \times n}$, sets of respectively orthonormal eigenvectors and eigenvalues of the matrix C. Then, the POD modes associated with $\lambda_{\mathscr{J}}$, are given by:

$$\Phi_{\mathscr{J}}(x,y) = \frac{1}{\sqrt{\lambda_{\mathscr{J}}}} \sum_{\mathscr{I}=1}^{m \times n} A_{\mathscr{I},\mathscr{J}} \mathscr{U}_{\mathscr{I}}(x,y) \quad , \forall x \in \Omega. \tag{4.2}$$

Snapshots are approximated by orthogonal projection on the space generated by a truncation of the POD basis: $U(p)(t,x,y) \approx \Sigma_{k=1}^{\hat{m}} \alpha_k(p,t)\Phi_k(x,y)$, where $\hat{m} \leq m \times n$, and α_k are called the generalized coordinates. Meta-models are then trained to predict the generalized coordinates of a new solution from the parameter values.

4.2.2 Deep Convolutional Neural Regressor

A Deep convolutional Neural Regressor (*DcNR*) consists in learning to generate the physical data (U) over a grid with the parameters vector (p) as an input. As indicated by its name, the internal structure of this network is formed by a succession of transposed convolutional layers of adequate dimensions in order to obtain a regression model of the physical field in the adequate size. The objective function in this case being the mean squared error:

$$\min_{\theta} \frac{1}{|T| \times |\mathbb{P}_{Train}|} \sum_{p \in \mathbb{P}_{Train}} \sum_{t \in T} ||U(p)(t) - N(p)(\theta,t)||_2^2 \tag{4.3}$$

Where N denotes the neural network, θ its trainable weights, $\mathbb{P}_{Train}$ the training set of parameters vectors, T the set of time steps and $|.|$ the cardinal of each space.

4.2.3 Wasserstein Generative Adversarial Network

Generative adversarial networks were introduced in Goodfellow et al. [2014] as an unsupervised framework to learn probabilistic densities over data. It showed an empirical success as an efficient method for learning and sampling from a complicated multi-modal distribution. It relies on the adversarial training of two neural networks:

- Discriminator: a neural network whose role is to compute the Wasserstein distance between the real data distribution and the data generated by the second network distribution. Its architecture is a succession of convolutional layers to determine distance values in $\mathbb{R}$. The Wasserstein distance is defined as follows :

$$\mathbb{W}(\mathbf{P}_r, \mathbf{P}_g) = \inf_{\eta \in \Pi(\mathbf{P}_r, \mathbf{P}_g)} \mathop{\mathbb{E}}_{(x,y) \sim \eta} [||x - y||_2] \tag{4.4}$$

  Where $\mathbf{P}_r$ and $\mathbf{P}_g$ denote the real data distribution and the generated data distribution. In Gulrajani et al. [2017], it has been shown that using Kantorovitch-Rubinstein duality, for a function f with value in $\mathbf{R}$, we obtain:

$$\mathbb{W}(\mathbf{P}_r, \mathbf{P}_g) = \sup_{||f||_L \leq 1} \mathop{\mathbb{E}}_{x \sim \mathbf{P}_r} [f(x)] - \mathop{\mathbb{E}}_{x \sim \mathbf{P}_g} [f(x)] \tag{4.5}$$

Where :

$$||f||_L = \sup_{x \neq y} \frac{|f(x) - f(y)|}{||x - y||_2} \tag{4.6}$$

- Generator: a generative model, whose role is to generate new data resembling the real data from a random vector (the input) in order to fool the discriminator. Its architecture is a succession of transposed convolutional layers.

Then using the discriminator as the 1-Lipschitz function to compute the Wasserstein distance in an objective function defined as follows:

$$\begin{aligned} &\min_{\theta_{gen}} \max_{\theta_{disc}} \mathop{\mathbb{E}}_{z \sim \mathcal{N}(0,1)} [D(\theta_{disc}, G(\theta_{gen}, z))] \\ &- \mathop{\mathbb{E}}_{p \in \mathbb{P}_{Train}} [D(\theta_{disc}, U(p))] \end{aligned} \tag{4.7}$$

will lead the generator to convergence and being able to sample from the real data distribution using the random vector as a latent space descriptor. The 1-Lipschitz property of the discriminator has to be preserved through training. In 6.9, G and D denote respectively the generator and discriminator networks, θ_{gen}, θ_{disc} their respective trainable weights, and $\mathbb{E}$ the mathematical expectation. In practice the empirical mean is used to approximate the expectation. In the exploitation phase, the discriminator is no longer used and the generator can be viewed as a randomized simulation generator on the submodel, which can be used as a Monte Carlo estimator.

4.3 Use Case

4.3.1 Domain definition

We define two 2D Cartesian space grids Ω_h and Ω'_h, with $\Omega'_h \subset \Omega_h$ representing the zone of interest. Ω_h and Ω'_h are triangular space discretization of sizes $[N_x, N_y]$ and $[N'_x, N'_y]$ of the domains $[-L_x, L_x] \times [-L_y, L_y]$ and $[-L'_x, L'_x] \times [-L'_y, L'_y]$. And finally a temporal grid T is defined as discretization of size N_T of the space $[0, T_{final}]$ and the time step $\Delta t = \frac{T_{final}}{N_T - 1}$.

4.3.2 Finite element models

The objective here is to train a generator on data from a FEM code (Fenics, see Alnæs et al. [2015]) so as to extract boundary values for a submodel that occupies the zone of interest Ω'. For visual representation of this approach, refer to Figure 5.3. Let g be the boundary values, g is defined as Dirichlet boundary conditions for both models as:

- For the initial FEM model, zero Dirichlet boundary conditions ($g = 0$) are enforced on $\partial\Omega$.
- For the FEM submodel, g is the output of the pretrained neural network (the generator).

Using the transformation $\tilde{u} = u - g$ we can use the same formulation for both models. We choose to solve the 2D wave equation, given as follows:

$$\begin{cases} \frac{1}{c^2} \frac{\partial^2 u}{\partial t^2} - \Delta u = f \ \ on \ \ \Omega \ \ \forall t > 0 \\ u = 0 \ \ on \ \ \partial\Omega \ \ \forall t > 0 \\ u = u_0 \ \ on \ \ \Omega \ \ for \ \ t = 0 \end{cases} \tag{4.8}$$

where u is the amplitude of the wave. The variational problem can be written as:

$$\forall v \in V_h,\ a(u^n, v) = L^n(v) \tag{4.9}$$

Where V_h is the is the Lagrange P1 finite-element space defined on Ω_h.

The time discretization used for the FEM formulation is the second-order central difference scheme:

$$\frac{\partial^2 u}{\partial t^2} = \frac{u^n - 2u^{n-1} + u^{n-2}}{\Delta t^2} \tag{4.10}$$

Then we obtain for the FEM formulation (4.9):

$$a(u^n, v) = \int_\Omega u^n v dx + \Delta t^2 c^2 \int_\Omega \nabla u^n \nabla v dx \tag{4.11}$$

$$L^n(v) = \int_\Omega (\Delta t^2 c^2 f^n v + 2u^{n-1} v - u^{n-2} v) dx \tag{4.12}$$

4.3.3 Dataset generation

A source point is determined for the problem resolution where (x_S, y_S) are the source point coordinates, it is chosen to be outside the zoom domain: i.e. $(x_S, y_S) \in \Omega \backslash \Omega'$.
The source term at the right hand side of the wave equation is set as:

$$(\forall t \in T),\ \ f(x,y,t) = sin(\omega t)\delta_{(x_S,y_S)}(x,y) \tag{4.13}$$

where $\delta_{(x_S,y_S)}$ denotes the 2D Dirac distribution centered at (x_S, y_S). A three-dimensional parameter vector $p = (\omega, x_s, y_s)$ is chosen and determined then sampled, (note that c is constant over all samples since it is a parameter needed for the submodel). Sampling is done using latin hypercube sampling routines. For every parameter vector p a simulation vector $U(p)$ is generated using FEM model described in section 4.3. One sample of data is then$(p, U(p))$ where $p \in D_p \subset \mathbb{R}^3$ and $U(p)(t) \in V_h \subset \mathbb{R}^{N_X \times N_Y}$.
Then, 4 datasets are generated as the following:

- Training data set: 100 samples generated, used for training parametric approach models.
- Training data set 2 : 20 samples generated, used for training non-parametric approach models.
- Test data set: 10 samples generated, used for testing the training process of each neural network, and comparing the models we used in our study.
- Monte Carlo samples: 1000 samples generated, used for uncertainty quantification and comparison of the estimate of the real probability density with the density from the neural networks.

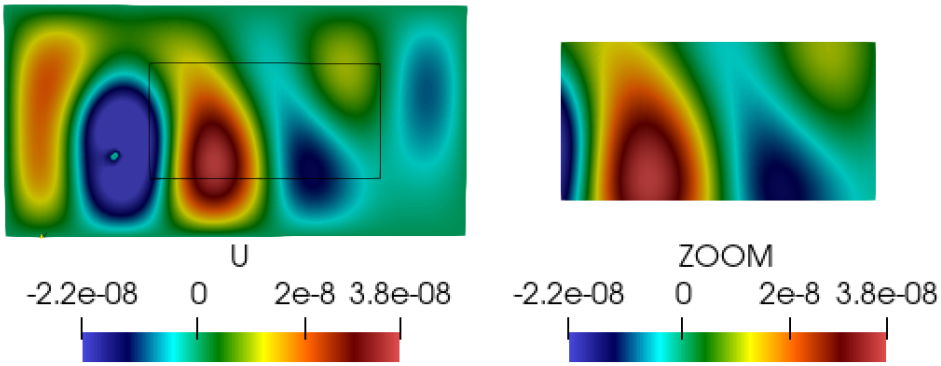


Figure 4.1: Visualization of the FEM output on Ω and $\Omega^{'}$

4.4 Numerical Results

4.4.1 Data Sampling

In this section we present the data range used for sampling and generating data for the training and testing phase. Values were chosen as: $L_x = 8m$, $L_y = 4m$, $L'_x = 4m$, $L'_y = 2m$, $N_x = 40$, $N_y = 20$, $N_T = 100$, $\Delta t = 4 \times 10^{-5}s$, $c = 2000m/s$. We recall that boundary conditions for the model over Ω are set to be zero Dirichlet boundary conditions.

The variable parameters identified in Section 4.3 are sampled following Table 4.1 values.

Table 4.1: Parameters sampling

P	Mean Value	Variation (%)	Min Value	Max Value
ω	5 kHz	5%	4.75 kHz	5.25 kHz
x_S	-1.85 m	17.5% of L_x	-2.2 m	-1.5 m
y_S	-0.65 m	28.75% of L_y	-1.8 m	0.5 m

4.4.2 Trained submodels

For every model described in Section 4.2, we train multiple version in order to do a full comparison for the two approaches:

- **POD**: We train different POD models with multiple metamodels over the orthogonal projection coefficients (random forest, Gaussian process, linear ...). We choose to keep a POD model with random forest considering it held the best trade-off between precision and computational cost for our problem. It will be referred to as *POD_RF*.
- **DcNR**: We train multiple DcNR :
  - *NN*: it takes as input the parameter vector p and outputs the value of U over all the area of interest.
  - *NN_BC*: it takes as input the parameter vector p and outputs the boundary values around the area of interest.

  - *NN_t*: it takes as input the parameter vector p and the time value t and outputs the value of U over all the area of interest at the instant t.
  - *NN_BC_t*: it takes as input the parameter vector p and the time value t and outputs the boundary values around the area of interest at the instant t.

- **GAN**: Like for the DcNR, we trained two versions, both taking as an input a random vector z and outputs the value of U over all the area of interest (*WGAN*) or the boundary values around the area of interest (*WGAN_BC*).

Predictions of every model described before restricted to the boundary of Ω' are also applied as Dirichlet boundary conditions to the submodel, they would be denoted with a suffix "_ZOOM". For information about training time of each neural network, refer to Table 4.2.

Table 4.2: Training time

Nets	Training time	GPU card
NN	12.4 Hours	NVIDIA V100
NN_BC	2.7 Hours	NVIDIA V100
WGAN	24 Hours	NVIDIA A100
WGAN_BC	7.8 Hours	NVIDIA A100

We define a relative error indicator over the time and space grid of the interest zone, in order to quantify the precision of our submodels as ε. For a submodel M, a parameter vector p (resp. random vector z for a GAN), and a time value t:

$$\varepsilon(M,p,t) = \frac{\mathop{\mathbb{E}}_{(x,y)\in\Omega'}[|M(p)(t,x,y) - U(p)(t,x,y)|]}{\max\limits_{x,y\in\Omega'}|U(p)(t,x,y)|} \tag{4.14}$$

For a comparison over the testing data set:

$$\varepsilon(M,t) = \mathop{\mathbb{E}}_{p\in\mathbb{P}_{Test}}[\varepsilon(M,p,t)] \; or \; \mathop{\mathbb{E}}_{z\sim\mathcal{N}(0,1)}[\varepsilon(M,z,t)] \tag{4.15}$$

For a comparison of the prediction of physical quantities we choose to compute the kinetic energy over the zone of interest grid using a finite difference scheme as follows:

$$K_e(p,t,x,y) = \frac{m}{2}(V(p)(t,x,y))^2 \tag{4.16}$$

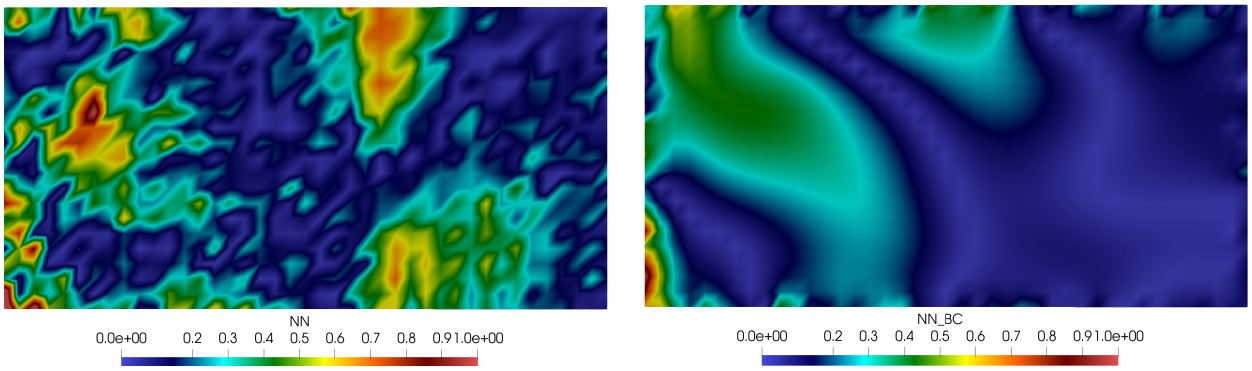
Where:

$$V(p)(t,x,y) = \frac{U(p)(t,x,y) - U(p)(t-dt,x,y)}{dt} \tag{4.17}$$

Since the mass (m) is constant over the space grid and over all parameter vectors, it will be omitted when computing the relative error over the kinetic energy prediction.

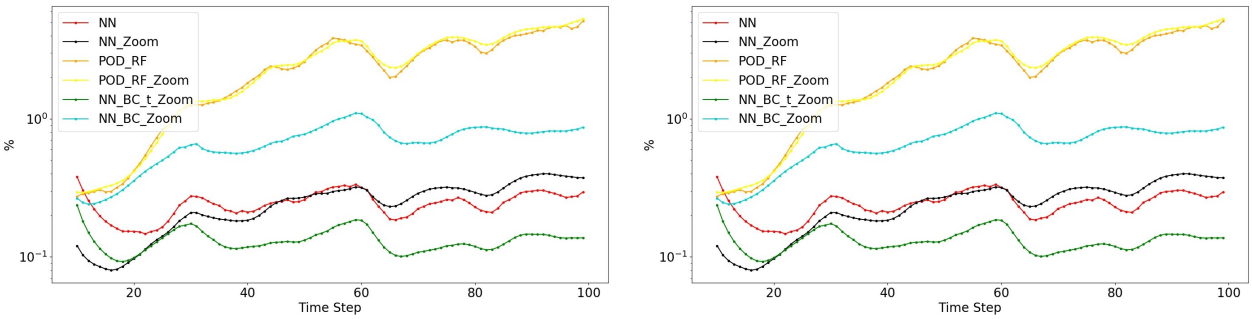
4.4.3 Parametric approach results

For the parametric approach, comparison is done by computing the error indicator defined in the previous section for all our parametric submodels, for all the samples in the testing data set described in Section 4.3 in the area of interest.



(a) NN (b) NN_ZOOM

Figure 4.2: Relative error ε



(a) Displacement (b) K_e

Figure 4.3: Pointwise relative error

Figure 4.2 shows one of the known problems when using convolutional layers to predict physical fields, which is errors and noise introduced in the output following the structure of the different convolutions, this phenomenon is corrected by the zoom operation by the submodel. As shown, the noise is still visible on the boundaries but not propagated inside the interest area. Figure 4.3 shows that the submodels approaches are better in predicting physical values such as kinetic energy, this can be explained by the fact that the submodels consists in running a partial physical model and thus having better physical properties, and as expected the POD performs poorly against non-linear methods.

4.4.4 Non-parametric approach results

For the non-parametric approach, since comparison on regression models is impossible, we used our submodels as Monte-Carlo estimators of statistical quantities and compared the estimated values with the same Monte-Carlo approach on the real data. We choose to estimate the mean and compute the error indicator defined in the previous section. And to evaluate the generative capacity of our models, we define a discrepancy indicator as follows:

$$\sigma(M,t,x,y) = \sqrt{\mathop{\mathbb{E}}_{z\sim\mathcal{N}(0,1)}[(M(z)(t,x,y) - \mathbb{E}_{Train})^2]} \tag{4.18}$$

Where M is a WGAN-based network and $\mathbb{E}_{Train}$ is the pointwise mean over the training data:

$$\mathbb{E}_{Train} = \mathop{\mathbb{E}}_{p \in \mathscr{P}_{Train}} [U(p)(t,x,y)] \tag{4.19}$$

σ computes a point wise discrepancy value to show our models capacity to generate different data from the training data. To evaluate this generative capacity we define a relative discrepancy indicator as follows:

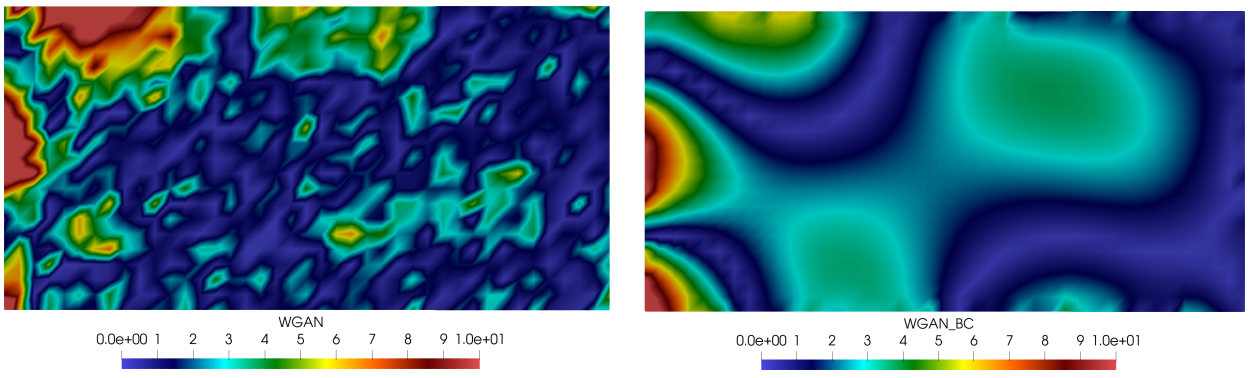
$$\sigma_{rel}(M,t) = \frac{\mathop{\mathbb{E}}_{(x,y)\in\Omega'} [|\sigma(M,t,x,y) - \sigma_{Train}|]}{\max\limits_{x,y\in\Omega'} \sigma_{Train}} \tag{4.20}$$

where σ_{Train} is the pointwise standard deviation over the training data:

$$\sigma_{Train} = \sqrt{\mathop{\mathbb{E}}_{p \in \mathscr{P}_{Train}} [(U(p)(t,x,y) - \mathbb{E}_{Train})^2]} \tag{4.21}$$

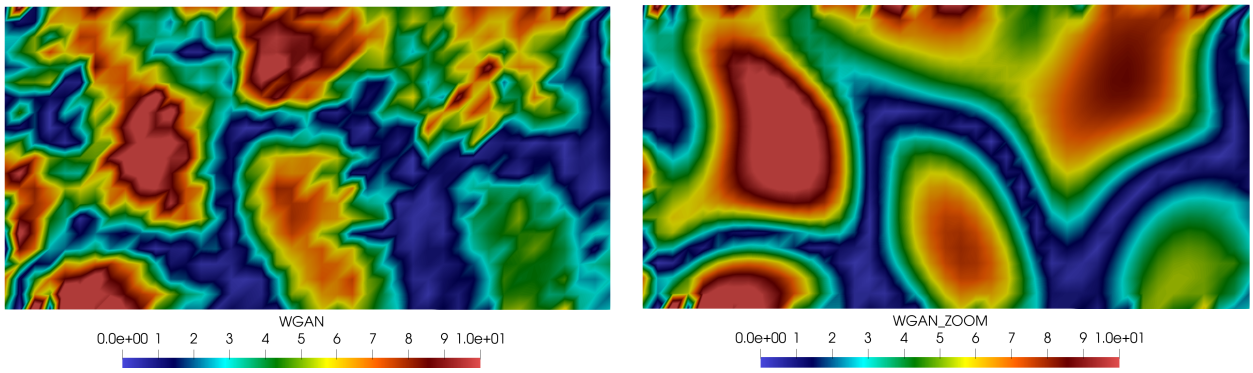
We choose as physical value the maximum amplitude defined as follows:

$$A(p)(x,y) = \left| \max_t U(p)(t,x,y) - \min_t U(p)(t,x,y) \right| \tag{4.22}$$

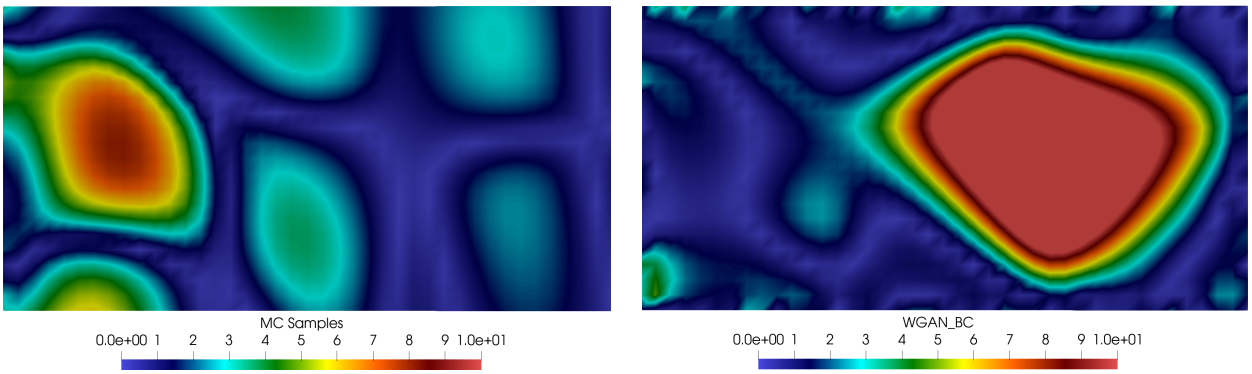


(a) WGAN (b) WGAN_ZOOM

Figure 4.4: Pointwise relative error on mean

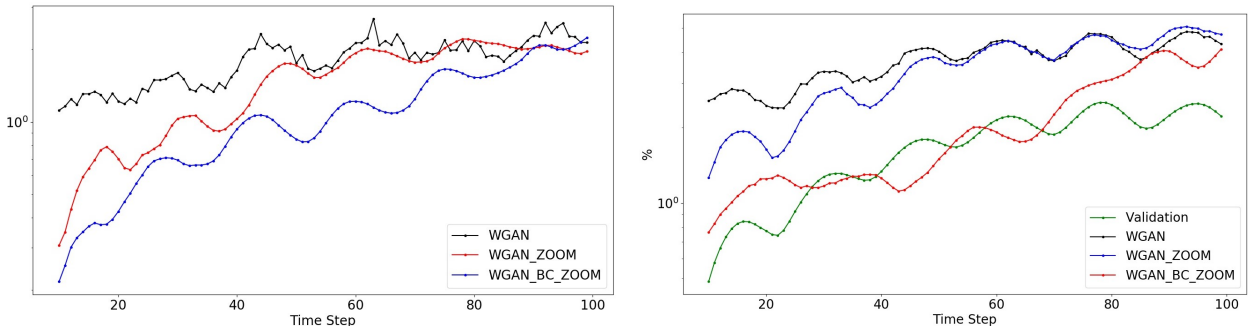


(a) WGAN (b) WGAN_ZOOM



(c) Monte Carlo samples (d) GAN_BC_ZOOM

Figure 4.5: Point wise relative discrepancy indicator



(a) Mean (b) σ_{rel}

Figure 4.6: Error indicator on WGANs predictions

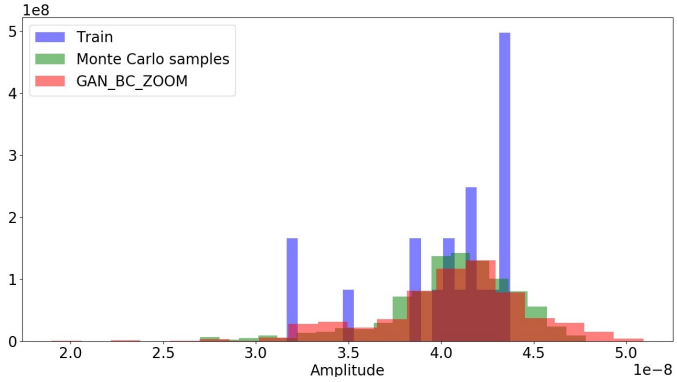
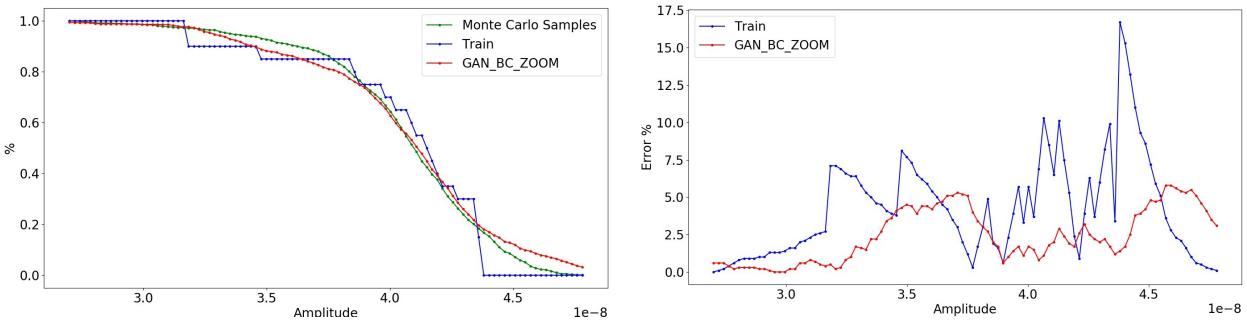


Figure 4.7: Histogram of maximum amplitude prediction



(a) Probability

(b) Relative error

Figure 4.8: Threshold crossing probability predictions

Figure 4.4 shows that the zoom operation of the submodel helds the same correction properties over the noise and errors introduced by the use of convolutional layers and also noise introduced by the GAN's random component. Figure 4.6 shows that our submodel approaches perform better on the mean prediction. Figures 4.5 shows that our approach hold better generative properties that are useful for uncertainty quantification, the *WGAN_BC_ZOOM* shows the best performance, since higher discrepancy values in the other approaches can be explained by the accumulation of the error on different non structured areas, furthermore the pointwise discrepancy in the *WGAN_BC_ZOOM* approach shows a more structured shape holding more statistically representative physical information. Figure 4.6 shows also the discrepancy indicator computed on the 1000 Monte Carlo samples described in Section 4.3 and that our approach performs better to match the discrepancy of the Monte Carlo samples. In addition, our approach shows better generative capacity exploring extremum values that have not been considered in the training set as shown in Figure 4.7 where our approach has good generative capacity on the density tails, since our model learn has learned from sparse and restricted data (Train) and was able to generate data similar to the Monte Carlo framework on the physical model, the latter is not accessible in practice unlike the model in our approach where the simulations are fast. Figure 4.8 shows that our approach is more accurate to compute threshold crossing probabilities.

4.5 Conclusion

In this chapter we presented novel methods for parametric and non-parametric uncertainty quantification relying on physical submodels over an area of interest. We have empirically shown that our methods obtain comparable and slightly better estimation of physical fields than classical neural networks approaches, while reducing the dimensionality of the learning problem and thus reducing the training cost of our models by restricting our attention to the boundary of a submodel. We fulfill the necessary condition that the cost of each run of the physical submodel is smaller than the cost of running the full physical model. Better precision is reached in the parametric view, by using DCnR's. Besides, in situation where the parameters distribution is unknown (epistemic uncertainties), only non-parametric approaches are feasible. For that, using the Wasserstein-GAN as a boundary conditions generator, we showed a higher value of the discrepancy in the Monte Carlo sampling method compared to high-fidelity solutions, while keeping physical consistency thanks to the learned boundary conditions, thus offering better generative behavior in the exploration of density tails.

Chapter 5

Physics oriented data preprocessing for deep learning and optimization of deep learning architecture for physical data

Abstract

Convolutional Neural Networks (CNNs) are widely used in various fields, such as image classification, medical imaging analysis, and autonomous driving cars. However, these models have millions of parameters, making it challenging to install them on devices with limited memory. Compression techniques, such as model pruning and weight quantization, can reduce the number of parameters and complexity of the models, which also reduces overfitting and the time required for predictions and training. In this chapter, we present a method for compressing CNN models for Finite Element Method (FEM) physical data using a filter decomposition of each layer and preprocessing approaches to optimize data for CNN training. We validate our compressed models on physical data from an FEM model solving a 2D wave equation.

Résumé

Les réseaux de neurones convolutionnels (CNN) sont largement utilisés dans divers domaines, tels que la classification d'images, l'analyse d'imagerie médicale et les voitures autonomes. Cependant, ces modèles ont des millions de paramètres, ce qui rend difficile de les installer sur des appareils avec une mémoire limitée. Les techniques de compression, telles que l'élagage de modèle et la quantification de poids, peuvent réduire le nombre de paramètres et la complexité des modèles, ce qui réduit également le surapprentissage et le temps requis pour les prédictions et l'entraînement. Dans ce chapitre, nous présentons une méthode de compression de modèles CNN pour des données physiques de méthode des éléments finis (FEM) en utilisant une décomposition de filtre de chaque couche et des approches de prétraitement pour optimiser les données pour l'entraînement des CNN. Nous validons nos modèles compressés sur des données physiques provenant d'un modèle FEM résolvant une équation d'onde 2D.

Contents

5.1 Introduction . . . 86
5.2 Proposed Approach . . . 87
5.2.1 Background . . . 87
5.2.2 Canonical Polyadic Decomposition of convolutional filters . . . 87
5.2.3 Approximation of the CPD of convolutional layers . . . 89
5.2.4 A priori decomposed convolutional layers . . . 89
5.3 Weight sharing . . . 91
5.4 Time regularization . . . 92
5.5 Developped models . . . 92
5.5.1 Models annotations . . . 93
5.6 Physical Data preprocessing . . . 94
5.7 Numerical results . . . 97
5.7.1 2D Wave propagation with one source point and early stopping . . . 97
5.7.2 2D wave propagation with one source point . . . 102
5.7.3 2D wave propagation with four source points . . . 105
5.8 Conclusion . . . 110

5.1 Introduction

Convolutional neural networks are now seeing widespread use in a variety of fields, including image classification, facial and object recognition, medical imaging analysis, and many more. In addition, there are applications such as autonomous driving cars in which accurate forecasts in real time with a minimal lag are required. The present neural network designs include millions of parameters, which makes it difficult to install such complex models on devices that have limited memory. Compression techniques might be able to resolve these issues by decreasing the size of CNN models that are created by reducing the number of parameters that contribute to the complexity of the models. The overfitting phenomena will be reduced also. The time needed to make predictions or time required for training using the original Convolutional Neural Networks model would be cut significantly if there were fewer parameters to deal with. In Chen et al. [2015]; Pilipović et al. [2018] one can find surveys of compression techniques for CNN models as model pruning and weights quantization, in Hameed et al. [2022] the authors use Kronecker product decomposition for weights matrices for CNN to compress the models. In this chapter we present a method of compressing convolutional neural networks for FEM physical data relying on a decomposition of filters of each layers and adequate preprocessing of data, and approaches to optimize data from FEM models for CNN training. Afterward we validate our compressed models on physical data from a FEM model solving a 2D wave equation.

5.2 Proposed Approach

5.2.1 Background

Let us consider $W = (w_{i,o}) \in \mathbb{R}^{n_i,n_o}$ the weight matrix for a fully connected layer, bias vectors will be omitted in this section for simplification purpose, however, we can easily extend this approach by taking into account the bias. The output of a fully connected layer who takes as an input a vector $x \in \mathbb{R}^{n_i}$ is a vector $y \in \mathbb{R}^{n_o}$.

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{n_o} \end{bmatrix} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n_i} \\ \vdots & \ddots & \vdots \\ w_{n_o,1} & \cdots & w_{n_o,n_i} \end{bmatrix} \times \begin{bmatrix} x_1 \\ \vdots \\ x_{n_i} \end{bmatrix} \tag{5.1}$$

Where W is a dense matrix, usually initialized as $(\forall (i,o) \in [|1,n_i|] \times [|1,n_o|]) w_{i,o} \neq 0$. Let us consider $K \in \mathbb{R}^{k_x \times k_y}$ a 2D convolutional filter and $x \in \mathbb{R}^{n_i,n_o}$ a 2D image to which we would like to apply the filter K. For visualization let us consider the example where $k_x = k_y = 2$, $n_i = n_o = 3$, developpements in this section are written in the case of 2D convolutional filters but can easily be generalized for higher dimensions. Having : $K = \begin{bmatrix} k_{1,1} & k_{1,2} \\ k_{2,1} & k_{2,2} \end{bmatrix}$ and $x = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix}$, and considering the flattened vector of x, the flattened output of this convolutional layer is a vector $y \in \mathbb{R}^4$:

$$\begin{bmatrix} y_{1,1} \\ y_{1,2} \\ y_{2,1} \\ y_{2,2} \end{bmatrix} = \begin{bmatrix} k_{1,1} & k_{1,2} & 0 & k_{2,1} & k_{2,2} & 0 & 0 & 0 & 0 \\ 0 & k_{1,1} & k_{1,2} & 0 & k_{2,1} & k_{2,2} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{1,1} & k_{1,2} & 0 & k_{2,1} & k_{2,2} & 0 \\ 0 & 0 & 0 & 0 & k_{1,1} & k_{1,2} & 0 & k_{2,1} & k_{2,2} \end{bmatrix} \times \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \\ x_{2,1} \\ x_{2,2} \\ x_{2,3} \\ x_{3,1} \\ x_{3,2} \\ x_{3,3} \end{bmatrix} \tag{5.2}$$

Thus convolutional layers can be considered as sparse fully connected layers that have a lower computational complexity than dense layers.

5.2.2 Canonical Polyadic Decomposition of convolutional filters

Canonical Polyadic Decomposition (CPD) Kolda and Bader [2009]; Evert et al. [2022]; Hitchcock [1927] describes a tensor as a sum of rank one tensors. In contrast to the matrix scenario, the CPD of a low rank tensor is unique given mild assumptions. CPD's intrinsic distinctiveness makes it a strong tool in many applications, allowing for the extraction of component information from a signal of interest. The generalized eigenvalue decomposition (GEVD), which picks a tensor matrix subpencil and then computes the generalized eigenvectors of the pencil, is a common approach for algebraic calculation of a CPD.

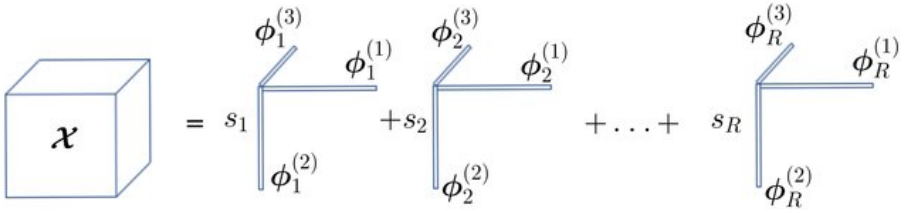


Figure 5.1: Canonical Polyadic Decomposition from Pham et al. [2018a]

Let us consider a CPD of the convolutional filter k defined in Section 5.2.1 with 2 vectors Q, R:

$$K \approx Q \otimes R^T = \begin{bmatrix} q_{1,1} \\ q_{1,2} \end{bmatrix} \otimes \begin{bmatrix} r_{1,1} & r_{2,1} \end{bmatrix} = \begin{bmatrix} q_{1,1}r_{1,1} & q_{1,1}r_{2,1} \\ q_{1,2}r_{1,1} & q_{1,2}r_{2,1} \end{bmatrix} \tag{5.3}$$

Where $\otimes$ denotes the outer vector product. Determining the values of Q, R will be discussed in a following section. Thus, the Equation 5.2 can be rewritten:

$$\begin{bmatrix} y_{1,1} \\ y_{1,2} \\ y_{2,1} \\ y_{2,2} \end{bmatrix} = \begin{bmatrix} q_{1,1}r_{1,1} & q_{1,1}r_{2,1} & 0 & q_{1,2}r_{1,1} & q_{1,2}r_{2,1} & 0 & 0 & 0 & 0 \\ 0 & q_{1,1}r_{1,1} & q_{1,1}r_{2,1} & 0 & q_{1,2}r_{1,1} & q_{1,2}r_{2,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{1,1}r_{1,1} & q_{1,1}r_{2,1} & 0 & q_{1,2}r_{1,1} & q_{1,2}r_{2,1} & 0 \\ 0 & 0 & 0 & 0 & q_{1,1}r_{1,1} & q_{1,1}r_{2,1} & 0 & q_{1,2}r_{1,1} & q_{1,2}r_{2,1} \end{bmatrix} \times \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \\ x_{2,1} \\ x_{2,2} \\ x_{2,3} \\ x_{3,1} \\ x_{3,2} \\ x_{3,3} \end{bmatrix} \tag{5.4}$$

Which can be rewritten as:

$$\begin{bmatrix} y_{1,1} \\ y_{1,2} \\ y_{2,1} \\ y_{2,2} \end{bmatrix} = \mathcal{Q} \times \mathcal{R} \times x = \begin{bmatrix} q_{1,1} & 0 & q_{1,2} & 0 & 0 & 0 \\ 0 & q_{1,1} & 0 & q_{1,2} & 0 & 0 \\ 0 & 0 & q_{1,1} & 0 & q_{1,2} & 0 \\ 0 & 0 & 0 & q_{1,1} & 0 & q_{1,2} \end{bmatrix} \times$$

$$\begin{bmatrix} r_{1,1} & r_{2,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{1,1} & r_{2,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{1,1} & r_{2,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_{1,1} & r_{2,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r_{1,1} & r_{2,1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_{1,1} & r_{2,1} \end{bmatrix} \times \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \\ x_{2,1} \\ x_{2,2} \\ x_{2,3} \\ x_{3,1} \\ x_{3,2} \\ x_{3,3} \end{bmatrix} \tag{5.5}$$

Where $\mathcal{R}$ is the matrix of the linear operation of applying the filter R to every row of $x \in \mathbb{R}^{3\times3}$ and $\mathcal{Q}$ is the matrix of the linear operation of applying the filter Q to every column of the output of applying R to x, then we have rewritten the 2D convolutional layer of x by the filter K to successfully applying two 1D filters, induced by the CPD of K, to different dimensions of x.

5.2.3 Approximation of the CPD of convolutional layers

In the previous section we presented an approach to reduce the dimensionality of a trained convolutional layer using CPD of each filter, the main issue remaining is determining the components of each decomposition. Generalized Eigenvalue Decomposition Domanov and Lathauwer [2014] is the default go-to approach. In this work we propose an equivalent approach relying on Singular Values Decomposition of filters. Let us first consider the case of 2D filters, let $K \in \mathbb{R}^{n_1 \times n_2}$ be a 2D filter. The SVD of K is written as :

$$K \approx \sum_{i=1}^{r} \sigma_i K_i^{(L)} \otimes K_i^{(R)}. \tag{5.6}$$

Where r is the number of singular values considered, $(\sigma_i)_{(i \leq r)}$ the singular values and $(K_i^{(L)}, K_i^{(R)})$ respectively the left and right singular vectors. Equation 5.6 can be rewritten as:

$$K \approx \sum_{i=1}^{r} K_i'^{(L)} \otimes K_i'^{(R)}. \tag{5.7}$$

Where each vector $K_i'^{(.)} = \sqrt{\sigma_i} K_i^{(.)}$, and by construction, $(K_i'^{(L)}, K_i'^{(R)})$ are rank one vectors, thus we obtained a Polyadic Decomposition of the filter, which is not canonical since the SVD is not unique, however it is a decomposition that fulfill a precision criteria which is sufficient for this application.

Let us now consider the case of 3D filters, let $K \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ a tensor of order 3, and let us consider $K_l = K[:,:,l]$, K can be written as:

$$K = \sum_{l=1}^{n_3} K_l \otimes I_l \tag{5.8}$$

Where $(I_l)_{l \leq n_3}$ are the vector rows of the identity matrix in $\mathbb{R}^{n_3 \times n_3}$, and each $(K_l)_{l \leq n_3}$ is a 2D matrix, thus a decomposition in the form of 5.7 is possible. Therefore K can be decomposed as:

$$K = \sum_{l=1}^{n_3} K_l \otimes I_l \approx \sum_{l=1}^{n_3} \sum_{i=1}^{r} K_{i,l}'^{(L)} \otimes K_{i,l}'^{(R)} \otimes I_l. \tag{5.9}$$

With straightforward work on sums indexes, one can rewrite this approximation as an unique sum of outer products of rank one vectors. Thus, we define a recursive method to build decompositions of a tensor of any dimension.

5.2.4 A priori decomposed convolutional layers

Applying this decomposition formalism to every kernel of convolutional layer leads to a compressed representation of the convolutional layer. We show that the a priori decomposition requires intermediate reshape and transpose operations on data. The new convolution approach is detailled in what follows.

Let us first consider $X \in \mathbb{R}^{n_B \times n_c \times n_t \times n_x}$ a tensor of physical 2D data, formed by n_B instances (or samples of data) and n_c channels (or features). In our target application, for each sample, n_t is the size of the temporal dimension of the data and n_x the size of the spatial dimension. Let $K \in \mathbb{R}^{n_f \times n_{kt} \times n_{kx}}$ where n_f is the number of 2D kernels considered, n_{kt} and n_{kx} respectively the kernels sizes in temporal and spatial dimension, and let $Y, \beta \in \mathbb{R}^{n_B \times n_f \times n_t' \times n_x'}$ being respectively the

output and the biases of the convolutional layer. Using the equation (2.32) defining the output of a convolution by a kernel we define the output Y of the convolution of X by K and β as :

$$Y(i,j,h,l) = \sum_{v=1}^{n_f} \sum_{o=1}^{n_{kt}} \sum_{p=1}^{n_{kx}} X(i,v,h+o-1,l+p-1)K(j,o,p) + \beta(i,j,h,l) \tag{5.10}$$

In these developpements we consider the case where all inputs channels are convolved with all output channels and default values are set for other convolution parameters ($stride = 1$, $padding = 0$, $dilation = 1$), generalizing these developpements for generic values or for higher dimension convolutional layers is a straightforward work on indexes or data structure. Let us consider a formal SVD of each kernel in K and only focusing on the first term of the sum (using more singular values for the decomposition can be performed by using more channels in the decomposed kernels):

$$j = 1, \;\; K[j,:,:] = K_j^t \otimes K_j^x + R[j,:,:] \tag{5.11}$$

where R is the residual tensor of the rank one approximation of K. Thus we can extract two subsets of 1D kernels forming 2 differents temporal and spatial 1D convolutional layers $K_j^t \in \mathbb{R}^{n_t}$ and $K_j^x \in \mathbb{R}^{n_x}$. Let us consider $\tilde{Y} \in \mathbb{R}^{n_B \times n_c \times n_t \times n'_x}$ the output of the spatial convolutional layer K_j^x on the columns of X without combining the ouptuts, we obtain:

$$\tilde{Y}(i,j,h,l) = \sum_{p=1}^{n_{kx}} X(i,(j,h),l+p-1)K_j^x(p) \tag{5.12}$$

where (j,h) is a multi index obtained by reshaping the data. In the sequel, the following transpose operation is also required:

$$\tilde{Y}^T(i,j,l,h) = \tilde{Y}(i,j,h,l) \tag{5.13}$$

Let us now consider $\hat{Y} \in \mathbb{R}^{n_B \times n_f \times n'_t \times n'_x}$ the output of the temporal convolutional layer K_j^t on the lines of $\tilde{Y}$ and combining the ouptuts and then adding the same biases $\beta \in \mathbb{R}^{n_B \times n_f \times n'_t \times n'_x}$, we obtain:

$$\begin{aligned}
\hat{Y}(i,\hat{j},h,l) &= \sum_{j=1}^{n_f} \sum_{o=1}^{n_{kt}} \tilde{Y}^T(i,(j,l),h+o-1)K_j^t(o) + \beta(i,\hat{j},h,l) \\
&= \sum_{j=1}^{n_f} \sum_{o=1}^{n_{kt}} \left(\sum_{p=1}^{n_{kx}} X(i,j,h+o-1,l+p-1)K_j^x(p) \right) K_j^t(o) + \beta(i,\hat{j},h,l) \\
&= \sum_{j=1}^{n_f} \sum_{o=1}^{n_{kt}} \sum_{p=1}^{n_{kx}} X(i,j,h+o-1,l+p-1) \left(K_j^x(p)\, K_j^t(o) \right) + \beta(i,\hat{j},h,l)
\end{aligned} \tag{5.14}$$

where $K_j^x(p)\, K_j^t(o)$ is the rank-one tensor approximation of the 2D convolution kernel. It follows that:

$$\begin{aligned}
&Y(i,\hat{j},h,l) - \hat{Y}(i,\hat{j},h,l) = \\
&\qquad \sum_{j=1}^{n_f} \sum_{o=1}^{n_{kt}} \sum_{p=1}^{n_{kx}} X(i,j,h+o-1,l+p-1)\, R(j,o,p)
\end{aligned} \tag{5.15}$$

Therefore the smaller R, the smaller the discrepancy between Y and $\hat{Y}$.

Thus, using decompositions of kernels of a 2D convolutinal layers we obtained an approximation of the output of said layer by using the decomposed kernels in two consecutive 1D convolutinal

layers with index manipulations via data reshaping and transpose operations.

Similarly, the rank-one decomposition can be extended to d-way kernels:

$$\begin{aligned}\hat{Y}(i,\hat{j},h_1,\ldots,h_d) = &\sum_{j=1}^{n_f}\sum_{p_1=1}^{n_{k1}}\ldots\sum_{p_d=1}^{n_{kd}} \\ &X(i,j,h_1+p_1-1,\ldots,h_d+p_d-1) \\ &\Pi_{k=1}^{d}K_j^k(p_k) \\ &+\beta(i,\hat{j},h_1,\ldots,h_d)\end{aligned} \tag{5.16}$$

Thus reducing the number of trainable parameters of the convolutional layer from $\Pi_{k=1^d n_{kk}}$ to $\sum_{k=1}^{d} n_{kk}$. Therefore we decomposed a model which complexity is exponentialy dependent on the dimension of the problem to a model which complexity is linealy dependent on the dimension, thus alienating the curse of dimensionality. In addition, since we approximate d-way kernels with a decomposition of d 1D convolutional layer, we can consider each layer appart and use activation functions after each 1D convolution, thus constructing a non linear decomposition of d-way kernels. Indeed this a priori decomposition of CNNs assume that parameters on each kernel are dimensionaly separable, nevertheless the update of those parameters while training the CNN depends on every dimension of the problem solved since the gradients backpropagated depends on all dimensions, then the dimensional separability of the problem is not a necessary hypothesis to perform such decomposition. Nevertheless, the precision of such decomposition is not an indicator of the precision of training models using this new form of convolutions. But it will rather be inforced by solving the optimization problem related to the objective function.

5.3 Weight sharing

Weight sharing or layer coupling is a deep learning model order reduction method in which multiple models which objective is to extract different features from same inputs share the first extraction layers. In Xie et al. [2021] the authors present an overview of different implementation and optimization method of weight sharing, as for Pham et al. [2018b] in which the authors present an approach to builds a large computational graph with each subgraph representing a neural network design, requiring all architectures to share their parameters. A policy gradient is used to train a controller to find the subgraph that maximizes the reward on a validation set. Liu and Tuzel [2016] propose a weight sharing approach for Generative Adversarial Network to learn joint distribution.

In our approach, weight sharing is a relevent choice, since the objective is to predict different physical fields with respect to the same input parameters. Thus, instead of training two different models, we train coupled models, therefore reducing the number of parameters in the first layers by half. The weight of the shared layers are updated with gradients induced by the minimization of the empirical risk for all physical fields predicted, the last regressive layers are updated by gradients induced by the only physical field predicted.

Let us denote M_d, M_v the models trained to approach the displacement and velocity fields using weight sharing, let us denote N_s the operation of applying the shared layers on a parameters vector input p, and N_d, N_v the operation of applying the remaining layers for the displacement and velocity models to the outputs of N_s. Outputs of both models can be written as:

$$\begin{cases} M_d(p) = N_d(N_s(p)) \\ M_v(p) = N_v(N_s(p)) \end{cases} \tag{5.17}$$

5.4 Time regularization

In our approach, since the objective is to predict dynamic physical fields, predicting fields that are dynamically linked is quite frequent, as for example predicting displacement and velocity fields. We propose an approach for time regularization of the predicted fields by adding a residual minimization in the cost function, which is quite similar to the approaches in Physics Informed Neural Network Raissi et al. [2017b] in which the residual of the Partial Differential Equation is minimized. Since for FEM models access to the residual of the PDE solved is intrusive and often infeasible, we rely on the temporal regularity of the approximated physical fields.

Let us consider P and T sets of collocation points for the time residual computing, and let M_u and M_v be two neural networks which objective is to predict displacement and velocity fields for the same FEM models parameterized by p. We define the time residual as :

$$\mathop{\mathbb{E}}_{p\in P}\mathop{\mathbb{E}}_{t'\in T}|\frac{\partial M_u(p,t')}{\partial t}-M_v(p,t')||_2^2 \tag{5.18}$$

For convolutional layers, computing the derivative of the output with respect to the inputs is not always feasible, so we approach the time derivative with an Euler finite difference scheme. In the following this regularization will be mentionned as the Euler regularization. Eventhough this approach was not implemented and tested for Generative Adversarial Networks training, but only for regression models, we propose the algorithm for training GANs with Euler regularization.

Algorithm 5: WGAN with gradient penalty and Euler regularization. We use default values of $\lambda = 10$, $n_{critic} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$

Require: The gradient penalty coefficient λ, the number of critic iteration per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2
Require: Initial critic parameters ω_0, inital generator parameters θ_0
while θ has not converged **do**
  for t = 1,..., n_{critic} **do**
    for i = 1,..., m **do**
      Sample real data x =(U,V) from training sets, latent variable $z \sim \mathcal{N}(0,1)$, a random number $\varepsilon \sim \mathcal{U}[0,1]$
      $\tilde{x} \leftarrow G_\theta(z)$
      $\hat{x} \leftarrow \varepsilon\tilde{x} + (1-\varepsilon)x$
      $L^{(i)} \leftarrow D_\omega(\tilde{x}) - D_\omega(x) + \lambda(||\nabla_{\hat{x}} D_\omega(\hat{x})||_2 - 1)^2$
    end for
    $\omega \leftarrow Adam(\nabla_\omega \frac{1}{m}\sum_{i=1}^m L^{(i)}, \omega, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{z^{(i)}\}_{i=1}^m \sim \mathcal{N}(0,1)$
  $U_i, V_i \leftarrow G_\theta(z_i)$
  $r_{Euler}(z_i) \leftarrow ||\frac{U_i(t+\Delta t)-U_i(t)}{\Delta t} - V_i(t)||_2^2$
  $\theta \leftarrow Adam(\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_\omega(G_\theta(z_i)) + r_{Euler}(z_i), \theta, \alpha, \beta_1, \beta_2)$
end while

5.5 Developped models

All models in this following work can be categorized into 9 categories depending on how they process each type of physical data and how spatial and temporal information is used to update the

weights at each training step. So for spatial and temporal information we distinguish 3 categories each:

- Sampled: only one sample of spatial or/and temporal data is used to update the weights, the models takes as inputs the spatial or temporal coordinates or both.
- Local: only a local amount of spatial or/and temporal data is used to update the weights, the model predicts the whole simulation but uses convolutional layers to treat data localy.
- Global: the whole information across the spatial or temporal information or both is used to update the weights, this is achieved in a convolutional layer by considering the global dimension as the channels or with a fully connected layer.

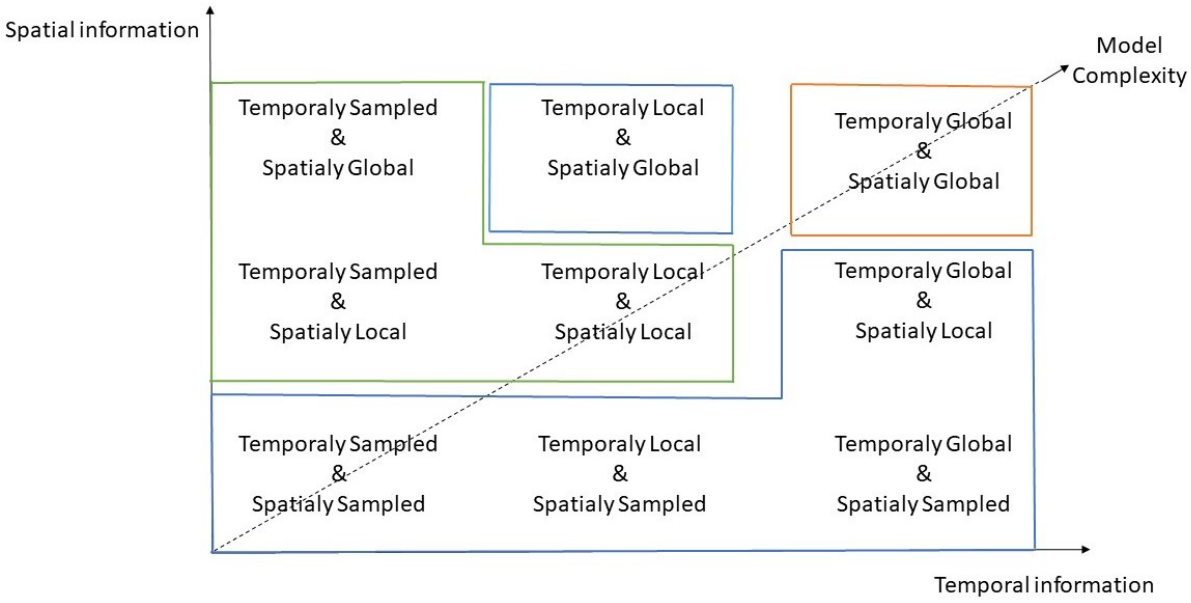


Figure 5.2: Model complexity evolution

Figure 5.2 shows the evolution of model complexity according to the approaches considered spatialy and temporally, in green approaches that were considered and developed in this work, in blue feasible approaches but hold no interest whatsoever to our application and in orange non-feasible approaches.

5.5.1 Models annotations

Regarding our developed model, time sampled approaches or time conditioned approaches will be suffixed by "*_t*", spatialy local approaches are the convolutional networks that will be prefixed by "*Conv*" followed by their dimension, local and global temporal approaches can be distinguished by the dimension of the convolutional net, for example using a 3D convolutional network to approach a 2D spatial and 1D physical field is temporally local, using a 2D convolutional layer to approach the same field is temporally global. All fully connected models developed are spatialy global and temporally sampled. All of these models can have multiple variants considering if convolutional decomposition is performed a suffixe "N.5D" means that a convolutional layer of

dimension $N+1$ has been decomposed using the decomposition approach presented in this chapter, $2.5D$ means that the model has been decomposed to a 2D spatial layer and a 1D temporal layer, $2.5Db$ means that a 3D model has been decomposed to three 1D layers. Additional variants appears depending on the regularization technique used, Batch Normalization Ioffe and Szegedy [2015b] is designated by "*BN*", Euler regularization by "*E*", Weight sharing or layer sharing by "*LR*", "*BASIC*" will denote a network with no regularization applied. All these type of regularization can be combined except for Batch Normalization and Euler, since Batch Normalization makes the derivatives computed in Euler Regularization erroneous.

5.6 Physical Data preprocessing

One of the approaches developed in this chapter is the construction of submodels using data from a finite element model (FEM). This method rely on the general principle which is a submodel formed of two components:

- A neural network learning boundary conditions around a predetermined zone of interest.
- A finite element submodel in the zone of interest using boundary conditions generated by the neural network. We assume that there is no modeling error in the zone of interest covered by the proposed submodel.

For the validation of our approach we use the same FEM simulation of the 2D wave equation defined in Section 4.3

We choose to solve the 2D wave equation. We define two 2D Cartesian space grids Ω_h and Ω'_h, with $\Omega'_h \subset \Omega_h$ representing the zone of interest. Ω_h and Ω'_h are triangular space discretization of sizes $[N_x, N_y]$ and $[N'_x, N'_y]$ of the domains $[-L_x, L_x] \times [-L_y, L_y]$ and $[-L'_x, L'_x] \times [-L'_y, L'_y]$. And finally a temporal grid T is defined as discretization of size N_T of the space $[0, T_{final}]$ and the time step $\Delta t = \frac{T_{final}}{N_T - 1}$. The 2D wave equation is given as follows:

$$\begin{cases} \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} - \Delta u = f \ \ on \ \ \Omega \ \ \forall t > 0 \\ u = 0 \ \ on \ \ \partial\Omega \ \ \forall t > 0 \\ u = u_0 \ \ on \ \ \Omega \ \ for \ \ t = 0 \end{cases} \tag{5.19}$$

where u is the amplitude of the wave. A source point is determined for the problem resolution where (x_S, y_S) are the source point coordinates, it is chosen to be outside the zoom domain: i.e. $(x_S, y_S) \in \Omega \backslash \Omega'$.
The source term at the right hand side of the wave equation is set as:

$$(\forall t \in T), \ \ f(x,y,t) = sin(\omega t)\delta_{(x_S,y_S)}(x,y) \tag{5.20}$$

where $\delta_{(x_S,y_S)}$ denotes the 2D Dirac distribution centered at (x_S, y_S). A three-dimensional parameter vector $p = (\omega, x_s, y_s)$ is chosen and determined then sampled, (note that c is constant over all samples since it is a parameter needed for the submodel). Sampling is done using latin hypercube sampling routines. For every parameter vector p a simulation vector $U(p)$ is generated using FEM model described in section 4.3. One sample of data is then$(p, U(p))$ where $p \in D_p \subset \mathbb{R}^3$ and $U(p)(t) \in V_h \subset \mathbb{R}^{N_X \times N_Y}$.
Then, 2 datasets are generated from the same uniform distribution as the following:

- Training data set: 100 samples generated, used for training parametric approach models.

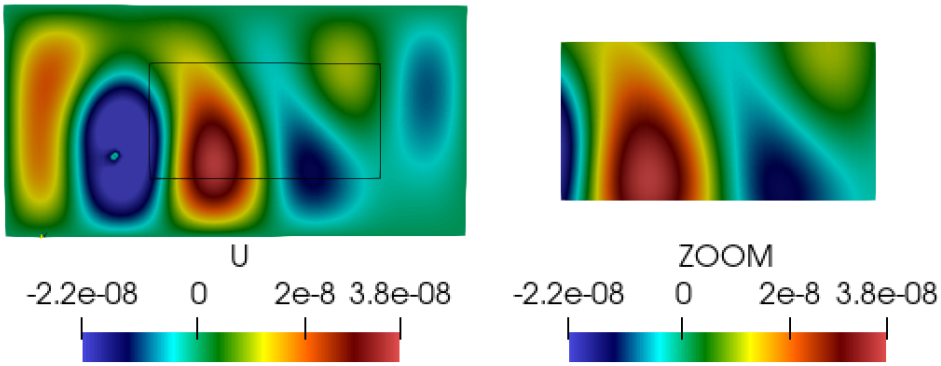


Figure 5.3: Visualization of the FEM output on Ω and $\Omega^{'}$

- Test data set: 25 samples generated, used for testing the training process of each neural network, and comparing the models we used in our study. All generated data are scaled with a standard scaler.

Since our approach requires the extraction and the processing of boundary conditions in the following we presents the approach adopted for our boundary data. For each parametric simulation of the wave equation we extract the boundary values as 1D vectors aligned in a specific order to maintain physical local information for the convolutional layers as shown in Figure 5.4. An issue rises concerning the physical locality of information for the extremities node, to solve this issue we use circular padding in the extremities in the convolution layers, fully connected layers do not suffer from this issue. Learning boundary data instead of the whole spatial information reduces the learning problem dimensionality in addition to the fact that the submodeling approach maintains physical properties in the area of interest.

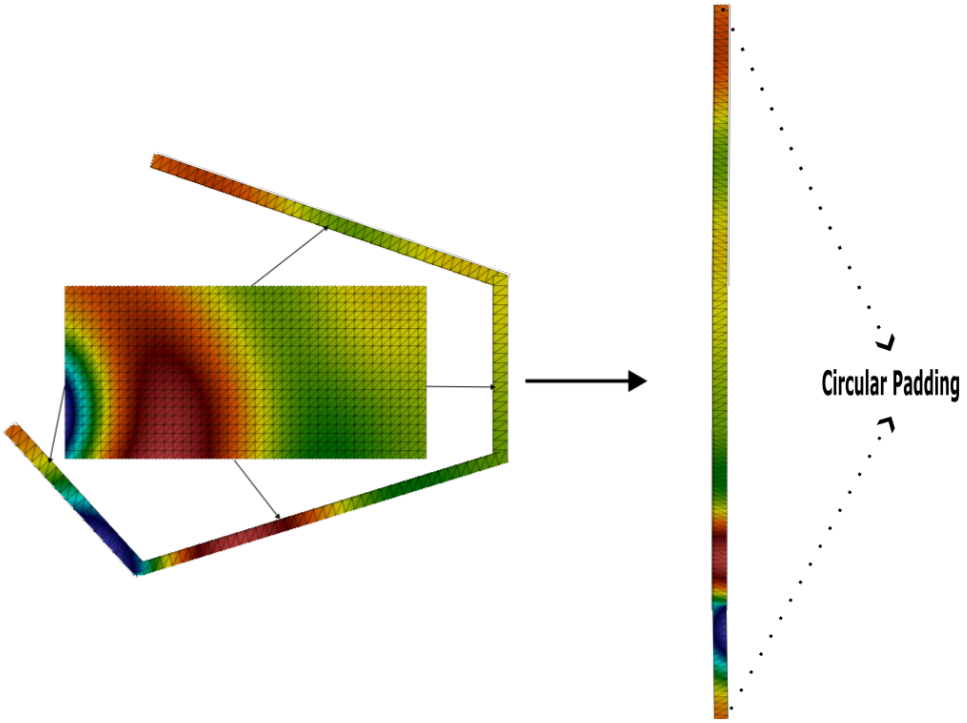


Figure 5.4: Boundary Data Extraction

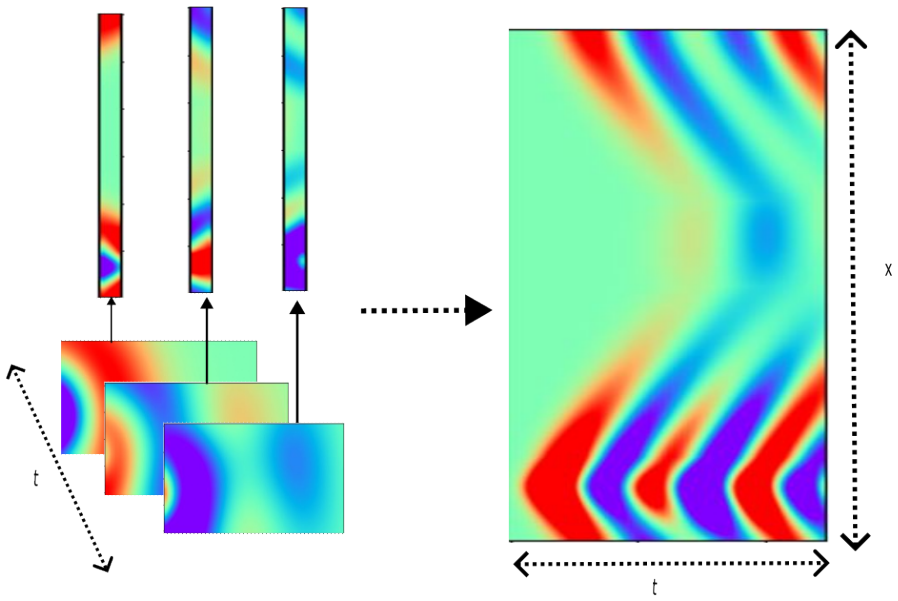


Figure 5.5: Temporal Boundary Data Extraction

Thus for each time step for a fixed parametric value, we can extract a boundary vector. All

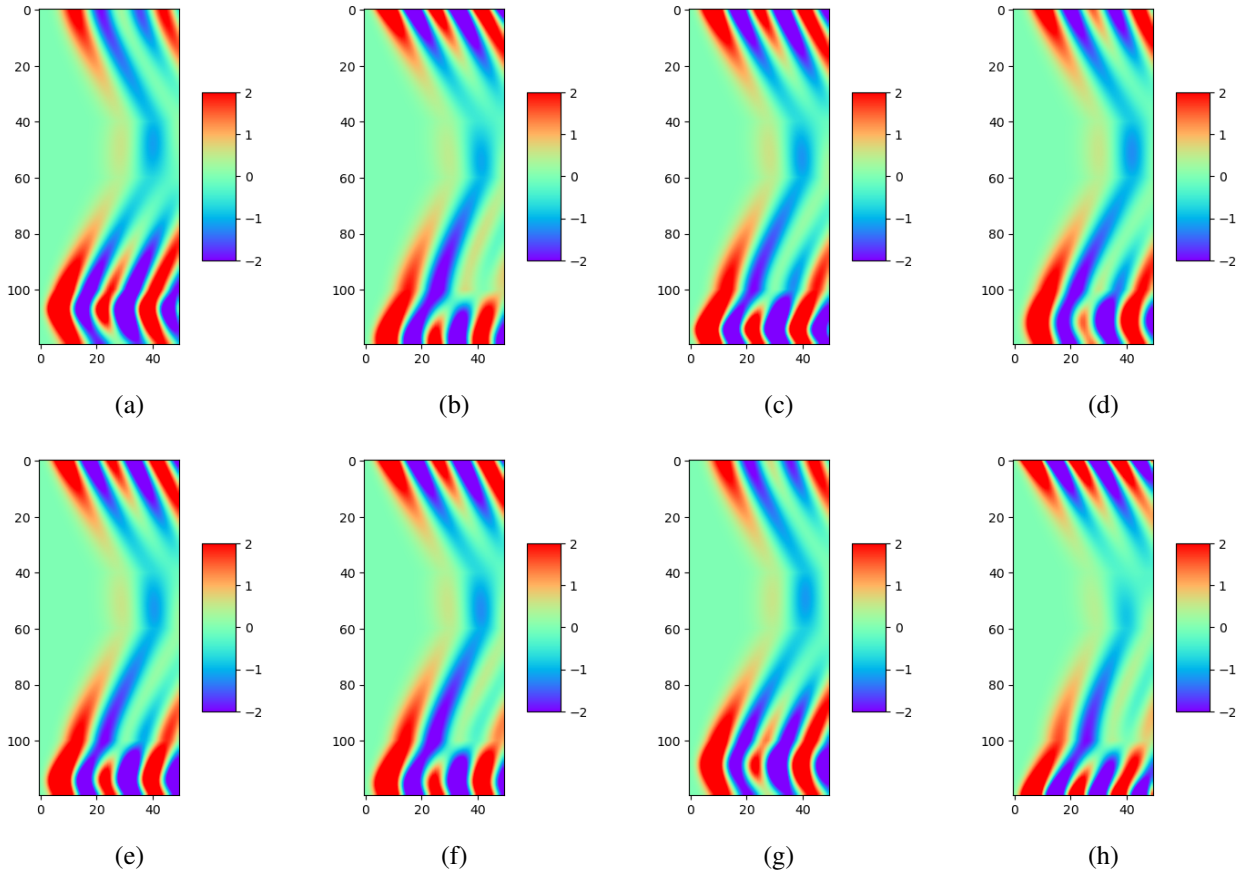


Figure 5.6: Boundary data for different parametric values

boundary vectors for a particular parametric value p can then be aggregated in one 2D matrix in which the lines represents the temporal evolution of values and the column the physical evolution of values, as shown in Figure 5.5. Then, the boundary data for the whole simulation can be viewed as a 2D image by convolutional layers. Figure 5.6 shows samples of boundary data for different parametric values.

5.7 Numerical results

5.7.1 2D Wave propagation with one source point and early stopping

We trained all regression models defined in the previous section on both boundary and full data for 1000 epoch using Adam Kingma and Ba [2014] optimizer and a learning rate of $1e^{-3}$. In all tables, results in bold black indicate best results, results in blue indicate results within an acceptable threshold, and results in red indicate unacceptable results.

Table 5.1: Number of trainable parameters (million parameters)

	Basic	BN	E	SL	BN & SL	E & SL
FC t	14.34	14.37	14.34	14.22	14.26	14.22
Conv2D	23.30	23.31	23.30	12.63	12.64	12.63
Conv2D t	1.52	1.52	1.52	0.81	0.81	0.81
Conv3D	7.25	7.25	7.25	3.83	3.83	3.83
Conv2.5D	2.72	2.72	2.72	1.46	1.46	1.46
Conv2.5Db	1.63	1.64	1.63	0.85	0.86	0.85
FC t Boundary	0.27	0.28	0.27	0.26	0.26	0.26
Conv1D Boundary	6.98	6.99	6.98	3.65	3.66	3.65
Conv1D t Boundary	0.45	0.45	0.45	**0.24**	0.24	**0.24**
Conv2D Boundary	2.19	2.19	2.19	1.13	1.13	1.13
Conv1.5D Boundary	1.18	1.19	1.18	0.61	0.62	0.61

Table 5.1 shows the number of trainable parameters for each model combined with every regularization technique described before. As expected fully connected layers and approaches using temporal dimension as feature maps have the highest number of parameters, and our decomposition approach is efficient in reducing the number of trainable parameters in each model, Sharing Layers approach reduce drastically the number of parameters in convolutional layer, fully connected layers cannot benefit from this approach since first layers have the lowest number of parameters. Batch norm regularization only add a negligible number parameters.

Table 5.2: Train error on full displacement

	Basic	BN	E	SL	BN & SL	E & SL
FC t	2.27	0.54	2.28	2.02	0.55	2.02
Conv2D	4.61	**0.22**	4.86	4.90	0.43	4.66
Conv2D t	26.91	0.41	26.91	26.91	0.31	26.92
Conv3D	5.87	0.24	5.76	6.08	0.27	5.86
Conv2.5D	4.64	0.34	5.11	4.55	0.37	5.35
Conv2.5Db	4.66	0.54	5.37	4.27	0.79	0.61

Table 5.3: Train error on full velocity

	Basic	BN	E	SL	BN & SL	E & SL
FC t	2.26	0.63	2.42	2.27	0.60	2.25
Conv2D	4.30	**0.18**	4.55	4.55	0.22	4.31
Conv2D t	21.69	0.25	21.69	21.68	0.27	21.69
Conv3D	6.06	0.23	5.83	6.02	0.26	5.96
Conv2.5D	4.44	0.25	4.78	4.33	0.43	4.81
Conv2.5Db	4.47	0.63	4.80	4.18	0.58	0.54

Table 5.4: Test error on full displacement

	Basic	BN	E	SL	BN & SL	E & SL
FC t	1.34	0.37	1.37	1.24	0.46	1.29
Conv2D	3.25	0.24	3.15	3.22	0.31	3.46
Conv2D t	24.45	0.28	24.46	24.47	0.29	24.47
Conv3D	4.72	**0.18**	4.66	4.64	0.26	4.81
Conv2.5D	3.47	0.28	3.30	3.50	0.38	3.20
Conv2.5Db	3.33	0.43	2.81	3.72	0.73	0.53

Table 5.5: Test error on full velocity

	Basic	BN	E	SL	BN & SL	E & SL
FC t	1.60	0.43	1.61	1.55	0.48	1.55
Conv2D	3.02	0.18	2.88	2.95	0.24	3.19
Conv2D t	20.56	0.25	20.55	20.55	0.26	20.57
Conv3D	4.93	**0.17**	5.02	5.02	0.23	5.14
Conv2.5D	3.22	0.24	3.01	3.29	0.43	3.14
Conv2.5Db	2.92	0.58	2.60	3.30	0.54	0.48

Tables 5.2 and 5.3 show respectively the training error of each model variant for full data prediction on Ω considering all the possible regularization methods described in the previous sections. The tables indicate that the model that achieves the best accuracy in training is *Conv2D* with BatchNorm regularization, but achieves poor generalization error as shown by Tables 5.4 and 5.5 where best compromise between training error and generalization error is achieved by *Conv3D*, this behavior can be explained by the overfitting occurring in *Conv2D* considering the large amount of parameters within the model. Tables 5.2, 5.3, 5.4 and 5.5 show that the approaches using our convolutional decomposition presented in this chapter achieve comparable training precision and generalization error with much fewer parameters as shown by Table 5.1. Only the models with the Batch Normalization regularization could achieve acceptable error threshold for training with few epoch and a high learning rate. Euler regularization shows no improvements on the precision of the models, however combined with sharing layers it achieves equivalent error threshold to the Batch Normalization results.

Table 5.6: Train error on displacement after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	2.01	0.49	2.06	1.83	0.47	1.88
Conv2D	4.75	0.13	4.80	4.74	0.17	4.61
Conv2D t	27.45	0.25	27.46	27.44	**0.13**	27.44
Conv3D	4.61	0.22	4.47	4.57	0.20	4.46
Conv2.5D	4.72	0.19	4.94	4.57	0.16	4.86
Conv2.5Db	4.69	0.28	5.07	4.43	0.51	0.29
FC t Boundary	1.59	0.48	1.64	1.58	0.41	1.48
Conv1D Boundary	4.62	0.19	4.80	4.70	0.14	4.84
Conv1D t Boundary	27.45	0.30	27.45	27.44	0.38	27.45
Conv2D Boundary	11.25	0.15	11.59	11.22	0.21	11.36
Conv1.5D Boundary	5.38	0.18	5.57	5.60	0.18	5.62

Table 5.7: Train error on velocity after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	2.80	0.65	2.85	2.64	0.65	2.71
Conv2D	4.42	0.12	4.47	4.41	0.15	4.29
Conv2D t	22.49	0.18	22.49	22.48	**0.09**	22.47
Conv3D	4.57	0.18	4.44	4.52	0.18	4.44
Conv2.5D	4.45	0.21	4.66	4.35	0.17	4.56
Conv2.5Db	4.42	0.39	4.66	4.23	0.55	0.37
FC t Boundary	2.35	0.62	2.42	2.28	0.54	2.21
Conv1D Boundary	4.30	0.16	4.45	4.37	0.13	4.48
Conv1D t Boundary	22.49	0.21	22.49	22.48	0.24	22.49
Conv2D Boundary	9.58	0.11	9.85	9.52	0.17	9.62
Conv1.5D Boundary	5.80	0.20	5.93	5.95	0.18	5.83

Table 5.8: Test error on displacement after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	1.34	0.37	1.37	1.24	0.46	1.29
Conv2D	3.25	0.24	3.15	3.22	0.31	3.46
Conv2D t	24.45	0.28	24.46	24.47	0.29	24.47
Conv3D	4.72	0.18	4.66	4.64	0.26	4.81
Conv2.5D	3.47	0.28	3.30	3.50	0.38	3.20
Conv2.5Db	3.33	0.43	2.81	3.72	0.73	0.53
FC t Boundary	1.02	0.22	1.08	1.06	0.24	1.07
Conv1D Boundary	5.01	0.13	4.84	4.94	**0.08**	4.79
Conv1D t Boundary	25.27	0.26	25.28	25.27	0.36	25.28
Conv2D Boundary	13.28	0.12	13.17	13.04	0.13	13.21
Conv1.5D Boundary	5.99	0.29	6.19	6.16	0.14	6.03

Table 5.9: Test error on velocity after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	2.13	0.43	2.28	2.11	0.43	2.08
Conv2D	2.84	0.14	2.80	2.90	0.19	3.02
Conv2D t	21.76	0.09	21.77	21.76	0.09	21.74
Conv3D	3.90	0.13	3.98	3.99	0.10	4.02
Conv2.5D	3.00	0.17	2.97	3.06	0.13	3.11
Conv2.5Db	2.97	0.30	2.77	3.26	0.43	0.39
FC t Boundary	1.77	0.35	1.76	1.65	0.38	1.79
Conv1D Boundary	4.39	0.12	4.25	4.27	**0.07**	4.20
Conv1D t Boundary	21.76	0.16	21.77	21.76	0.24	21.76
Conv2D Boundary	11.51	0.12	11.38	11.26	0.12	11.40
Conv1.5D Boundary	6.29	0.29	6.38	6.46	0.15	6.42

Tables 5.6, 5.7, 5.8 and 5.9 show respectively the training error and generalization error of each model variant for full and boundary data prediction after the zoom operation in the area of interest considering all the possible regularization methods described in the previous sections. The tables indicate that the model that achieves the best training and generalization are boundary generative models, achieving this results with even fewer trainable parameters, the convolutional decomposition approaches achieve equivalent results with fewer trainable parameters.

Table 5.10: Average time of one epoch

	Basic	BN	E	SL	BN & SL	E & SL
FC t	0.49	0.49	0.53	0.42	0.46	0.51
Conv2D	0.60	0.63	0.72	0.43	0.44	0.53
Conv2D t	0.37	0.40	0.65	0.34	0.37	0.62
Conv3D	0.32	0.36	0.78	0.26	0.30	0.73
Conv2.5D	0.69	0.75	1.07	0.68	0.75	1.03
Conv2.5Db	1.10	1.21	1.67	1.05	1.08	1.53
FC t Boundary	0.20	0.21	0.26	**0.19**	0.20	0.25
Conv1D Boundary	0.35	0.36	0.46	0.29	0.30	0.40
Conv1D t Boundary	0.34	0.36	0.51	0.35	0.36	0.54
Conv2D Boundary	0.22	0.23	0.30	0.19	0.21	0.28
Conv1.5D Boundary	0.69	0.73	1.05	0.65	0.69	1.03

The table 5.10 shows the average time for each model to train for one epoch, it indicates that the convolutional decomposition approaches have higher training time, this can be explained by comparing our first implementation with optimized implementation of the deep learning libraries, the objective to attain comparable results with decomposed convolutional layers was achieved but the optimization of training time is yet to be achieved.

5.7.2 2D wave propagation with one source point

We trained all models defined in the previous section on both boundary and full data for 10000 epoch using Adam Kingma and Ba [2014] optimizer and a learning rate of $1e^{-3}$ and learning rate decay to achieve a learning rate of $1e^{-4}$ at the last epoch.

Table 5.11: Train error on full displacement

	Basic	BN	E	SL	BN & SL	E & SL
FC t	0.88	0.21	0.86	0.76	0.23	0.75
Conv2D	0.11	0.10	0.13	0.12	**0.08**	0.13
Conv2D t	0.22	0.19	0.24	0.25	0.21	0.28
Conv3D	0.12	0.09	0.12	0.14	0.11	0.15
Conv2.5D	0.15	0.15	0.14	0.13	0.12	0.17
Conv2.5Db	0.15	0.16	0.16	0.16	0.16	0.19

Table 5.12: Train error on full velocity

	Basic	BN	E	SL	BN & SL	E & SL
FC t	0.82	0.28	0.83	0.83	0.24	0.80
Conv2D	0.12	**0.08**	0.12	0.11	0.08	0.11
Conv2D t	0.25	0.19	0.20	0.23	0.22	0.22
Conv3D	0.14	0.10	0.13	0.13	0.12	0.15
Conv2.5D	0.14	0.16	0.13	0.13	0.13	0.16
Conv2.5Db	0.15	0.17	0.12	0.16	0.18	0.18

Table 5.13: Test error on full displacement

	Basic	BN	E	SL	BN & SL	E & SL
FC t	0.92	0.27	0.90	0.80	0.28	0.78
Conv2D	1.12	0.77	1.22	1.13	0.79	1.11
Conv2D t	0.24	0.22	0.26	0.27	0.23	0.31
Conv3D	0.23	0.20	0.24	0.23	**0.19**	0.24
Conv2.5D	0.55	0.64	0.54	0.53	0.58	0.51
Conv2.5Db	0.66	0.63	0.63	0.54	0.68	0.68

Table 5.14: Test error on full velocity

	Basic	BN	E	SL	BN & SL	E & SL
FC t	0.86	0.34	0.86	0.87	0.30	0.82
Conv2D	1.12	0.81	1.24	1.09	0.74	1.11
Conv2D t	0.26	0.21	0.22	0.25	0.23	0.24
Conv3D	0.28	0.21	0.32	0.22	**0.20**	0.24
Conv2.5D	0.54	0.63	0.50	0.49	0.55	0.48
Conv2.5Db	0.70	0.72	0.76	0.50	0.64	0.64

Tables 5.11 and 5.12 shows respectively the training error of each model variant for full data prediction considering all the possible regularization methods described in the previous sections. The tables indicate that the model that achieves the best accuracy in training is *Conv2D* with BatchNorm regularization, but achieves poor generalization error as shown by Tables 5.13 and 5.14 where best compromise between training error and generalization error is achieved by *Conv3D*, this behavior can be explained by the overfitting occuring in *Conv2D* considering the large amount of parameters within the model. Tables 5.11, 5.12, 5.13 and 5.14 show that the approaches using the convolutional decomposition presented in this chapter achieve comparable training precision and generalization error with much fewer parameters as shown by Table **??**. Results here show that all model could converge thanks to the higher number of epochs and learning rate decay, thus

combining Batch Normalization within our decomposition approach allow the models to learn and have good generalization error with high learning rate and few epochs.

Table 5.15: Train error on displacement after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	0.79	0.17	0.75	0.64	0.19	0.61
Conv2D	0.04	0.03	0.05	0.05	0.03	0.05
Conv2D t	0.12	0.10	0.12	0.13	0.11	0.16
Conv3D	0.06	0.04	0.05	0.06	0.06	0.08
Conv2.5D	0.07	0.08	0.05	0.07	0.06	0.09
Conv2.5Db	0.07	0.08	0.07	0.08	0.07	0.12
FC t Boundary	0.55	0.17	0.56	0.52	0.17	0.51
Conv1D Boundary	0.03	**0.02**	0.03	0.04	0.02	0.05
Conv1D t Boundary	0.09	0.08	0.07	0.12	0.11	0.11
Conv2D Boundary	0.04	0.06	0.06	0.06	0.06	0.05
Conv1.5D Boundary	0.05	0.02	0.04	0.05	0.03	0.04

Table 5.16: Train error on velocity after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	1.23	0.24	1.18	1.05	0.27	0.99
Conv2D	0.05	0.04	0.06	0.05	0.03	0.06
Conv2D t	0.12	0.12	0.12	0.16	0.13	0.17
Conv3D	0.07	0.05	0.06	0.07	0.06	0.09
Conv2.5D	0.08	0.07	0.07	0.09	0.07	0.10
Conv2.5Db	0.09	0.10	0.10	0.11	0.10	0.12
FC t Boundary	0.90	0.25	0.92	0.84	0.25	0.84
Conv1D Boundary	0.03	**0.02**	0.03	0.03	0.02	0.04
Conv1D t Boundary	0.09	0.09	0.08	0.11	0.11	0.12
Conv2D Boundary	0.04	0.05	0.06	0.06	0.07	0.05
Conv1.5D Boundary	0.05	0.03	0.05	0.05	0.04	0.05

Table 5.17: Test error on displacement after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	0.92	0.27	0.90	0.80	0.28	0.78
Conv2D	1.12	0.77	1.22	1.13	0.79	1.11
Conv2D t	0.24	0.22	0.26	0.27	0.23	0.31
Conv3D	0.23	0.20	0.24	0.23	0.19	0.24
Conv2.5D	0.55	0.64	0.54	0.53	0.58	0.51
Conv2.5Db	0.66	0.63	0.63	0.54	0.68	0.68
FC t Boundary	0.60	0.23	0.62	0.58	0.25	0.58
Conv1D Boundary	1.51	1.42	1.55	1.57	1.33	1.45
Conv1D t Boundary	0.13	0.11	**0.11**	0.16	0.14	0.15
Conv2D Boundary	0.30	0.24	0.28	0.25	0.23	0.25
Conv1.5D Boundary	0.78	1.07	1.03	0.96	0.85	1.02

Table 5.18: Test error on velocity after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	1.25	0.30	1.21	1.08	0.33	1.01
Conv2D	1.01	0.85	1.08	0.95	0.88	1.09
Conv2D t	0.16	0.16	0.15	0.19	0.17	0.20
Conv3D	0.18	0.18	0.19	0.19	0.16	0.20
Conv2.5D	0.61	0.76	0.58	0.60	0.69	0.54
Conv2.5Db	0.76	0.77	0.74	0.61	0.84	0.82
FC t Boundary	0.93	0.31	0.96	0.88	0.33	0.88
Conv1D Boundary	1.42	1.35	1.46	1.48	1.26	1.36
Conv1D t Boundary	0.14	0.13	**0.12**	0.16	0.14	0.16
Conv2D Boundary	0.31	0.24	0.29	0.25	0.23	0.25
Conv1.5D Boundary	0.75	1.03	0.96	0.90	0.80	0.95

Tables 5.15, 5.16, 5.17 and 5.18 show respectively the training error and generalization error of each model variant for full and boundary data prediction after the zoom operation in the area of interest considering all the possible regularization methods described in the previous sections. The tables indicate equivalent results as the results from the previous experiment, in the exception of all models achieve equivalent performance as previously explained. Models relying on Euler regularization achieve the best error threshold.

5.7.3 2D wave propagation with four source points

In this section we modify the 2D Wave equation used in the previous experiment to include 4 different source points, each with his own source frequency, the right handside of the equation becomes:

$$(\forall t \in T), \quad f(x,y,t) = \sum_{i=1}^{4} sin(\omega_i t)\delta_{(x_{Si},y_{Si})}(x,y) \tag{5.21}$$

We obtain a then a parametrization of out FEM models with a vector of size 12, a frequency and 2D coordination corresponding to each source points. The variable parameters identified are sampled following Table 5.19 values.

Table 5.19: Parameters sampling

P	Mean Value	Variation (%)	Min Value	Max Value
ω_1	5 kHz	5%	4.75 kHz	5.25 kHz
x_{S1}	-1.85 m	17.5% of L_x	-2.2 m	-1.8 m
y_{S1}	-0.65 m	17.5% of L_x	-0.2 m	0.2 m
ω_2	5 kHz	5%	4.75 kHz	5.25 kHz
x_{S2}	-1.85 m	17.5% of L_x	3.25 m	3.45 m
y_{S2}	-0.65 m	17.5% of L_x	-0.2 m	0.2 m
ω_3	5 kHz	5%	4.75 kHz	5.25 kHz
x_{S3}	-1.85 m	17.5% of L_x	-0.2 m	0.2 m
y_{S3}	-0.65 m	17.5% of L_x	-1.7 m	-1.3 m
ω_4	5 kHz	5%	4.75 kHz	5.25 kHz
x_{S4}	-1.85 m	17.5% of L_x	-0.2 m	0.2 m
y_{S4}	-0.65 m	17.5% of L_x	1.3 m	1.7 m

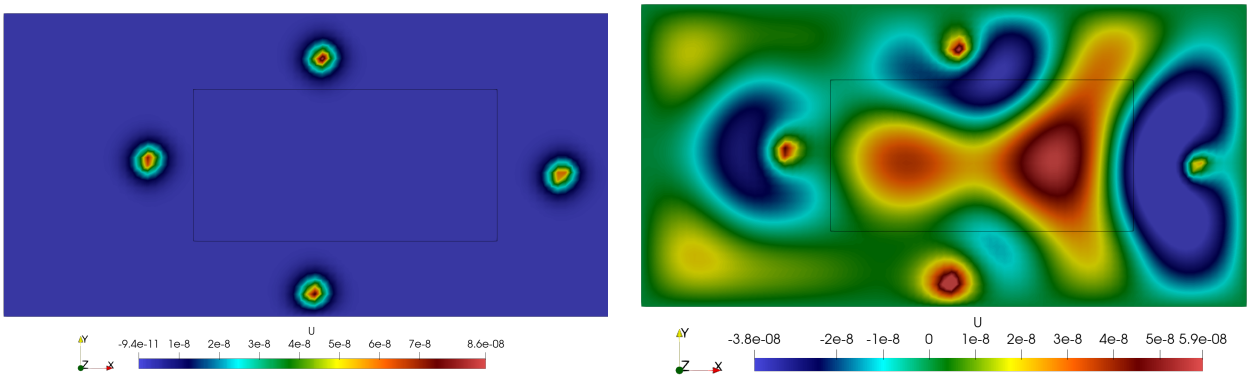
Figure 5.7 shows an example of output with this configuration, and Figure 5.8 shows samples of boundary data extracted as before. With the exception here that training points are sampled over the simulation temporal grid, we use 25% of time steps sampled uniformly from the simulation temporal grid. To set up the zoom operation we need prediction over the whole simulation temporal grid, thus for models that are temporally conditioned we use the regressive property of the models to generate data over the temporal grid, for temporally global approaches we use piecewise polynomial interpolation for each sample. Let M_u, M_v be temporally global models trained on the sampled displacement and velocity data, and p a parametric value from training or testing set, and t_1, t_2 two points from the training sampled temporal grid, we generalize the output of M_u and M_v for each time step $t \in [t_1, t_2]$ as:

$$\begin{aligned} M_u(p,t) &= \alpha_1 t^3 + \alpha_2 t^2 + \alpha_3 t + \alpha_4 \\ M_v(p,t) &= 3\alpha_1 t^2 + 2\alpha_2 t + \alpha_3 \end{aligned} \tag{5.22}$$

Where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are solution of the problem:

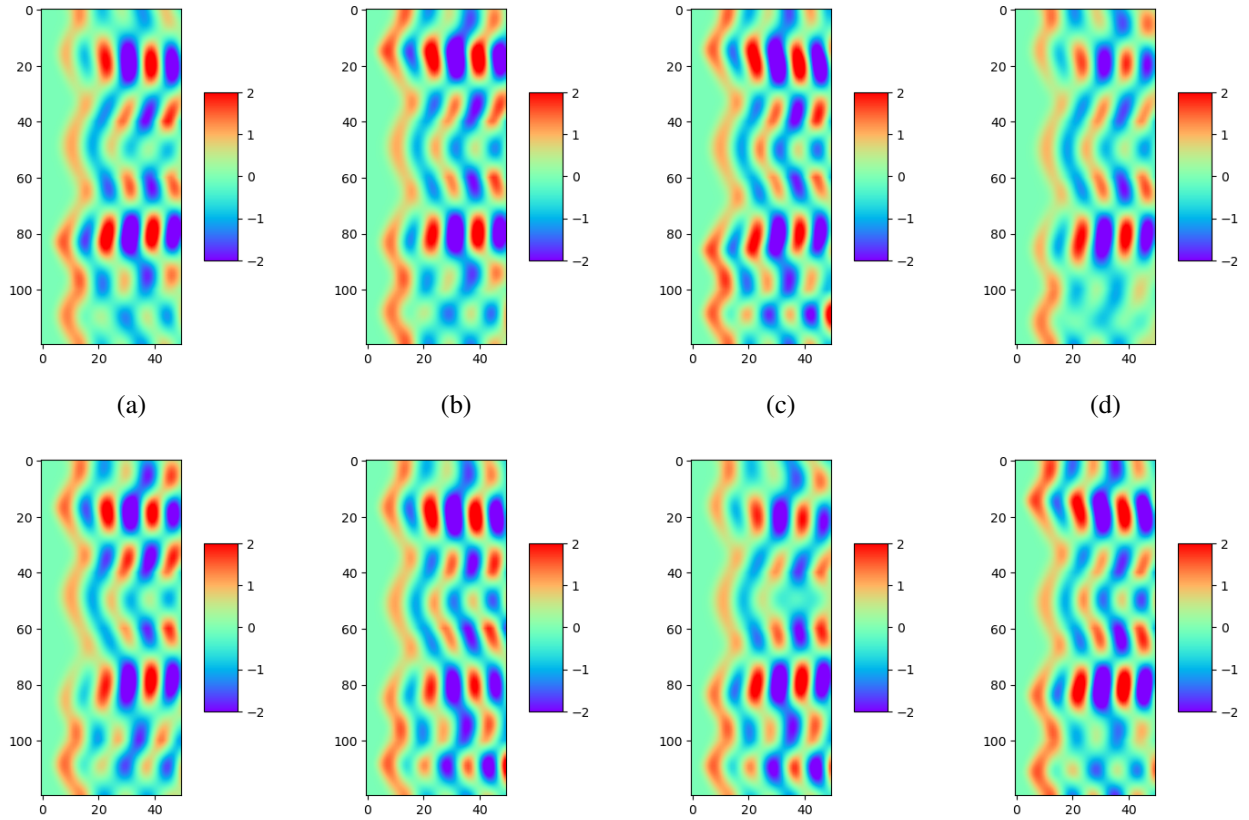
$$\begin{bmatrix} t_1^3 & t_1^2 & t_1 & 1 \\ t_2^3 & t_2^2 & t_2 & 1 \\ 3t_1^2 & 2t_1 & 1 & 0 \\ 3t_2^2 & 2t_2 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} M_u(p,t_1) \\ M_u(p,t_2) \\ M_v(p,t_1) \\ M_v(p,t_2) \end{bmatrix} \tag{5.23}$$

This interpolation approach is similar to Hermit polynomial interpolation which is a generalization of Lagrange polynomial interpolation in the presence of derivative values of the function approached.



(a) Start of simulation (b) During Simulation

Figure 5.7: FEM Output with 4 source points



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.8: Boundary data for different parametric values

Table 5.20: Train error on full displacement

	Basic	BN	E	SL	BN & SL	E & SL
FC t	37.57	21.63	38.88	36.75	13.02	36.71
Conv2D	1.88	1.71	1.76	1.73	4.62	1.76
Conv2D t	4.36	4.29	4.94	5.81	4.46	5.88
Conv3D	4.07	1.11	1.31	1.60	**1.09**	1.57
Conv2.5D	1.72	1.50	1.69	1.83	1.37	1.72
Conv2.5Db	4.24	2.86	2.96	2.44	2.30	1.97

Table 5.21: Train error on full velocity

	Basic	BN	E	SL	BN & SL	E & SL
FC t	9.75	3.32	8.43	8.58	4.29	7.76
Conv2D	2.89	2.96	3.66	2.80	4.44	3.03
Conv2D t	1.18	**1.06**	1.22	1.44	1.18	1.50
Conv3D	148.38	2.45	2.59	2.86	2.56	2.83
Conv2.5D	2.96	2.65	2.75	2.73	2.61	2.78
Conv2.5Db	3.14	2.97	3.38	3.15	2.88	2.85

Table 5.22: Test error on full displacement

	Basic	BN	E	SL	BN & SL	E & SL
FC t	11.85	5.38	13.05	10.12	5.42	9.81
Conv2D	0.67	0.52	0.52	0.55	1.38	0.54
Conv2D t	1.33	1.25	1.55	1.61	1.31	1.84
Conv3D	3.05	**0.34**	0.44	0.54	0.35	0.51
Conv2.5D	0.54	0.46	0.56	0.55	0.44	0.56
Conv2.5Db	1.09	0.81	0.81	0.76	0.69	0.61

Table 5.23: Test error on full velocity

	Basic	BN	E	SL	BN & SL	E & SL
FC t	5.06	1.86	5.10	4.81	2.44	4.81
Conv2D	2.21	2.15	2.25	2.14	2.64	2.18
Conv2D t	0.63	**0.61**	0.70	0.76	0.66	0.82
Conv3D	138.39	2.00	2.08	2.14	2.05	2.13
Conv2.5D	2.18	2.09	2.12	2.09	2.09	2.12
Conv2.5Db	2.29	2.26	2.36	2.25	2.19	2.19

Tables 5.20, 5.21 5.22 and 5.23 show respectively the training error of each model variant for full data prediction considering all the possible regularization methods described in the previous sections. The tables indicate equivalent results to the previous experiments with the exception of networks with high numbers of parameters have very poor training performance and an even poorer generalization error. The tables also shows that temporally global models which have been interpolated achieve better precision than regressive models.

Table 5.24: Train error on displacement after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	29.61	11.52	32.84	23.34	8.42	25.88
Conv2D	0.32	0.21	0.41	0.26	0.31	0.26
Conv2D t	2.35	1.97	2.21	2.71	1.93	2.75
Conv3D	1.14	0.29	0.33	0.35	0.30	0.44
Conv2.5D	0.61	0.58	0.59	0.57	0.46	0.45
Conv2.5Db	0.85	0.78	0.82	0.72	1.01	0.72
FC t Boundary	27.66	9.58	22.90	27.24	9.90	19.77
Conv1D Boundary	0.12	**0.08**	0.11	0.15	0.10	0.10
Conv1D t Boundary	1.27	1.20	1.15	1.97	2.01	1.54
Conv2D Boundary	0.22	0.19	0.24	0.25	0.19	0.22
Conv1.5D Boundary	0.34	0.32	0.25	0.25	0.30	0.26

Table 5.25: Train error on velocity after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	17.62	6.93	18.39	13.90	6.00	14.49
Conv2D	0.31	0.23	0.36	0.29	0.30	0.28
Conv2D t	1.44	1.12	1.30	1.60	1.32	1.73
Conv3D	1.23	0.27	0.28	0.30	0.26	0.33
Conv2.5D	0.54	0.44	0.47	0.44	0.35	0.41
Conv2.5Db	0.61	0.60	0.64	0.60	0.63	0.55
FC t Boundary	13.19	5.81	13.05	12.94	5.86	10.50
Conv1D Boundary	0.15	**0.10**	0.13	0.16	0.13	0.13
Conv1D t Boundary	0.85	0.83	0.77	1.03	1.27	0.94
Conv2D Boundary	0.20	0.18	0.20	0.21	0.17	0.19
Conv1.5D Boundary	0.24	0.25	0.20	0.22	0.23	0.23

Table 5.26: Test error on displacement after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	11.85	5.38	13.05	10.12	5.42	9.81
Conv2D	0.67	0.52	0.52	0.55	1.38	0.54
Conv2D t	1.33	1.25	1.55	1.61	1.31	1.84
Conv3D	3.05	0.34	0.44	0.54	0.35	0.51
Conv2.5D	0.54	0.46	0.56	0.55	0.44	0.56
Conv2.5Db	1.09	0.81	0.81	0.76	0.69	0.61
FC t Boundary	5.92	3.03	7.77	7.83	4.11	6.18
Conv1D Boundary	0.06	**0.04**	0.06	0.07	0.08	0.05
Conv1D t Boundary	0.41	0.44	0.42	0.55	0.65	0.53
Conv2D Boundary	0.11	0.07	0.09	0.10	0.08	0.11
Conv1.5D Boundary	0.12	0.11	0.10	0.11	0.11	0.11

Table 5.27: Test error on velocity after zoom

	Basic	BN	E	SL	BN & SL	E & SL
FC t	10.46	4.35	10.38	8.50	4.55	8.27
Conv2D	0.23	0.18	0.22	0.25	0.20	0.21
Conv2D t	0.90	0.75	0.77	0.96	0.87	1.14
Conv3D	1.75	0.17	0.21	0.22	0.18	0.23
Conv2.5D	0.33	0.27	0.30	0.30	0.24	0.29
Conv2.5Db	0.41	0.41	0.44	0.40	0.39	0.38
FC t Boundary	6.42	3.39	7.64	7.30	3.74	6.33
Conv1D Boundary	0.11	**0.09**	0.11	0.11	0.11	0.11
Conv1D t Boundary	0.48	0.48	0.46	0.58	0.73	0.56
Conv2D Boundary	0.14	0.11	0.13	0.14	0.11	0.14
Conv1.5D Boundary	0.16	0.14	0.14	0.16	0.16	0.15

Tables 5.24, 5.25 5.26 and 5.27 shows respectively the training error of each model variant for full data prediction considering all the possible regularization methods described in the previous sections. The tables indicate that the zoom operation corrects various error from the full data prediction and indication also that the boundary models achieves way better precision than the full data approaches. Euler regularization shows a perceptible amelioration for some models interpolated or regressed over the time grid for the zoom operation.

5.8 Conclusion

In this work, we presented a novel method of compressing convolutional neural networks for FEM physical data and approaches to optimize data from FEM models for CNN training. Our compression approach can also be applied to learning data in higher dimensions since the complexity of the models is linearly dependent on the dimension and actual deep learning code library

only allow up to 3D data learning. After that, we validated our compressed models on physical data derived from a FEM model that was used to solve a 2D wave equation, we combined each approach with different regularization approaches, and showed that our convolutinal compression technique achieves equivalent performance as classical convolutional layers with fewer trainable parameters.

Chapter 6

Uncertainty quantification in impact simulation via conditional Wasserstein Generative Adversarial Network

Abstract

This chapter proposes a new training algorithm for Wasserstein Generative Adversarial Networks (WGAN) to construct a stochastic model of large dynamical systems for uncertainty quantification. The model is trained on data from a finite element method code, with the objective of extracting stochastic boundary conditions for faster finite element predictions on a submodel. The proposed framework is viewed as a randomized and parameterized simulation generator on the submodel, which can be used as a Monte Carlo estimator. The algorithm is applied to an explicit dynamic contact 3D problem to evaluate its stability and precision compared to true solutions of interest, which are the displacement and velocity fields.

Résumé

Ce chapitre propose un nouvel algorithme d'entraînement pour les réseaux de neurones adversaires génératifs de Wasserstein (WGAN) pour construire un modèle stochastique de grands systèmes dynamiques pour la quantification d'incertitudes. Le modèle est entraîné sur des données provenant d'un code de méthode des éléments finis, dans le but d'extraire des conditions aux limites stochastiques pour des prévisions plus rapides sur un sous-modèle. Le cadre proposé est considéré comme un générateur de simulation randomisé et paramétré sur le sous-modèle, qui peut être utilisé comme estimateur de Monte Carlo. L'algorithme est appliqué à un problème de contact dynamique explicite en 3D pour évaluer sa stabilité et sa précision par rapport aux vraies solutions d'intérêt, qui sont les champs de déplacement et de vitesse.

Contents

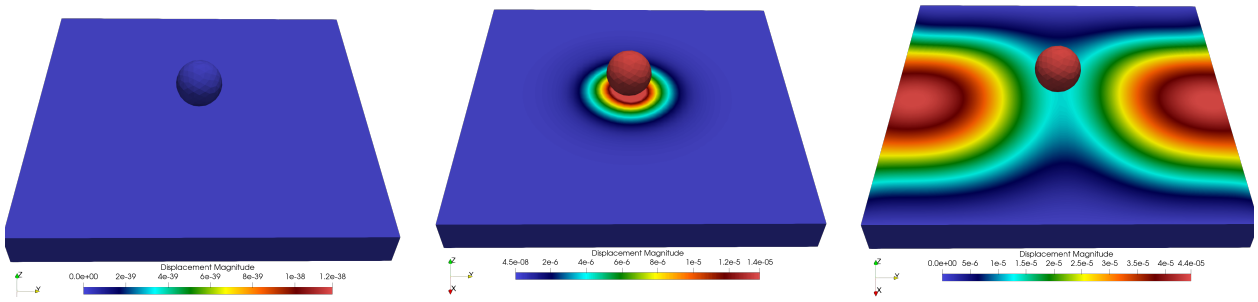
6.1 Introduction . . . 114
6.2 Problem Definition . . . 115
6.2.1 Data processing . . . 117
6.2.2 Data Sampling . . . 120
6.3 Models . . . 120
6.3.1 Deep Convolutional Neural Regressor . . . 121
6.3.2 Generative Adversarial Network . . . 121
6.3.3 Vanilla GAN . . . 121
6.3.4 Wasserstein GAN . . . 121
6.3.5 Gradient Penalty and Spectral Normalization . . . 121
6.3.6 Relativistic Discriminator . . . 122
6.4 Numerical Results . . . 123
6.4.1 Regression - DcNR and Auto Encoder . . . 123
6.4.2 Adversarial Regression - GANs . . . 127
6.4.3 Uncertainty quantification by a conditional GAN . . . 129
6.5 Conclusion . . . 132

6.1 Introduction

In order to analyse non-parametric uncertainties of very large dynamical systems, it is necessary to construct a stochastic model of said system that verifies the same physical properties. Such models are usually built using physics-informed neural networks by increasing precision over physical properties. This is generally obtained using a penalization term given by the residual of the Partial Differential Equation (PDE) as a cost function. However, with Finite Element Method (FEM) models designed for engineering applications, it is quite intrusive to get access to the residual of the PDE. Therefore, to address this issue, we propose a new training algorithm for the Wasserstein Generative Adversarial Networks (WGAN) for the construction of a stochastic model of dynamical systems. We show that this enables efficiency in the training stage and more accurate results on mean quantities obtained in the generative stage. The objective of our approach is to train a GAN on data from a finite element method code so as to extract stochastic boundary conditions for faster finite element predictions on a submodel. The submodel and the training data have both the same geometrical support. It is a zone of interest for uncertainty quantification and we assume that such zone of interest is relevant to engineering purposes. In the exploitation phase, the framework can be viewed as a randomized and parameterized simulation generator on the submodel, which can be used as a Monte Carlo estimator. Finally we apply this algorithm to an explicit dynamic contact 3D problem. The main purpose of this work is to evaluate the stability of the training algorithm proposed and the precision of the Monte Carlo estimator constructed with respect to the true solutions of interest, which are the displacement and velocity fields. The dynamic contact 3D problem considered is formed by a sphere which comes into contact with a plate, a rectangular cuboid, with an initial velocity along the z-axis. Our objective is to study wave

propagation and reflection phenomena on the plate following the shock over a discretized time grid. The sphere is considered formed by uniform elastic and isotropic materials behaving linearly following the Hooke's law, the plate is formed by elasto-plastic and isotropic materials behaving following the Johnson-Cook law. The area of interest considered is the cuboid volume on the plate under the area of contact.

6.2 Problem Definition



(a) Before impact (b) Impact (c) After impact

Figure 6.1: Main steps of the simulation

In this chapter we investigate the reducibility of a contact case in structural dynamics. We consider a sphere of perimeter $R_{cylinder}$ which comes into contact with a plate, a rectangular cuboid of length, height and width $L_{plate}, W_{plate}, H_{plate}$ with an initial velocity v_z along the z-axis. Our objective is to study wave propagation and reflection phenomena on the plate following the shock over a discretized time grid denoted $T = (t_k)_{k \in [\![1\ ;\ n]\!]}$ with a time step Δt. This analysis requires the constructions of models to generate new parameterized simulations with a lower cost than solving the governing physical equations. To satisfy this requirement, we choose to apply data-driven methods such as metamodeling and deep learning models.

In this chapter both epistemic and aleatory uncertainties are taken into account, while the objective is to construct a model capable of learning both uncertainties and offer tools to explore both or any one of them following the user objective, this will be achieved using conditional generative adversarial networks, where the conditioned dimensions will model the aleatory uncertainties and the epistemic uncertainties will be captured by the random dimension of the GAN.

Figure 6.1 shows the main steps of the finite element method simulation. The sphere is considered formed by uniform elastic and isotropic materials behaving linearly following the Hooke's law, which is written in the matrix form as:

$$\boldsymbol{\sigma} = \frac{E}{1+\nu}\left(\varepsilon + \frac{\nu}{1-2\nu}\mathrm{Tr}(\varepsilon)\right), \tag{6.1}$$

The plaque is considered formed by elasto-plastic and isotropic materials behaving following Johnson-Cook law with no temperature effects:

$$\boldsymbol{\sigma} = \left(A_1 + A_2(\varepsilon^p_{eq})^{\lambda_2})\right)\left((1+\lambda_1)\ln(\frac{\dot{\varepsilon}^p_{eq}}{\dot{\varepsilon}^p_{eq,ref}})\right), \tag{6.2}$$

where σ and ε are respectively the second-order stress and strain tensors, E is the Young modulus, ν is the Poisson ratio, and Tr is the trace operator. $A_1, A_2, \lambda_1, \lambda_2$ are values determined experimentally for lead components with the respective values 266, 229, 0.0294 and 0.8, $\dot{\varepsilon}^p_{eq,ref}$ is the reference strain rate. Additionally, we denote the density of the materials by ρ, which plays a part in the response of the plate in such transient dynamical regimes. We introduce a unique parametrization of each simulation results : let $\theta = (E_{Plaque}, \rho_{Plaque}, \nu_{Plaque}, V_z, Sphere_x, Sphere_y)$ a parameters vector that defines a unique finite element simulation denoted $S(\theta)$, S being the vector of the fields values on the grid of the plate. V_z being the initial velocity of the sphere, $Sphere_x, Sphere_y$ being the coordinate of the center of the sphere in the initial plane. Materials of the sphere are fixed as $E_{Sphere} = 2e^9, \rho_{Sphere} = 7800, \nu_{Sphere} = 0.03$. Physical parameters are necessary for the submodel of the area of interest, thus they will be conditioned into our models to represent the aleatory uncertainty, the velocity and coordinates of the sphere are the parameters associated with the epistemic uncertainty. The objective here is to construct generative models which function is to map the parameters vector θ to the simulation vector $S(\theta)$. As physical fields, we consider displacement of the elements of the plate on the z-axis (u_z).

Since we consider only the case of frictionless, adhesive-free normal contact, and considering a space discretization of both solids relying on the Finite Element discretization, the dynamic contact problem can be written as (see Balajewicz et al. [2015]; Fauque et al. [2018]) :

$$\begin{cases} M\ddot{u} + Ku = B^T\lambda \\ Bu - c \leq 0 \\ \lambda(Bu - c) = 0 \\ \lambda \geq 0 \\ u(0) = u_0 \\ \dot{u}(0) = \dot{u}_0. \end{cases} \tag{6.3}$$

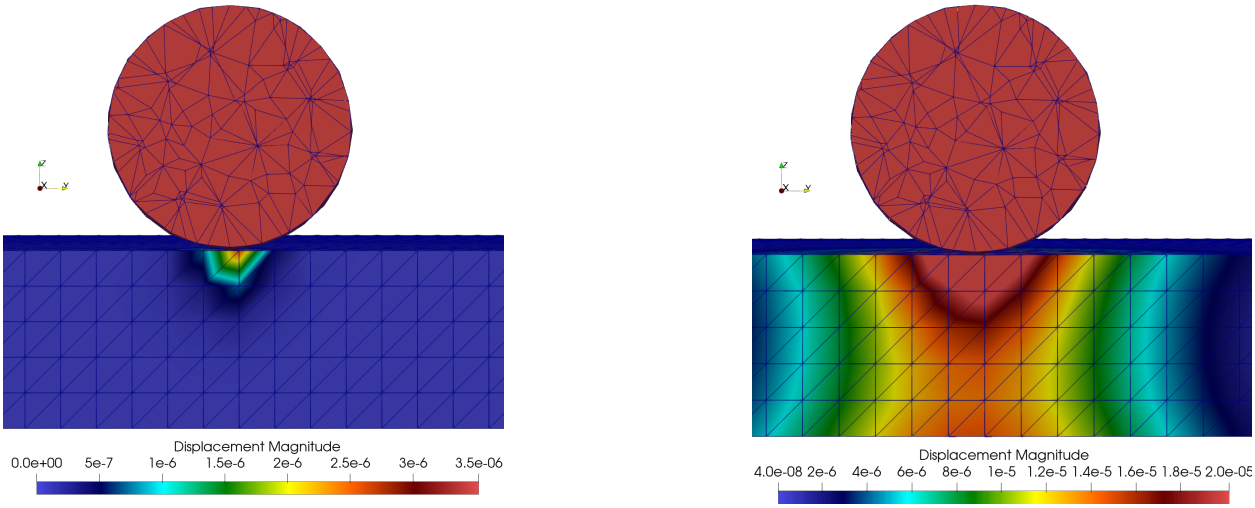
Where a dot designates a time derivative, the first equation expresses the dynamic equilibrium of the two solids, the inequality constraint derives from the space discretization of the Hertz-Signorini-Moreau contact conditions which are enforced by introducing dual Lagrange multipliers. u is a space discretized displacement vector of the plaque and the cylinder degrees of liberty, M and K are respectively the block diagonal mass and stiffness matrices for the two solids. B is a signed boolean matrix which extracts from u the pair of dof governed by a contact condition, c is the vector of initial clearances, and λ is the vector of discretized Lagrange multipliers.

Then an explicit time discretization is performed in the dynamic case, finally solving the dynamic problem can be written as:

$$\forall n \geq 2, (u_n, \lambda_n) = \arg\min_{v,\mu} v^T\left(\frac{1}{\Delta t^2}M + K\right)v + \frac{1}{\Delta t^2}v^T M(-2u_{n-1} + u_{n-2}) - \mu^T(Bv - c) \tag{6.4}$$

Where the superscript n designates the n-th time-step and Δt designates the difference between two consecutive time-steps.

Both solids are discretized in finite element meshes using 3D tetrahedric elements, 8546 nodes combined between the two solids, around 5000 nodes for the plaque mesh.



(a) t_1 (b) t_{10}

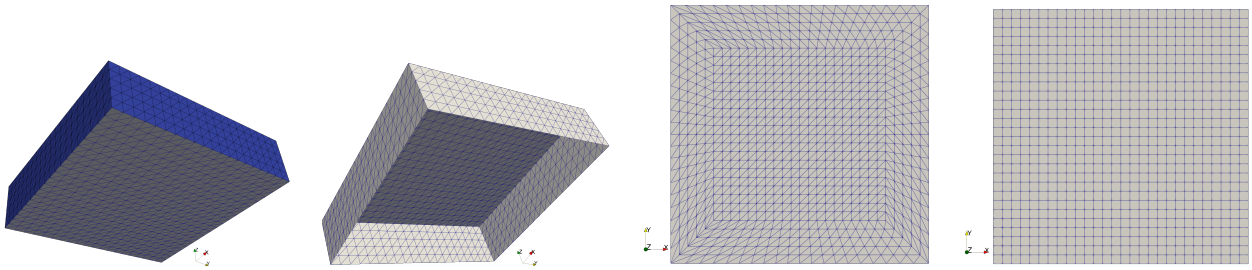
Figure 6.2: Cut-away view of the impact zone

6.2.1 Data processing

This chapter develops a method simular to the previous chapters relying on FEM data to construct generative models. This technique relies on the same concept, which is a submodel comprised of two different parts. A neural network that is trained to learn boundary conditions around a preset zone of interest, and a finite element submodel that is constructed to use the boundary conditions that were created by the neural network inside the zone of interest. We make the assumption that there are no modeling error in the area of interest that the proposed submodel covers. We choose as a region of interest a cuboid containing the zone of the impact of the sphere on the plaque.

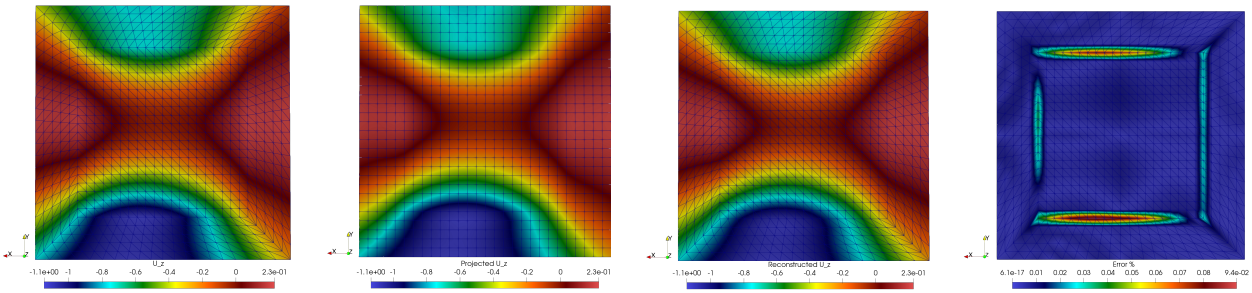
To carry out the projection phase of the data on a Euclidean 2D grid, which is necessary to ensure that our simulation data can be processed by convolutional neural networks, we need to map the 3D boundary grid of the 5 faces of the cuboid (see 6.3), to a 2D grid. The reversibility of this operation is maintained by carrying an unique identifier for each node of the grid. The different grid concerning this operation can be seen on figure 6.3 and results of data projection and reconstruction on these grid on figures 6.4 and 6.5. Even though this method is only adapted for this problem or for a similar grid, one can use generalized methods to construct the 2D representative grid as Multi Dimensional Borg and Groenen [2005] Scaling and Mesh Parametrization Sheffer et al. [2007].

Then, utilizing the finite element basis functions, we construct the projection operator from the initial mesh onto the 2D grid, in addition to the inverse projection operator. This allows us to ensure that our data can be processed by convolutional neural networks. The next step, which follows the training of our models and the computing of test results, involves performing an inverse projection operation on the initial grid. After the projection procedure has been completed, the physical data are normalized by having the mean value removed and being scaled to the unit variance. This is done with the use of a standard affine scaler.



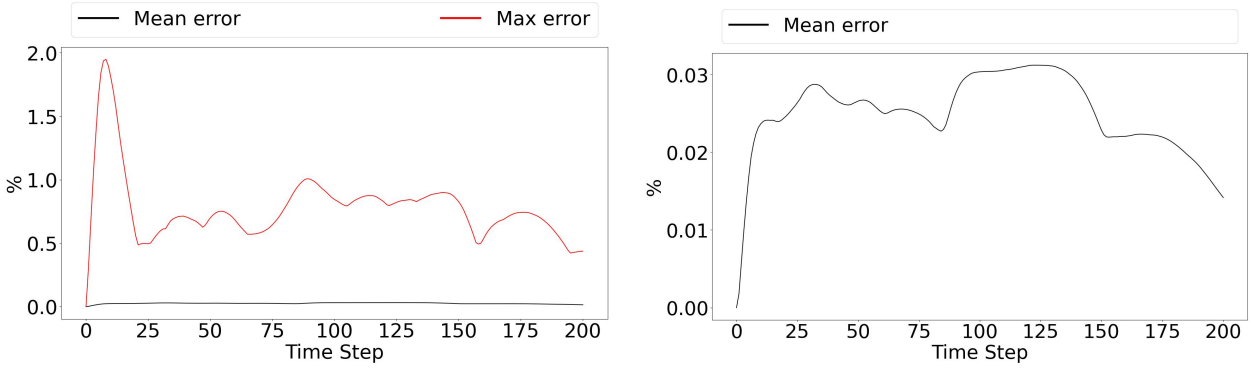
(a) Zoom Mesh (b) Boundary Mesh (c) Unfolded Boundary Mesh (d) Cartesian Grid for Convolution

Figure 6.3: Processing of the boundary of the area of interest



(a) Field On the Unfolded Mesh (b) Projected Field (c) Inverse Projection Field (d) Reconstruction Error

Figure 6.4: Different steps of projection and reconstruction error



(a) Maximum of reconstruction error (b) Mean of reconstruction error

Figure 6.5: Reconstruction error of the projection phase

Since our approach requires the extraction and the processing of boundary conditions in the following we present the approach adopted for our boundary data. For each parametric simulation of the impact we extract the boundary values as a 2D projected matrix in a specific order to maintain physical local information for the convolutional layers as shown in Figure 6.6. Learning boundary data instead of the whole spatial information reduces the learning problem dimensionality in addition to the fact that the submodeling approach maintains physical properties in the area of interest.

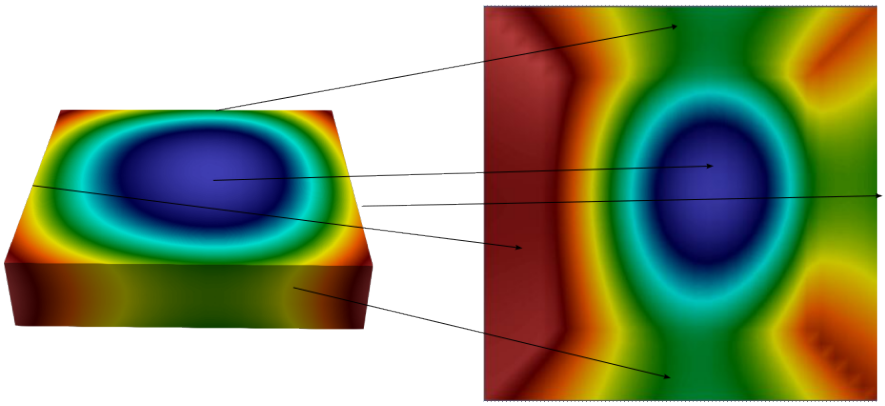


Figure 6.6: Extraction of boundary data

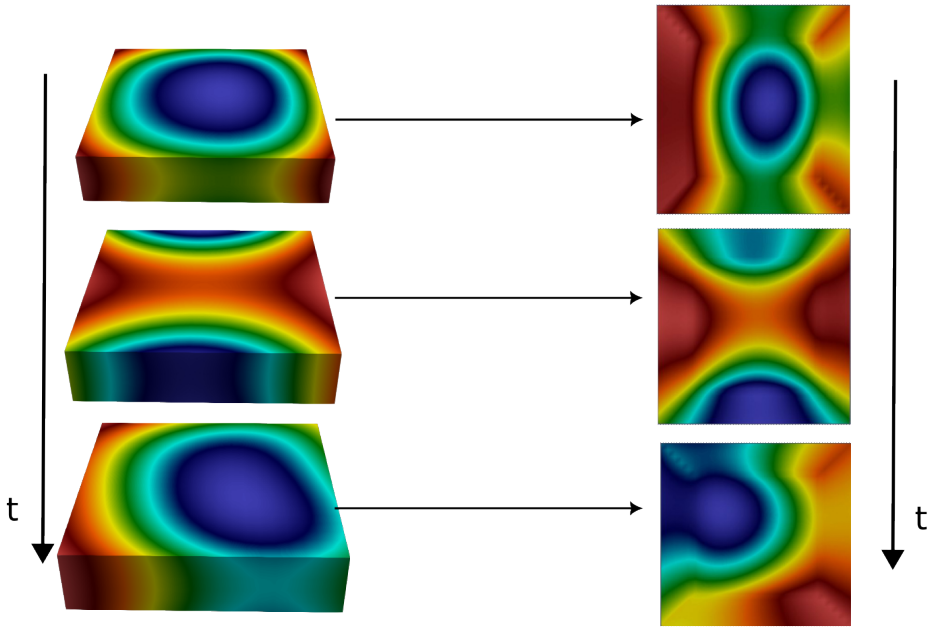


Figure 6.7: Extraction of temporal boundary data

Thus for each time step for a fixed parametric value, we can extract a boundary matrix. All

boundary matrix for a particular parametric value θ can then be aggregated in one 3D tensor in which the spatial matrices are stored in accordance with temporal evolution of values as shown in Figure 6.7. Then, the boundary data for the whole simulation can be viewed as a 3D Video by convolutional layers.

6.2.2 Data Sampling

In this section we present the data range used for sampling and generating data for the training and testing phase. Values were chosen as: $L_{plate} = 8m$, $W_{plate} = 4m$, $H_{plate} = 0.5m$, $R_{Sphere} = 0.1m$, $N_T = 100$ and $\Delta t = 4 \times 10^{-2} s$.

The variable parameters identified in Section 6.2 are sampled following Table 6.1 values with a latin hypercube sampling framework. We sample $N_{Training} = 100$ values for the training data set and $N_{Testing} = 25$ values for the testing data set from the same uniform distibution.

Table 6.1: Parameters sampling

P	Mean Value	Variation (%)	Min Value	Max Value
Young modulus E	2.2×10^{11} MPa	20%	1.7×10^{11} MPa	2.64×10^{11} MPa
Densityν	$7800 kg/m^3$	20%	$6240 kg/m^3$	$9360 kg/m^3$
Poisson's ratio ρ	0.3	20%	0.24	0.35
V_z	-10	20%	-12	-8
$Sphere_x$	0	4% of H_{plate}	-0.01	0.01
$Sphere_y$	0	4% of H_{plate}	-0.01	0.01

We define Ω as the discretized space of the plate. As a quantity of interest, we choose to train our models to predict displacement on the z-axis for the plate, to visualy assess of the performance of our model we fix two time steps $t_1 = 0.01s$ and $t_2 = 0.03s$, two parameters vectors $\theta_1 = \left(1.75 \times 10^{11}, 7.61 \times 10^3, 2.44 \times 10^{-1}, -11.77, -3.38 \times 10^{-3}, -5.11 \times 10^{-4}\right)$ and $\theta_2 = \left(1.81 \times 10^{11}, 7.36 \times 10^3, 2.9 \times 10^{-1}, -9.1, -6.88 \times 10^{-3}, -3.05 \times 10^{-3}\right)$, and two surfaces u_{z1}, u_{z2} respectively the upper surface getting the impact and the free lower surface as surfaces to asses the precision of our models to predict values of interest.

We define a relative error indicator in order to quantify the precision of our metamodels, noted η. For a metamodel M, a parameter vector θ, a space point $p = (x, y, z)$, and a time value t:

$$\eta_p(M, \theta, p, t) = \frac{|M(\theta, p, t) - S(\theta, p, t)|}{\sqrt{\mathop{\mathbb{E}}_{p \in \Omega} |S(\theta, p, t)|^2}}. \tag{6.5}$$

Then we define the global error indicator to compare models:

$$\eta(M, \theta, t) = \mathop{\mathbb{E}}_{p \in \Omega} [\eta_p(M, \theta, p, t)]. \tag{6.6}$$

6.3 Models

In this section we present the different model developed for learning boundary data.

6.3.1 Deep Convolutional Neural Regressor

To assess of the precision of our submodeling approach we first solve a classical regression problem using a Deep convolutional Neural Regressor (*DcNR*) which consists in learning to generate the physical data (S) over a grid with the parameters vector (p) as an input. As indicated by its name, the internal structure of this network is formed by a succession of transposed convolutional layers of adequate dimensions in order to obtain a regression model of the physical field in the adequate size. The objective function in this case being the mean squared error:

$$\min_{\theta} \frac{1}{|T| \times |\mathbb{P}_{Train}|} \sum_{p \in \mathbb{P}_{Train}} \sum_{t \in T} ||S(p)(t) - N(p)(\theta, t)||_2^2 \tag{6.7}$$

Where N denotes the neural network, θ its trainable weights, $\mathbb{P}_{Train}$ the training set of parameters vectors, T the set of time steps and $|.|$ the cardinal of each space. Some of the regularization and decomposition techniques presented in the previous chapter are used in this experiment, same prefixes will be used.

6.3.2 Generative Adversarial Network

6.3.3 Vanilla GAN

To learn the induced epistemic and aleatory uncertainties presents in the data we use a conditional GAN, where the parameters responsible for aleatory uncertainty are being conditioned and the remaining uncertainty induced by unknown parameters is being modeled by the random dimension of the GAN. Let's consider first classical GANs, introduced in Goodfellow et al. [2014] as an unsupervised framework to learn probabilistic densities over data. It relies on the adversarial training of two neural networks a generator and a discriminator. The cost function in this paradigm called Standard GAN or vanilla GAN is:

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x \sim \mathbb{P}_r}[Log(D(x))] + \mathbb{E}_{z \sim \mathbb{P}_z}[Log(1 - D(G(z)))] \tag{6.8}$$

Models using this cost functions will be suffixed by "*_S*"

6.3.4 Wasserstein GAN

In the other hand, in an IPM-Based GAN using the Wasserstein Distance and using the discriminator as the 1-Lipschitz function to compute the Wasserstein distance in an objective function defined as follows:

$$\min_{\theta_{gen}} \max_{\theta_{disc}} \mathop{\mathbb{E}}_{z \sim \mathcal{N}(0,1)} [D(\theta_{disc}, G(\theta_{gen}, z))] - \mathop{\mathbb{E}}_{p \in \mathbb{P}_{Train}} [D(\theta_{disc}, U(p))] \tag{6.9}$$

The principle of GANs can be extended to a conditioned model Mirza and Osindero [2014] if the generator and the discriminator are both conditioned by additional information y. In the generator, noise and information are combined, while in the discriminator, noise and information are considered as 2 network inputs.

6.3.5 Gradient Penalty and Spectral Normalization

The authors in Gulrajani et al. [2017] propose a way to ensure the Lipschitz constraint on the critic, taking into consideration directly restricting the gradient norm of the critic's output with

regard to its input, since a differentiable function is said to be 1-Lipschtiz if and only if it has gradients with norms of at most 1 everywhere. A a soft form of the constraint was implemented in order to get around tractability concerns, and we do this by imposing a penalty on the gradient norm for both real and fake samples. Thus the algorithm 2 to train a Wasserstein GAN with gradient penalty. Although this method solves instability issues and propose a better method to fulfill the Lipschitz constraint it is computationaly costly. In Miyato et al. [2018] the authors suggest an innovative weight normalization approach known as spectral normalization in order to make the training of discriminator networks more consistent and thus enforce the Lipschitz constraint without computing any additional gradient, reducing the computational cost of discriminator training. The Lipschitz constant of the discriminator function f is controlled via spectral normalization, which is achieved by restricting the spectral norm of each layer. Power iteration method Golub and Van der Vorst [2000] is used to estimate the Spectral Norm at each optimization step of the Critic in a GANs optimization Setup, thus fulfilling the 1-Lipschitz condition with less computational cost than gradient penalty. Nonetheless this regularization technique is also useful to stabilize training in Standard GAN, since it provides regularity in the Discriminator. Models trained by Gradient Penalty will be denoted by "*GP*" and using spectral normalization will be denoted by "*SN*".

6.3.6 Relativistic Discriminator

The intuition that the likelihood that real data are real $D(x_r)$ should decline as the likelihood that fake data are real $(D(G(z))$ increases, is the main argument for this approach. This is the main missing attribute of SGAN. The authors (Jolicoeur-Martineau [2019]) proposed $D(x) = \sigma(C(x))$ as a way to define the discriminator in a typical GAN in terms of the transformed layer $C(x)$. A straightforward method for making the discriminator relativistic (i.e., having the output of D depend on both real and false data) is to sample from real/fake data pairs $x = (x_r, G(z))$ and define $D(x) = \sigma(C(x_r) - C(G(z))$.

This alteration may be interpreted as follows: The discriminator evaluates the likelihood that the provided real data is more realistic than a randomly picked set of fake data. In a similar fashion, we may define $D_{rev}(x) = \sigma(C(G(z)) - C(x_r))$ as the probability that the provided fake data is more realistic than randomly picked real data, for an Average Relativistic Discriminator. Models having a relativistic discriminator will be suffixed by "*_Rel*" and for average relativistic discriminator the suffixe "*_RelA*" will be used.

Fréchet Inception Distance

Images generated by a generative model, such as a generative adversarial network, may be evaluated using the Fréchet inception distance (FID) Heusel et al. [2017], the FID compares your generated data to a real-world dataset, the training data set. This score relies on the Fréchet Distance between two multidimensional gaussian distribution p_1, p_2 where $p_i \sim \mathcal{N}(\mu_i, \Sigma_i)$, defined as follow:

$$d_F(p_1, p_2) = ||\mu_1 - \mu_2||_2^2 + Tr(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{\frac{1}{2}}) \tag{6.10}$$

Inception score is then defined as the Fréchet distance between the output of the last convolutional layer of the Inception Model Szegedy et al. [2015] trained on image classification for the generated data with the generator and also the training data. The Gaussian is the maximum entropy distribution for given mean and covariance, therefore we assume the coding units of Inception Model for unseen data to follow a multidimensional Gaussian distribution. Let f be the Inception model stripped from his last activation layer, the Fréchet Inception Score of a generative model G trained on a data set $(x_i)_{1 \leq i \leq n}$ is defined as:

$$F_{score}(G) = d_F(f(x), f(G(z)) \tag{6.11}$$

Where $(z_j)_{1\leq j\leq n_s}$ is a set of random variable to sample data from the generator, n_s to be determined empirically. Although this score is adequate to assess generative capabilities of GAN models trained on image generation, it is not suited for physical data, in Preuer et al. [2018] the authors propose a method to adapt the Fréchet Inception Score for chemical data by swapping the Inception model by a Chemb1Net model which was trained on molecular data. In Obukhov and Krasnyanskiy [2020] authors propose methods to adapt FID for generic Generative Adversarial Network by swapping Inception model with an adequate classifier depending on the distribution being learnt. In this chapter we propose an alternative to the inception model to compute the FID. We train multiple Auto Encoder on our data and we use the latent space as a feature extractor for computing the FID, every auto encoder model will be prefixed by "AE_".

The computation of the FID is an a posteriori approach to choose an appropriate GAN or appropriate samples of the latent space of a GAN for which the generative phase of the generator respects at best the distibution of the real physical data. In what follows, we only use the FID to choose between several types of GANs (SGAN, WGAN, etc...) but not to choose subsamples from the random latent space based on the FID computation. This intelligent subsampling may be a prospect to this work in order to improve furthermore the generation mode of the generator. In Akkari et al. [2021] the authors proposed a new cyclic architecture of the Variational AutoEncoder (VAE) in order to improve the compression of the inputs training data within the Gaussian latent space of the VAE. This cyclic architecture provides furthermore a practical criteria for stopping the learning phase of a variational bayesian method which is in general a hard task. The VAE trained by the cyclic architecture on reacting fluid flow instantaneous velocities and pressures, provided in the generation phase velocity fields that respect the mass conservation principle, which gives some insights to adapt the same paradigm to obtain physical stopping criteria for GANs training and physical quality assessment of the Generator's samples.

6.4 Numerical Results

All models were trained using Kingma and Ba [2014] optimizer and a learning rate of 0.0001, 1000 epoch for Regression models, and 10000 epochs for GAN models. In the following, all figures show results on the area of interest. In all tables, results in bold black indicate best results, results in blue indicate results within an acceptable threshold, and results in red indicate unacceptable results.

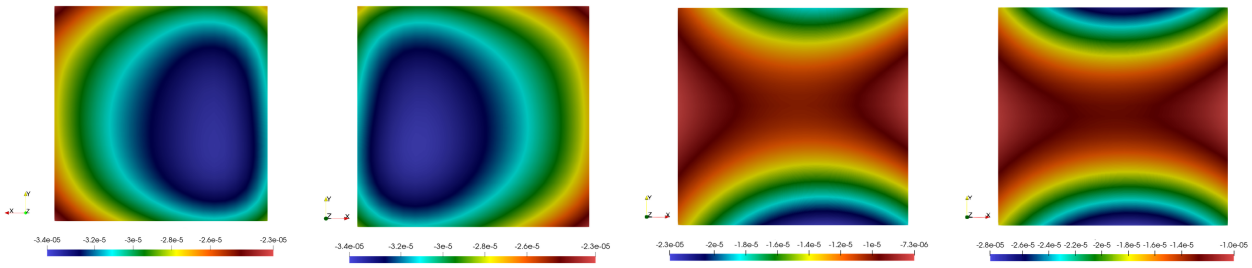
6.4.1 Regression - DcNR and Auto Encoder

Table 6.2: Train error on boundary data

	Basic	BN
Conv2D t	1.63	1.19
Conv3D	2.15	**0.82**
Conv2.5D	1.71	1.07
Conv2.5Db	14.91	6.04

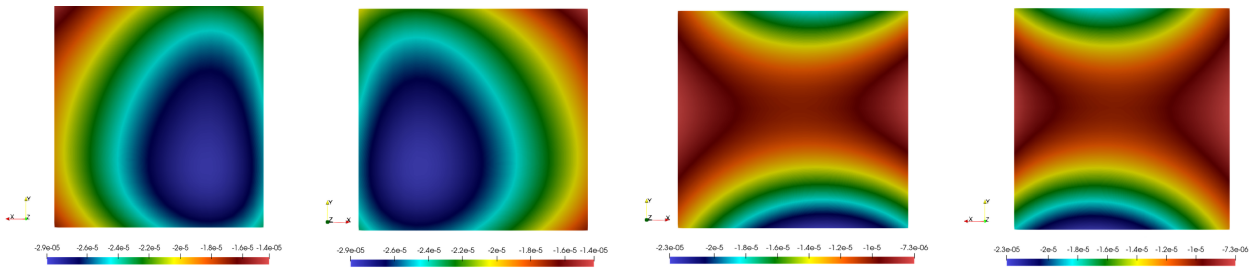
Table 6.3: Test error on boundary data

	Basic	BN
Conv2D t	2.99	2.73
Conv3D	3.63	2.34
Conv2.5D	2.51	**2.27**
Conv2.5Db	16.91	6.68



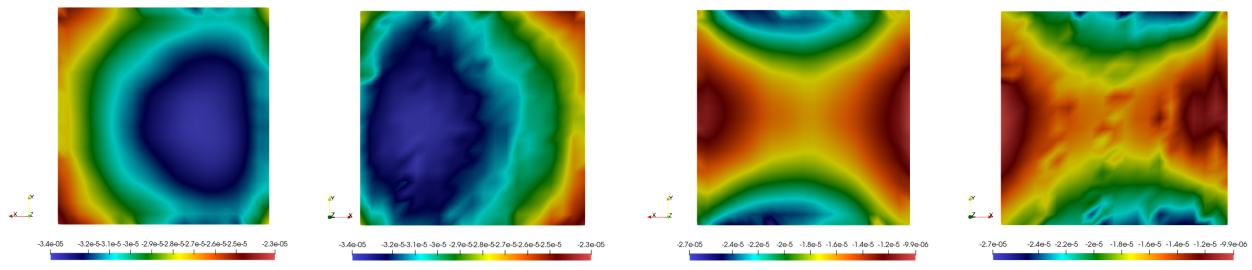
(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.8: Real values displacement : train example



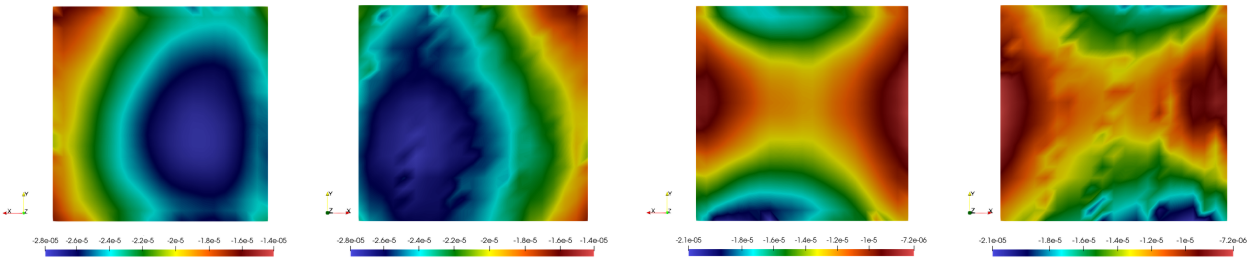
(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.9: Real values displacement : test example



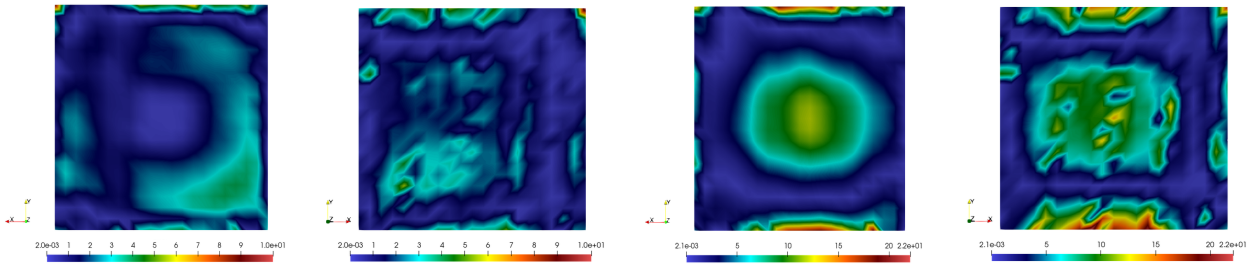
(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.10: Regression prediction : train



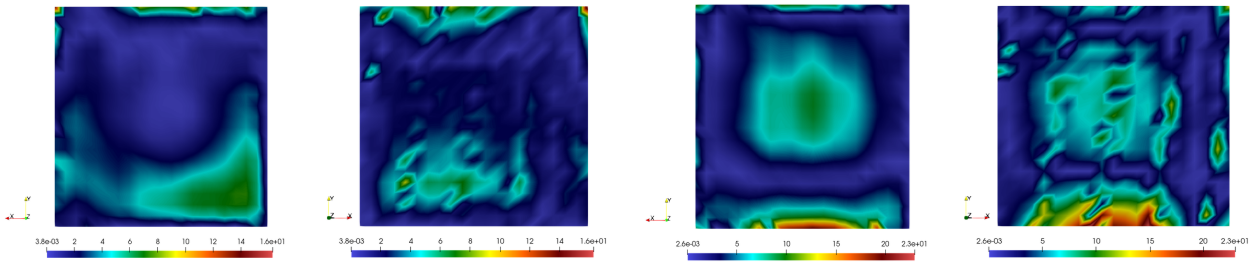
(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.11: Regression prediction : test



(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.12: Regression error : train



(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.13: Regression error : test

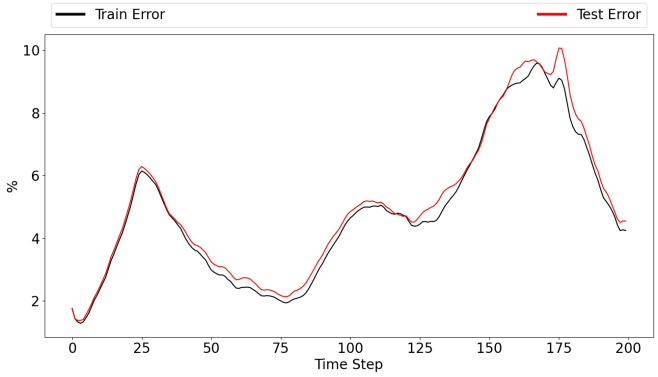


Figure 6.14: Regression error

Tables 6.2 and 6.3 show the time averaged errors of each of the trained models, show also that the model that achieved the best bias-variance trade off is the decomposed model *Conv*2.5*D*, which we use to visualize field prediction and errors in figures 6.10, 6.9, 6.11 and 6.13. These figures show that the zoom operation hold the regularization property over the error, since prediction error from the neural networks on the boundary is damped and not propagated by the FEM Zoom model inside the area of interest. Figure 6.14 as well as the figure cited before, shows that the model holds an acceptable generalization error considering its training error.

Table 6.4: Train reconstruction error

	Basic	BN
Conv2D t	1.06	**0.42**
Conv3D	3.81	1.18
Conv2.5D	18.81	18.76
Conv2.5Db	19.77	19.04

Table 6.5: Test reconstruction error

	Basic	BN
Conv2D t	1.80	**1.19**
Conv3D	8.18	5.12
Conv2.5D	18.02	17.96
Conv2.5Db	19.05	18.29

Tables 6.4 and 6.5 show the time averaged reconstructions errors of each of the trained auto encoder models, they also show that the model that achieved the best bias-variance trade off is the model *Conv*2*D_t* which we use in computing the FID for every GAN model in the following section.

6.4.2 Adversarial Regression - GANs

A proposal for doing high-dimensional non-linear regression together with uncertainty estimates is called adversarial regression. We were able to acquire an estimate of the whole predictive distribution for a new observation by making use of a Conditional Generative Adversarial Network conditioned by the full parameters vector, and considering a prediction is the mean over 20 samples of the GAN.

Table 6.6: Train error on boundary data

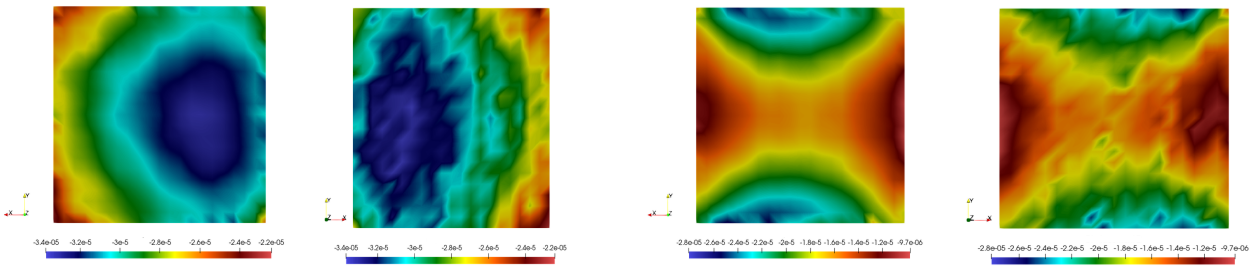
	GP	SN	GP Rel	SN Rel	GP RelA	SN RelA
W Conv3D	**3.33**	98.24	2798.90	1294.64	786.27	699.39
W Conv2.5D	124.14	7.43	6265.38	27072.76	2549.51	7174.98
W Conv2.5Db	15.53	41.68	2200.40	2522.38	57.71	5865.62
S Conv3D	36.30	36.30	36.30	36.30	36.30	36.30
S Conv2.5D	38.40	38.40	38.40	38.40	38.40	38.40
S Conv2.5Db	37.72	37.72	37.72	37.72	37.72	37.72

Table 6.7: Test error on boundary data

	GP	SN	GP Rel	SN Rel	GP RelA	SN RelA
W Conv3D	**4.67**	98.97	2798.85	1308.06	864.86	694.23
W Conv2.5D	124.01	8.97	6282.58	27535.28	2552.15	7172.90
W Conv2.5Db	16.26	41.76	2213.97	2538.36	56.58	5913.20
S Conv3D	36.35	36.35	36.35	36.35	36.35	36.35
S Conv2.5D	38.55	38.55	38.55	38.55	38.55	38.55
S Conv2.5Db	37.82	37.82	37.82	37.82	37.82	37.82

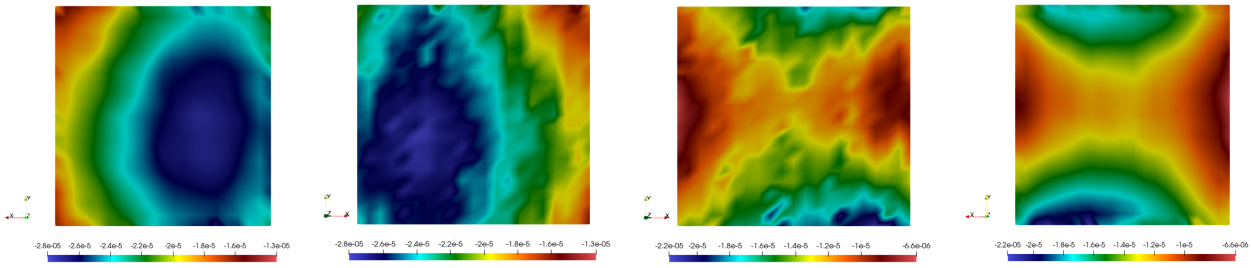
Table 6.8: FID

	GP	SN	GP Rel	SN Rel	GP RelA	SN RelA
W Conv3D	**2.97**	40.42	45.69	41.74	40.49	50.61
W Conv2.5D	51.42	9.06	42.21	42.54	41.77	48.95
W Conv2.5Db	11.79	53.07	40.69	43.14	53.43	38.21
S Conv3D	63.12	63.12	63.12	63.12	63.12	63.12
S Conv2.5D	63.12	63.12	63.12	63.12	63.12	63.12
S Conv2.5Db	53.96	53.96	53.96	53.96	53.96	53.96



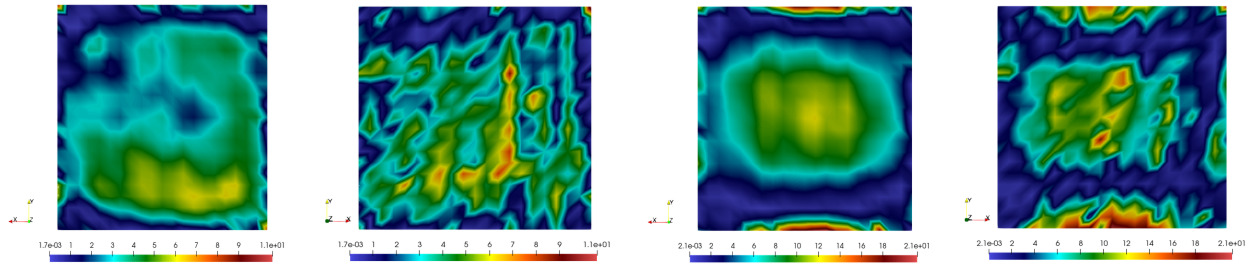
(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.15: Adversarial Regression prediction : train



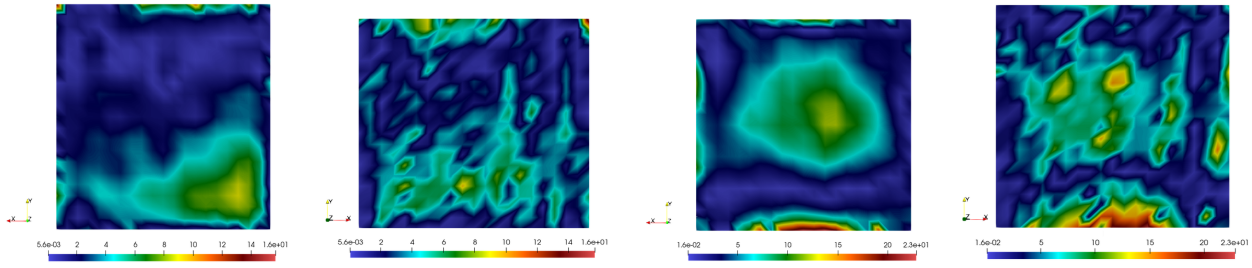
(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.16: Adversarial Regression prediction : test



(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.17: Adversarial Regression error : train



(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.18: Adversarial Regression error : test

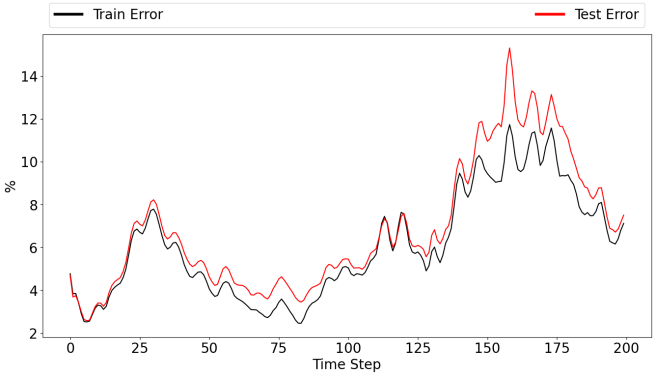


Figure 6.19: Adversarial regression error

Table 6.8 shows that the model that achieves the best FID (the lower the better since its a mesure of dissimilarity with real data) is a Wasserstein GAN relying on gradient Penalty and *Conv*2.5*D* architecture, thus having equivalent results to the tables 6.6 and 6.7 which show that this model achieves the best training and testing errors. In addition it is the only model that achieves acceptable scores, since most other models suffered from mode collapse during training, despite the fact of using different regularization methods as Spectral Normalization and Relativistic discriminator. The tables shows that Spectral Normalization does not suffice to enforce the Lipschitz constraint on the Discriminator since the same model where we used SN instead of gradient penalty achieves poor scores. Models with Standard cost functions suffered the most from mode collapse despite the use of gradient penalty and or Relativistic discriminator, the use of the latter in Wasserstein GAN with Gradient penalty seems to worsen the results even more. We use the WGAN-GP model with a *Conv*2.5*D* architecture to visualize field prediction and errors in figures 6.15, 6.16, 6.17 and 6.18. These figures show that the zoom operation hold the regularization property over the error, since prediction error from the neural nets on the boundary is damped and not propagated by the FEM Zoom model inside the area of interest. Figure 6.19 as well as the figure cited before, shows that the model holds an acceptable generalization error considering its training error, an error superior of the CNN generalization error, which can be explained by the random input of the generator, thus it can be used as a parameterized uncertainty propagation tool for full parametric values.

6.4.3 Uncertainty quantification by a conditional GAN

Table 6.9: Train error on boundary data

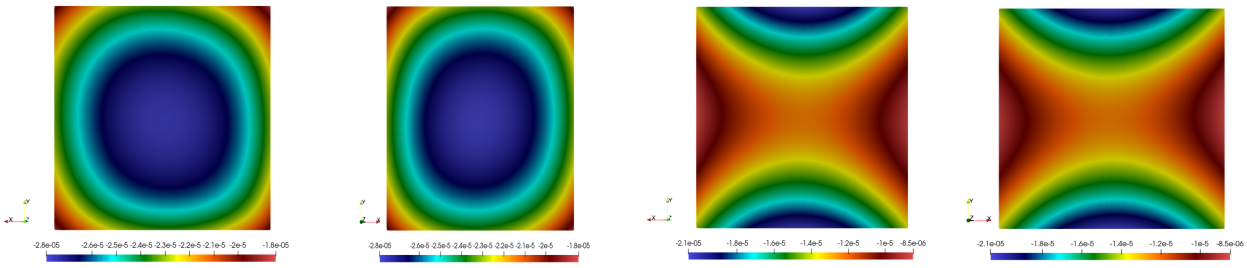
	GP	SN	GP Rel	SN Rel	GP RelA	SN RelA
W Conv3D	**4.60**	66.01	452.31	258.70	142.19	176.07
W Conv2.5D	172.21	34.96	411.73	670.92	2691.18	1972.43
W Conv2.5Db	21.78	74.79	97.61	1621.86	58.24	162.57
S Conv3D	41.66	41.66	41.66	41.66	41.66	41.66
S Conv2.5D	46.42	46.42	46.42	46.42	46.42	46.42
S Conv2.5Db	48.78	48.78	48.78	48.78	48.78	48.78

Table 6.10: Test error on boundary data

	GP	SN	GP Rel	SN Rel	GP RelA	SN RelA
W Conv3D	**5.20**	59.60	811.07	483.07	75.23	210.59
W Conv2.5D	160.90	34.06	369.64	1682.36	3762.59	1942.01
W Conv2.5Db	21.05	72.71	101.29	2013.95	57.66	158.27
S Conv3D	41.39	41.39	41.39	41.39	41.39	41.39
S Conv2.5D	45.77	45.77	45.77	45.77	45.77	45.77
S Conv2.5Db	48.19	48.19	48.19	48.19	48.19	48.19

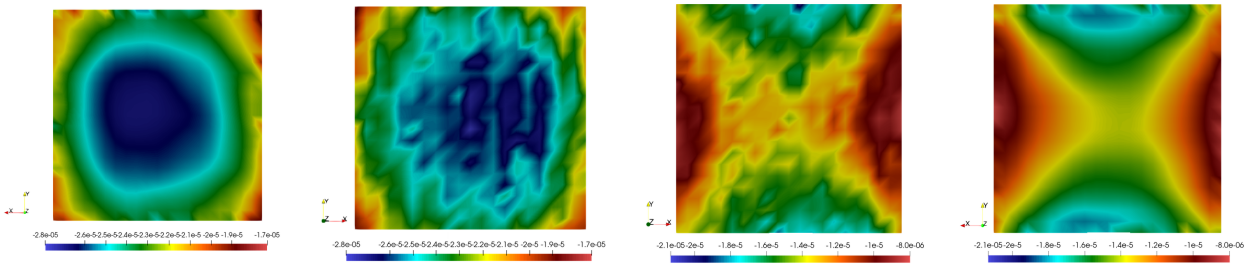
Table 6.11: FID

	GP	SN	GP Rel	SN Rel	GP RelA	SN RelA
W Conv3D	18.78	42.42	43.83	42.42	40.71	45.41
W Conv2.5D	54.81	46.00	46.00	51.62	41.51	44.41
W Conv2.5Db	**14.36**	58.62	44.74	44.25	46.77	42.66
S Conv3D	63.12	63.12	63.12	63.12	63.12	63.12
S Conv2.5D	53.96	53.96	53.96	53.96	53.96	53.96
S Conv2.5Db	63.12	63.12	63.12	63.12	63.12	63.12



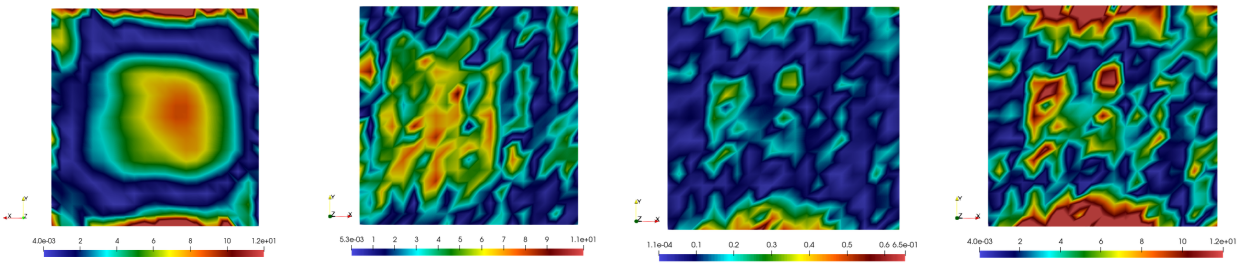
(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.20: Mean over training data



(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.21: GAN mean prediction



(a) (t_1, uz_1) (b) (t_1, uz_2) (c) (t_2, uz_1) (d) (t_2, uz_2)

Figure 6.22: GAN mean error

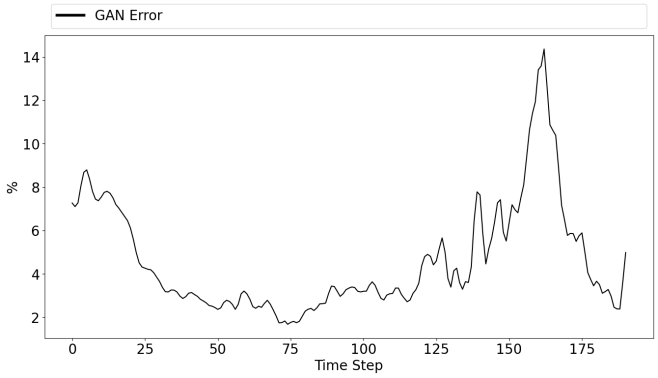


Figure 6.23: GAN mean error

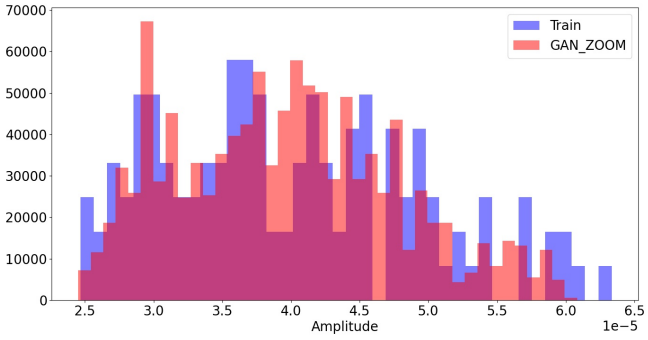


Figure 6.24: Maximum of amplitude histogram

Table 6.11 shows that the model that achieves the best FID is a Wasserstein GAN relying on gradient Penalty and *Conv*2_5*Db* architecture, thus having equivalent results to the tables 6.9 and 6.10 which show that this model achieves the best training and testing errors. Also mostly equivalent to results from adversarial regression where the same variant achieved the best scores but with a different reduced architecture. In addition it is the only model that achieves acceptable scores, since most other models suffered, again, from mode collapse during training, despite the fact of using different regularization methods as Spectral Normalization and Relativistic discriminator. The

tables demonstrate that Spectral Normalization is insufficient to fulfill its function of enforcing the Lipschitz constraint on the Discriminator. This is because the scores obtained by the same model when gradient penalty was substituted for SN were unsatisfactory. In spite of the application of gradient penalty and/or relativistic discriminator, models that used standard cost functions were the ones that suffered the most from mode collapse. Using relativistic discriminator in conjunction with gradient penalty in Wasserstein GAN appears to make the results even worse. We use the WGAN-GP model with a *Conv*2.5*Db* architecture to visualize statistical field prediction and histogram plots, we choose to predict the maximum of amplitude of the wave propagated following the axis z. In figure 6.21, one can see that GAN achieve an acceptable error for estimating the mean of the training data. The Figure shows that the zoom operation of the submodel helds the same correction properties over the noise and errors introduced by the use of convolutional layers and also noise introduced by the GAN's random component. Figures 6.24 shows that our approach hold generative properties that are useful for uncertainty quantification, although not exploring exactly the same probabilistic space, behavior that should be explained by either a not converged training or the effects of the low dimension size of the random vectors in GANs input (z) taken in training, 3 in this work. This figure also shows that the GAN plus physical submodel approach still holds acceptable generative properties that alloy the user to compute statistics over value of interest using Monte Carlo methods which are not feasible with the high fidelity model. In addition, our approach shows generative capacity exploring extremum values that have not been considered in the training set as shown in Figure 6.24 where our approach has good generative capacity on one of the density tails.

6.5 Conclusion

In this chapter we presented a novel method to assess for parametric and non-parametric uncertainty quantification using the same model, a Conditional Wasserstein GAN, relying on physical submodels over an area of interest. We used for training decomposition and regularization methods for training deep learning models for physical data which were discussed in length in this manuscript. We have empirically shown that our methods obtain good generative abilities while reducing the dimensionality of the learning problem and thus reducing the training cost of our models by restricting our attention to the boundary of a submodel. We fulfill the necessary condition that the cost of each run of the physical submodel is smaller than the cost of running the full physical model. Better precision is reached in the parametric view, by using DcNR's. Besides, in the situation where some of the parameters distribution is unknown (epistemic uncertainties), only mixed approaches, like conditional GAN, are feasible. For that, using the Wasserstein-GAN as a boundary conditions generator, we showed a high value of the discrepancy and generative ability in the Monte Carlo sampling method, while keeping physical consistency thanks to the learned boundary conditions, thus offering better generative behavior in the exploration of density tails.

Chapter 7

Conclusion

Contents

7.1 Main results and contributions 133
7.2 Publication and valorizations 134
7.3 Perspectives 135

7.1 Main results and contributions

The thesis major goal was to develop effective and resilient numerical approaches, as well as deep learning methods, for the reduction of parametric and/or non-parametric contact models in structural dynamics, with impact zone scenarios that might grow over time on cabin aviation equipment. It is a matter of being able to swiftly investigate various impact scenarios by learning submodels and identifying the contact boundary conditions and/or the physical fields of the deformed seat. The examination of the spatial, temporal, and parametric variability of the scenario is then refined using a stable physical reduced order model.

In this work, we were only interested in non-intrusive and non-projection-based model reduction approaches, i.e.without performing a projection of the continuous PDEs on a reduced basis. Non-intrusive approaches rely only on data driven method of modeling without using any physical information in the construction or the training of the reduced model, whereas intrusive methods use all the physical information available such as the exact PDE solved to construct their reduced model by projecting the equations on lower dimension spaces or constraining models with the equations in the training phase. In our approach we started by investigating the capacity of linear models relying on modal analysis to construct a regression model for a non-reducible problem, we also showed the benefits and the costs of considering both parametric and time-space information in the modal analysis step.

We compared linear methods to deep convolutional neural regressor and we illustrate that for contact cases such as the one investigated, linear methods behave very poorly and it is recommended to use non-linear data driven methods such as DcNR. We also illustrate that having the time step as a parameter of the DcNR helps reducing the dependency of the error with respect to the time.

Afterwards we proposed a random generator of boundary conditions for fast submodels by using machine learning. We showed that the use of non-linear techniques in machine learning and data-driven methods is highly relevant. Since the goal is to learn the underlying probabilistic distribution of uncertainty in the data. We used a Generative Adversarial Networks where the

Wasserstein-GAN with gradient penalty variant offered optimal convergence results for our problem.

The objective of our approach was to train a GAN on data from a finite element method code so as to extract stochastic boundary conditions for faster finite element predictions on a submodel. In the exploitation phase, the framework can be viewed as a randomized and parameterized simulation generator on the submodel, which can be used as a Monte Carlo estimator. We have empirically shown that our methods obtain comparable and slightly better estimation of physical fields than classical neural networks approaches, while reducing the dimensionality of the learning problem and thus reducing the training cost of our models by restricting our attention to the boundary of a submodel, thus offering better generative behavior in the exploration of density tails.

Then we proposed a compression techniques for convolutional neural networks which is able to decrease the size of CNN models that are created via the minimization of the number of parameters that contribute to the models elevated levels of complexity. The overfitting phenomena is also reduced. In this work we presented a method of compressing convolutional neural networks for FEM physical data and approaches to optimize data from FEM models for CNN training. The compression approach proposed can also be applied to learning data in higher dimensions since the complexity of the models is linearly dependent of the dimension and actual deep learning code library only allow up to 3D data learning.

Finally we apply the generative model with a structural zoom approach to an explicit dynamic contact 3D problem. The main purpose of this work is to evaluate the stability of the training algorithm proposed and the precision of the Monte Carlo estimator constructed with respect to the true solutions of interest, which are the displacement fields. Our objective was to study wave propagation and reflection phenomena on the plate following the shock over a discretized time grid. In this work we presented a novel method to assess for parametric and non-parametric uncertainty quantification using the same model, a Conditional Wasserstein GAN, relying on physical submodels over an area of interest. We used for training decomposition and regularization methods for training deep learning models for physical data which were discussed in length in this manuscript. We have empirically shown that our methods obtain good generative abilities while reducing the dimensionality of the learning problem and thus reducing the training cost of our models by restricting our attention to the boundary of a submodel. We fulfill the necessary condition that the cost of each run of the physical submodel is smaller than the cost of running the full physical model. Besides, in the situation where some of the parameters distribution is unknown (epistemic uncertainties), only mixed approaches, like conditional GAN, are feasible. For that, using the Wasserstein-GAN as a boundary conditions generator, we showed a high value of the discrepancy and generative ability in the Monte Carlo sampling method, while keeping physical consistency thanks to the learned boundary conditions, thus offering better generative behavior in the exploration of density tails.

7.2 Publication and valorizations

- Results from chapter 3 were submitted and accepted in ESAIM : Proceedings and Surveys as proceedings following CEMRACS 2021 : **Parametrized non-intrusive space-time approximation for explicit dynamic fem applications**, *Hamza BOUKRAICHI, Nassim RAZAALY, Nissrine AKKARI, Fabien CASENAVE, David RYCKELYNCK*.

- Results from chapter 4 were submitted and accepted in IFAC-PapersOnLine following Mathmod 2022 : **Uncertainty quantification in a mechanical submodel driven by a**

Wasserstein-GAN, *Hamza BOUKRAICHI, Nissrine AKKARI, Fabien CASENAVE, David RYCKELYNCK, IFAC-PapersOnLine,Volume 55, Issue 20, 2022.*

- Results from chapters 5 and 6 will soon be submitted for publication in journals.
- Some results from chapters 5 and 6 have been the subject of a patent application.
- Developpement undertaken in 5 and 6 lead to a code library combining all the approaches presented and developped in this thesis, this library is available open source *(Repository links will be added upon publication)* for results reproducibility purposes.

7.3 Perspectives

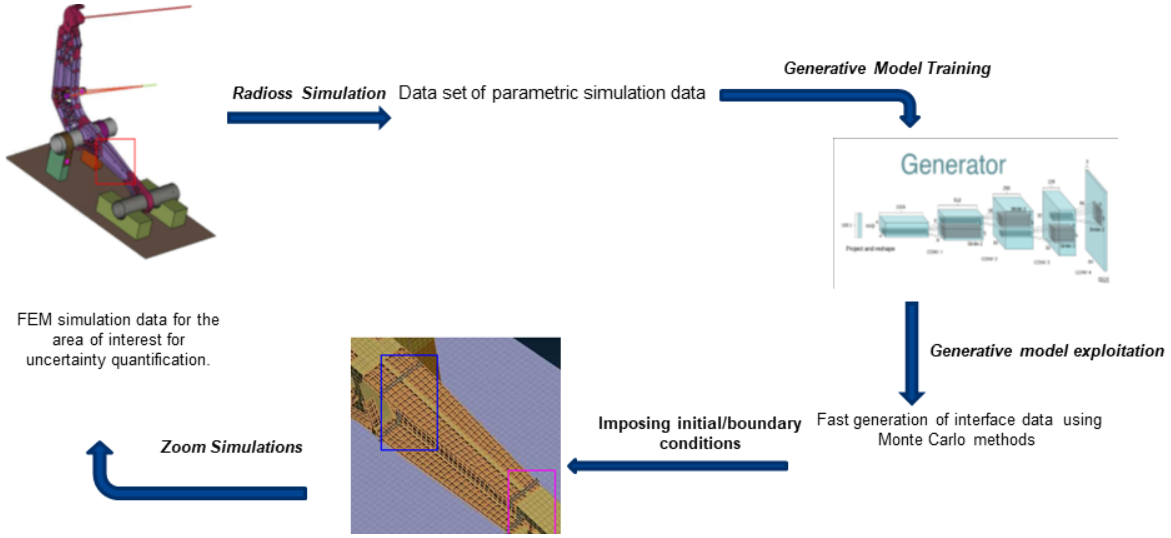


Figure 7.1: Uncertainty quantification computation loop for the spreader

The principal perspective of this work might the implementation of the full computational uncertainty quantification loop developed in this manuscript as it can be seen in Figure 7.1.

In addition, an extension of the work carried on this thesis is to focus on the reliability of a GAN to produce fully coherent physical boundary data, even though convergence results have been already published and metrics for quantifying their generative capabilities for image synthesis has already been proven, but except in the paradigm of physics informed neural nets, there is no metrics to assess for their reliability in physical uncertainty quantification context. A logical next step would be the development of such metrics. Finally, given the recent research on generative models, such as diffusion models, which outperformed GANs in most image synthesis problem, one could consider adapting them to the same paradigm proposed here for generating boundary conditions for FEM models.

Bibliography

Abbasnejad, M. E., Shi, Q., Hengel, A. v. d., and Liu, L. (2019a). A generative adversarial density estimator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10782–10791.

Abbasnejad, M. E., Shi, Q., Hengel, A. V. D., and Liu, L. (2019b). A generative adversarial density estimator. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Adamson, P. (2012). Stanford encyclopedia of philosophy.

Adhikari, S. and Chowdhury, R. (2010). A reduced-order random matrix approach for stochastic structural dynamics. *Computers & Structures*, 88(21-22):1230–1238.

Akkari, N., Casenave, F., Daniel, T., and Ryckelynck, D. (2021). Data-targeted prior distribution for variational autoencoder. *Fluids*, 6(10):343.

Akkari, N., Mercier, R., and Moureau, V. (2018). Geometrical reduced order modeling (rom) by proper orthogonal decomposition (pod) for the incompressible navier stokes equations. In *2018 AIAA Aerospace Sciences Meeting*, page 1827.

Almroth, B. O., Stern, P., and Brogan, F. A. (1978). Automatic choice of global shape functions in structural analysis. *Aiaa Journal*, 16(5):525–528.

Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., and Wells, G. N. (2015). The fenics project version 1.5. *Archive of Numerical Software*, 3(100).

An, S. S., Kim, T., and James, D. L. (2008). Optimizing cubature for efficient integration of subspace deformations. *ACM transactions on graphics (TOG)*, 27(5):1–10.

Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. (2018). Discriminator rejection sampling. *arXiv preprint arXiv:1810.06758*.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Balajewicz, M., Amsallem, D., and Farhat, C. (2015). Projection-based model reduction for contact problems.

Balajewicz, M., Amsallem, D., and Farhat, C. (2016a). Projection-based model reduction for contact problems. *International Journal for Numerical Methods in Engineering*, 106(8):644–663.

Balajewicz, M., Amsallem, D., and Farhat, C. (2016b). Projection-based model reduction for contact problems. *International Journal for Numerical Methods in Engineering*, 106(8):644–663.

Barratt, S. and Sharma, R. (2018). A note on the inception score. *arXiv preprint arXiv:1801.01973*.

Batou, A., Soize, C., and Audebert, S. (2015). Model identification in computational stochastic dynamics using experimental modal data. *Mechanical Systems and Signal Processing*, 50-51:307–322.

Beccantini, A., Bliard, F., Bouda, P., de Lambert, S., Drui, F., Galon, P., Jamond, O., and Lelong, N. (2022). Europlexus: un code de référence pour la dynamique rapide et l'interaction fluide-structure. In *CSMA 2022-15ème Colloque National en Calcul des Structures*.

Benaceur, A. (2018). *Réduction de modèles en thermique et mécanique non-linéaires*. Theses, Université Paris-Est Marne la Vallée.

Benaceur, A., Ern, A., and Ehrlacher, V. (2020). A reduced basis method for parametrized variational inequalities applied to contact mechanics. *International Journal for Numerical Methods in Engineering*, 121(6):1170–1197.

Bigolin Lanfredi, R., Schroeder, J. D., Vachet, C., and Tasdizen, T. (2019). Adversarial regression training for visualizing the progression of chronic obstructive pulmonary disease with chest x-rays. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 685–693. Springer.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877.

Bode, M., Gauding, M., Lian, Z., Denker, D., Davidovic, M., Kleinheinz, K., Jitsev, J., and Pitsch, H. (2019). Using physics-informed super-resolution generative adversarial networks for subgrid modeling in turbulent reactive flows. *CoRR*, abs/1911.11380.

Boget, Y. (2019). Adversarial regression. generative adversarial networks for non-linear regression: Theory and assessment. *arXiv preprint arXiv:1910.09106*.

Borg, I. and Groenen, P. J. (2005). *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.

Bottou, L. and Bousquet, O. (2007). The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20.

Boukraichi, H., Akkari, N., Casenave, F., and Ryckelynck, D. (2021). Uncertainty quantification in a mechanical submodel driven by a wasserstein-gan. *arXiv preprint arXiv:2110.13680*.

Box, G. E. and Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Calandra, H., Gratton, S., Riccietti, E., and Vasseur, X. (2019). On the approximation of the solution of partial differential equations by artificial neural networks trained by a multilevel levenberg-marquardt method. *arXiv preprint arXiv:1904.04685*.

Casella, G. (1985). An introduction to empirical bayes data analysis. *The American Statistician*, 39(2):83–87.

Casenave, F., Akkari, N., Bordeu, F., Rey, C., and Ryckelynck, D. (2020). A nonintrusive distributed reduced-order modeling framework for nonlinear structural mechanics—application to elastoviscoplastic computations. *International journal for numerical methods in engineering*, 121(1):32–53.

Chatterjee, A. (2000). An introduction to the proper orthogonal decomposition. *Current science*, pages 808–817.

Chen, W., Wilson, J. T., Tyree, S., Weinberger, K. Q., and Chen, Y. (2015). Compressing convolutional neural networks. *arXiv preprint arXiv:1506.04449*.

Chinesta, F., Ammar, A., and Cueto, E. (2010). Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models. *Archives of Computational methods in Engineering*, 17(4):327–350.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65.

Csáji, B. C. et al. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7.

Domanov, I. and Lathauwer, L. D. (2014). Canonical polyadic decomposition of third-order tensors: Reduction to generalized eigenvalue decomposition. *SIAM Journal on Matrix Analysis and Applications*, 35(2):636–660.

Dubey, S. R., Singh, S. K., and Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Eckart, C. and Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218.

Everson, R. and Sirovich, L. (1995a). Karhunen–loeve procedure for gappy data. *JOSA A*, 12(8):1657–1664.

Everson, R. and Sirovich, L. (1995b). Karhunen–loeve procedure for gappy data. *JOSA A*, 12(8):1657–1664.

Evert, E., Vandecappelle, M., and De Lathauwer, L. (2022). Canonical polyadic decomposition via the generalized schur decomposition. *IEEE Signal Processing Letters*, 29:937–941.

Farhat, C., Avery, P., Chapman, T., and Cortial, J. (2014). Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency. *International Journal for Numerical Methods in Engineering*, 98(9):625–662.

Farhat, C., Bos, A., Tezaur, R., Chapman, T., Avery, P., and Soize, C. (2018). A stochastic projection-based hyperreduced order model for model-form uncertainties in vibration analysis. In *2018 AIAA Non-Deterministic Approaches Conference*, page 1410.

Fauque, J., Ramière, I., and Ryckelynck, D. (2018). Hybrid hyper-reduced modeling for contact mechanics problems. *International Journal for Numerical Methods in Engineering*, 115(1):117–139.

Fefferman, C., Mitter, S., and Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049.

Friderikos, O., Olive, M., Baranger, E., Sagris, D., and David, C. (2021). A non-intrusive space-time interpolation from compact stiefel manifolds of parametrized rigid-viscoplastic fem problems. *arXiv preprint arXiv:2102.09216*.

Gastaldi, C., Zucca, S., and Epureanu, B. I. (2018). Jacobian projection reduced-order models for dynamic systems with contact nonlinearities. *Mechanical Systems and Signal Processing*, 100:550–569.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58.

Giacoma, A., Dureisseix, D., Gravouil, A., and Rochette, M. (2014). A multiscale large time increment/fas algorithm with time-space model reduction for frictional contact problems. *International Journal for Numerical Methods in Engineering*, 97(3):207–230.

Golub, G. H. and Van der Vorst, H. A. (2000). Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1-2):35–65.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Guedri, M., Cogan, S., and Bouhaddi, N. (2012). Robustness of structural reliability analyses to epistemic uncertainties. *Mechanical Systems and Signal Processing*, 28:458–469.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans.

Géradin, M. and Rixen, D. J. (2016). A 'nodeless' dual superelement formulation for structural and multibody dynamics application to reduction of contact problems. *International Journal for Numerical Methods in Engineering*, 106(10):773–798.

Haas, M. and Richter, S. (2020). Statistical analysis of wasserstein gans with applications to time series forecasting. *arXiv preprint arXiv:2011.03074*.

Hameed, M. G. A., Tahaei, M. S., Mosleh, A., and Nia, V. P. (2022). Convolutional neural network compression through generalized kronecker product decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 771–779.

Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510.

Hernandez, Q., Badías, A., González, D., Chinesta, F., and Cueto, E. (2021). Deep learning of thermodynamics-aware reduced-order models from data. *Computer Methods in Applied Mechanics and Engineering*, 379:113–763.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Ioffe, S. and Szegedy, C. (2015a). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Ioffe, S. and Szegedy, C. (2015b). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

James, G. M. (2003). Variance and bias for general loss functions. *Machine learning*, 51(2):115–135.

Jia, X. (2021). Physics-guided machine learning: A new paradigm for scientific knowledge discovery. *Microscopy and Microanalysis*, 27(S1):1344–1345.

Jolicoeur-Martineau, A. (2019). The relativistic discriminator: a key element missing from standard GAN. In *International Conference on Learning Representations*.

Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., and Solenthaler, B. (2019). Deep fluids: A generative network for parameterized fluid simulations. In *Computer graphics forum*, volume 38, pages 59–70. Wiley Online Library.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

Kullback, S. (1997). *Information theory and statistics*. Courier Corporation.

Labatie, A. (2019). Characterizing well-behaved vs. pathological deep neural networks. In *International Conference on Machine Learning*, pages 3611–3621. PMLR.

Ladevèze, P., Passieux, J.-C., and Néron, D. (2010). The latin multiscale computational method and the proper generalized decomposition. *Computer Methods in Applied Mechanics and Engineering*, 199(21-22):1287–1296.

Launay, H. (2021). *Reduced models via machine learning to analyse the criticality of defects*. PhD thesis. Thèse de doctorat dirigée par Ryckelynck, David et Willot, François Mécanique Université Paris sciences et lettres 2021.

Le Berre, S., Ramière, I., Fauque, J., and Ryckelynck, D. (2022). Condition number and clustering-based efficiency improvement of reduced-order solvers for contact problems using lagrange multipliers. *Mathematics*, 10(9):1495.

Le Guennec, Y., Brunet, J.-P., Daim, F.-Z., Chau, M., and Tourbier, Y. (2018). A parametric and non-intrusive reduced order model of car crash simulation. *Computer Methods in Applied Mechanics and Engineering*, 338:186–207.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Legault, J., Langley, R., and Woodhouse, J. (2012). Physical consequences of a nonparametric uncertainty model in structural dynamics. *Journal of Sound and Vibration*, 331(25):5469–5487.

Liu, J.-P., Shu, X.-B., Kanazawa, H., Imaoka, K., Mikkola, A., and Ren, G.-X. (2018). A model order reduction method for the simulation of gear contacts based on arbitrary lagrangian eulerian formulation. *Computer Methods in Applied Mechanics and Engineering*, 338:68–96.

Liu, M.-Y. and Tuzel, O. (2016). Coupled generative adversarial networks. *Advances in neural information processing systems*, 29.

Long, Z., Lu, Y., Ma, X., and Dong, B. (2018). Pde-net: Learning pdes from data. In *International Conference on Machine Learning*, pages 3208–3216. PMLR.

Lumley, J. L. (1967). The structure of inhomogeneous turbulent flows. *Atmospheric turbulence and radio wave propagation*, pages 166–178.

Mallat, S. (2016). Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203.

Mescheder, L., Geiger, A., and Nowozin, S. (2018). Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Mishra, S. (2019). A machine learning framework for data driven acceleration of computations of differential equations. *Mathematics in Engineering*, 1(1):118–146.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*.

Muller, N., Magaia, L., and Herbst, B. M. (2004). Singular value decomposition, eigenfaces, and 3d reconstructions. *SIAM Review*, 46(3):518–545.

Murthy, R., Wang, X., Perez, R., Mignolet, M. P., and Richter, L. A. (2012). Uncertainty-based experimental validation of nonlinear reduced order models. *Journal of Sound and Vibration*, 331(5):1097–1114.

Obukhov, A. and Krasnyanskiy, M. (2020). Quality assessment method for gan based on modified metrics inception score and fréchet inception distance. In *Proceedings of the Computational Methods in Systems and Software*, pages 102–114. Springer.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

Pham, G.-T., Boyer, R., and Nielsen, F. (2018a). Computational information geometry for binary classification of high-dimensional random tensors. *Entropy*, 20(3):203.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018b). Efficient neural architecture search via parameters sharing. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4095–4104. PMLR.

Pilipović, R., Bulić, P., and Risojević, V. (2018). Compression of convolutional neural networks: A short survey. In *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6. IEEE.

Preuer, K., Renz, P., Unterthiner, T., Hochreiter, S., and Klambauer, G. (2018). Fréchet chemblnet distance: A metric for generative models for molecules. *arXiv preprint arXiv:1803.09518*.

Raissi, M. and Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141.

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017a). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017b). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2018). Numerical gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 40(1):A172–A198.

Ranftl, S., von der Linden, W., and Committee, M. . S. (2021). Bayesian surrogate analysis and uncertainty propagation. In *Physical Sciences Forum*, volume 3, page 6. MDPI.

Reynders, E., Langley, R. S., Dijckmans, A., and Vermeir, G. (2014). A hybrid finite element – statistical energy analysis approach to robust sound transmission modeling. *Journal of Sound and Vibration*, 333(19):4621–4636.

Roman, V. (2019). Unsupervised machine learning: Clustering analysis. *Towards Data Science*.

Rumelhart, D. E., Durbin, R., Golden, R., and Chauvin, Y. (1995). Backpropagation: The basic theory. *Backpropagation: Theory, architectures and applications*, pages 1–34.

Ryckelynck, D. (2005). A priori hyperreduction method: an adaptive approach. *Journal of Computational Physics*, 1(202):346–366.

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29.

Schaeffer, H. (2017). Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446.

Sheffer, A., Praun, E., Rose, K., et al. (2007). Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision*, 2(2):105–171.

Simo, J. C. and Laursen, T. (1992). An augmented lagrangian treatment of contact problems involving friction. *Computers & Structures*, 42(1):97–116.

Singh, S., Uppal, A., Li, B., Li, C.-L., Zaheer, M., and Póczos, B. (2018). Nonparametric density estimation under adversarial losses. *arXiv preprint arXiv:1805.08836*.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. *Part III: dynamics and scaling. Quarterly of applied mathematics*, 45:583–590.

Soize, C. (2000). A nonparametric model of random uncertainties for reduced matrix models in structural dynamics. *Probabilistic Engineering Mechanics*, 15(3):277–294.

Soize, C. (2005). Random matrix theory for modeling uncertainties in computational mechanics. *Computer methods in applied mechanics and engineering*, 194(12-16):1333–1366.

Sun, R.-Y. (2020). Optimization for deep learning: An overview. *Journal of the Operations Research Society of China*, 8(2):249–294.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.

Turner, R., Hung, J., Frank, E., Saatchi, Y., and Yosinski, J. (2019). Metropolis-hastings generative adversarial networks. In *International Conference on Machine Learning*, pages 6345–6353. PMLR.

Villani, C. (2009). *Optimal transport: old and new*, volume 338. Springer.

Wang, X., Mignolet, M. P., and Soize, C. (2020). Structural uncertainty modeling for nonlinear geometric response using nonintrusive reduced order models. *Probabilistic Engineering Mechanics*, 60:103033.

Wriggers, P. (2006). *Computational contact mechanics*, volume 2. Springer.

Xie, L., Chen, X., Bi, K., Wei, L., Xu, Y., Wang, L., Chen, Z., Xiao, A., Chang, J., Zhang, X., et al. (2021). Weight-sharing neural architecture search: A battle to shrink the optimization gap. *ACM Computing Surveys (CSUR)*, 54(9):1–37.

Xie, Y., Franz, E., Chu, M., and Thuerey, N. (2018a). tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4):1–15.

Xie, Y., Franz, E., Chu, M., and Thuerey, N. (2018b). tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4):1–15.

Yang, Y. and Perdikaris, P. (2019). Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394:136–152.

Yeo, I.-K. and Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959.

You, L., Zhang, J., Du, X., and Wu, J. (2020). A new structural reliability analysis method in presence of mixed uncertainty variables. *Chinese Journal of Aeronautics*, 33(6):1673–1682.

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010a). Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE.

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010b). Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE.

RÉSUMÉ

L'Objectif principal de la thèse est le développement de méthodes numériques et des méthodes de deep learning efficaces et robustes pour la réduction des modèles paramétriques et/ou non-paramétriques de contact en dynamique des structures, avec des scénarios de zones d'impact qui peuvent évoluer au niveau de l'équipement aéronautique en cabine. L'approche consiste à déterminer une zone d'intérêt dans le modèle physique et à construire des modèles capables de générer des conditions aux limites autour de la zone d'intérêt pour le modèle physique. Cette modélisation permettra d'explorer l'espace paramétrique à l'aide du modèle génératif tout en conservant les caractéristiques de haute fidélité des solutions physiques en résolvant le problème physique dans la zone d'intérêt, puis de l'utiliser pour tester une variété de scénarios d'impact. Réduisant ainsi le coût de calcul du modèle physique. Notre code source pour Europlexus sera utilisé pour créer le programme. Il y aura plus de développement Python pour les méthodes d'apprentissage automatique.

MOTS CLÉS

Dynamique des structure, Apprentissage profond, Apprentissage automatique, Modèles d'ordre réduit, Modèles génératifs, Quantification d'incertitude, Sous-modélisation, Zoom structurel, Apprentissage supervisé, Modèles de régression.

ABSTRACT

The primary goal of this thesis is to develop efficient and reliable numerical methods and deep learning methods for the reduction of parametric and/or non-parametric contact models in structural dynamics, including impact zone scenarios that can evolve over time on cabin aeronautical equipment. The approach is to determine a zone of interest in the physical model and construct models capable of generating boundary conditions to the physical model around the zone of interest. This modelisation will allow to explore the parametric space using the generative model while keeping the high-fidelity caracteristics of the physical solutions by solving the physical problem in the area of interest, and then use it to test out a variety of impact scenarios. Thus reducing the computational cost of the physical model. Our source code for Europlexus will be used to create the program. There will be more Python development for deep learning methods.

KEYWORDS

Structural dynamics, Deep Learning, Machine Learning, Reduced Order Models, Generative Models, Uncertainty Quantification, Submodeling, Structural Zoom, Supervised Learning, Regression Models.