Machine learning for nonlinear model order reduction

Thomas Daniel

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES ParisTech

Machine learning for nonlinear model order reduction

Apprentissage statistique pour la réduction de modèle non-linéaire

Soutenue par
Thomas DANIEL
Le 24/09/2021

Ecole doctorale n° 621
Ingénierie des Systèmes, Matériaux, Mécanique, Energétique (ISMME)

Spécialité
Mécanique

Composition du jury :

Yvon MADAY LJLL – Sorbonne Université	*Président du Jury*
Virginie EHRLACHER CERMICS – Ponts ParisTech	*Rapporteuse*
Ioannis STEFANOU GeM – Centrale Nantes	*Rapporteur*
Stefanie REESE IFAM – RWTH Aachen University	*Examinatrice*
Kathrin SMETANA Stevens Institute of Technology	*Examinatrice*
Fabien CASENAVE SafranTech – Safran Group	*Examinateur*
Nissrine AKKARI SafranTech – Safran Group	*Examinatrice*
David RYCKELYNCK Centre des Matériaux – Mines ParisTech	*Directeur de thèse*

MINES ParisTech | PSL

École Doctorale ISMME

THÈSE

présentée pour l'obtention du diplôme de

DOCTEUR

de l'École des Mines ParisTech

Spécialité : Mécanique

par

Thomas DANIEL

intitulée

Machine learning for nonlinear model order reduction

soutenue le 24/09/2021, devant le jury composé de :

Yvon MADAY	Président du Jury	LJLL - Sorbonne Université
Virginie EHRLACHER	Rapporteuse	CERMICS - Ponts ParisTech
Ioannis STEFANOU	Rapporteur	GeM - Centrale Nantes
Stefanie REESE	Examinatrice	IFAM - RWTH Aachen University
Kathrin SMETANA	Examinatrice	Stevens Institute of Technology
Mickaël ABBAS	Invité	EDF R&D
Cédric LEBLOND	Invité	Naval Group & JLMT
David RYCKELYNCK	Directeur de thèse	Mines ParisTech
Fabien CASENAVE	Encadrant	SafranTech - Safran Group
Nissrine AKKARI	Co-encadrante	SafranTech - Safran Group

Remerciements

Tout d'abord, je souhaite remercier mon directeur de thèse David Ryckelynck et mes encadrants Fabien Casenave et Nissrine Akkari pour l'opportunité qu'ils m'ont offerte de travailler avec eux sur ce sujet de thèse, et pour l'excellent encadrement dont j'ai bénéficié tout au long de ces trois années. Vous m'avez transmis votre passion pour vos sujets de recherche, et avez su me guider tout en me donnant de l'autonomie et des libertés dans mon travail. Je garderai un très bon souvenir de nos nombreuses discussions scientifiques, et je vous suis très reconnaissant de votre accompagnement ainsi que de votre bienveillance à mon égard. Les résultats présentés dans ce mémoire de thèse sont le fruit d'un vrai travail d'équipe auquel je suis fier d'avoir participé. Cette expérience n'aurait pas été la même sans vous. Merci aussi à Fabien pour les nombreux outils de programmation qu'il a développés et qui m'ont été très utiles pour l'implémentation de nos idées, notamment la librairie Python *Mordicus* pour la réduction de modèle non-intrusive. Je remercie aussi Ali Ketata, que j'ai eu le plaisir d'encadrer pendant son stage à SafranTech pour creuser de nouvelles idées et explorer différentes pistes en lien avec ma thèse.

Un grand merci à Virginie Ehrlacher et Ioannis Stefanou d'avoir accepté de relire ce mémoire en tant que rapporteurs, ainsi que pour leurs précieux commentaires. Merci également à Yvon Maday, Stefanie Reese, Kathrin Smetana, Mickaël Abbas et Cédric Leblond d'avoir accepté de faire partie de mon jury de thèse. C'est un honneur pour moi d'avoir pu présenter mes travaux devant un jury aussi prestigieux.

J'ai eu plaisir à venir travailler à SafranTech où j'ai rencontré des personnes exceptionnelles, tant sur le plan humain que sur le plan scientifique. Travailler auprès de docteurs-ingénieurs experts dans leurs domaines, sur des thématiques de recherche motivées par de réels besoins en lien avec les activités du Groupe Safran, a été un vrai privilège. J'ai beaucoup appris à vos côtés, et ai largement bénéficié de votre savoir-faire et de vos outils pour mener à bien ma thèse. À ce sujet, je souhaite tout particulièrement mentionner Felipe, Julien et Fabien pour la superbe librairie Python *BasicTools*[1] dont je me suis beaucoup servi pour traiter des données de simulations, ainsi que Christian, Sébastien, Clément et Frédéric pour leurs précieux conseils. Je pense aussi à toute l'équipe du projet auquel ma thèse était rattachée, qui m'a permis de découvrir les thématiques passionnantes associées à la modélisation d'une aube de turbine haute pression dans un turboréacteur : merci à Augustin, Oana, Romain, Teddy, et Tonya. Je remercie aussi Ayoub, Brian, Grégory, Mohamed, Nicolas, Pierre et Xavier, avec qui j'ai eu plaisir à discuter, et garde un souvenir ému du passage malheureusement trop court d'Alexandros à SafranTech.

Je garderai un très bon souvenir de la bonne ambiance entre doctorants, à SafranTech comme au Centre des Matériaux de Mines ParisTech. Côté Safran, je me rappellerai du co-voiturage avec Martin accompagné des chansons de Dire Straits après les séances de sport, des bons conseils de mon voisin et compatriote nissart Adrien, de la bonne pastilla marocaine chez Mouad, des blagues d'Anthony, et des bons moments partagés avec eux et Clément, Florian, Hamza, Maxence, William, et Yannis. Côté Centre des Matériaux, je me souviendrai de l'excellente cohésion parmi les doctorants, et je tiens en particulier à remercier Laurent et Hugo pour leur aide précieuse sur le logiciel Zset.

Je tiens également à remercier le département GMM (Génie Mécanique et Matériaux) de l'École des Ponts ParisTech pour m'avoir donné l'opportunité de créer un nouveau

[1] https://gitlab.com/drti/basic-tools.

cours avec mon directeur de thèse sur l'application du machine learning à la simulation numérique en mécanique. En tant qu'ancien étudiant du département, j'ai été ravi de participer à ce projet et enseigner aux élèves-ingénieurs de l'école.

J'ai une pensée aussi pour toutes les personnes du Centre des Matériaux qui font en sorte que les thèses se déroulent au mieux, soutiennent les doctorants, leur proposent des formations, et les accompagnent dans les tâches administratives. Merci à elles.

Sur un plan plus personnel, ces dernières années marquées par la crise du Covid m'ont amené à redéfinir mes projets pour l'après-thèse. Je remercie chaleureusement toutes les personnes qui m'ont aidé dans cette réflexion et dans ma recherche d'emploi. J'ai aujourd'hui l'immense bonheur de retourner dans le sud de la France, et c'est un peu grâce à chacun d'entre vous que je concrétise ce projet qui m'était devenu si cher.

Enfin, je remercie Amal et ma famille pour leur soutien inconditionnel, dans les bons moments comme dans les moins bons. Plus généralement, je pense à toute ma famille au sens large, en France et au Maroc, et à mes amis, pour tous les moments agréables que nous avons partagés ensemble. Avec les nombreux événements qui ont rythmé ces trois dernières années, je mesure la chance que j'ai de vous avoir.

À vous tous, ce mémoire de thèse est dédié.

* *
*

Table of Contents

Part I

Introduction

Chapter 1

Introduction

Contents

1.1 Motivations

Numerical simulations in physics have become an essential tool in many engineering domains. The development of high-performance computing has enabled engineers and scientists to use complex high-fidelity models for real-world applications, with ultra-realistic simulations involving millions of degrees of freedom. However, such simulations are too time-consuming to be integrated in design iterations in the industry. They are usually limited to the final validation and certification steps, while the design process still relies on simplified models. Accelerating these complex simulations is a key challenge, as it would provide useful numerical tools to improve design processes. The development of numerical methods for fast simulations would also enable using new models that have not been applied to industrial problems yet, because of their complexities. Uncertainty quantification is another important example of analysis that would become practicable if the cost of simulations was sufficiently reduced. Indeed, quantities of interest monitored in numerical simulations depend on the environment of the physical system, which is usually not exactly known. In some cases, these uncertainties strongly influence simulation results, and the probabilistic behavior of the quantities of interest must be studied in order to ensure the reliability of the industrial product.

The work presented in this thesis is funded by Safran and is motivated by the aeronautical industry's need for fast numerical methods to control the uncertainties in the design of high-pressure turbine blades in an aircraft engine, see Figure 1.1. As one of the world leaders in the aeronautical industry, Safran is committed to the reduction of the environmental impact of aviation thanks to the development of fuel-efficient engines while maintaining a high level of reliability. Improving the engine's efficiency requires increasing the temperature of the gases leaving the combustion chamber, which is a real

Figure 1.1: The LEAP, turbofan developed by CFM International, a joint venture between Safran Aircraft Engines and GE Aviation. This engine powers Airbus A320neo, Boeing 737 MAX and COMAC C919 planes. Picture taken from https://medialibrary.safran-group.com/Photos/media/178745. ©2017 Antonio Gomez, Safran.

challenge given the extremely severe thermomechanical conditions that high-pressure turbine blades already face. It is therefore crucial to predict the effects of uncertainties such as thermal loading uncertainties (see Figure 1.2), in order to know to what extent the mechanical behavior and thus the durability of the turbine blades are affected. However, computing the fatigue lifetime of a high-pressure turbine may take several weeks, which is incompatible with uncertainty quantification since it requires running many simulations, typically thousands of simulations. Domain decomposition methods partially solve this issue by leveraging advances in high-performance computing, but they are not necessarily adapted to many-query problems because of their intensive use of computational resources. Another solution is to reduce a bit the quality of the numerical predictions in exchange for faster simulations requiring fewer computational resources. Indeed, for uncertainty quantification purposes, knowing the exact values of many physics variables on the entire system is not always necessary, since uncertainties may be summarized with a few well-chosen quantities of interest. In this field, *artificial intelligence* (AI), and more specifically *machine learning*, has a major role to play. Domain decomposition methods remain also indispensable for the generation of high-fidelity training data for machine learning algorithms.

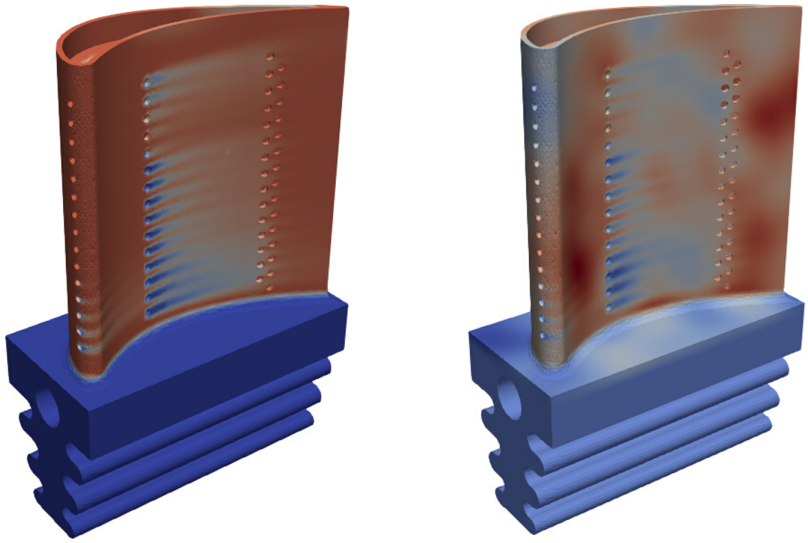

Figure 1.2: Uncertainties on the temperature field in a high-pressure turbine blade.

1.2 Computational physics assisted by artificial intelligence

Recent advances in machine learning has given birth to a new emerging research field, namely *AI-assisted computational physics*. This thesis focuses on the use of machine learning for the acceleration of computational methods in physics.

The most straightforward approach for the acceleration of numerical simulations is to use supervised learning to replace the high-fidelity numerical solver by a regression-based *surrogate model* (or *metamodel*), when the quantities of interest are well established. Using regression algorithms, one can directly estimate quantities of interest for given values of the uncertain parameters, without computing intermediate variables involved in the underlying physics model. When the quantity of interest is a field defined on the whole mesh of the physical system, dimensionality reduction may be required to reduce the number of outputs to predict. Using training data generated by the high-fidelity model, dimensionality reduction can identify a reduced set of latent variables[1] describing the quantity of interest. The *proper orthogonal decomposition* (POD [1, 2]) is a linear method defining the latent space as a low-dimensional subspace minimizing the projection error of training data. Generalization to nonlinear dimensionality reduction can be obtained with artificial neural networks using an *autoencoder* architecture [3, 4]. A regression algorithm can be applied in conjunction with one of these dimensionality reduction techniques in order to build a metamodel that predicts the latent variables given the time and the parameters' values. The quantity of interest can then be reconstructed, either using the autoencoder's decoder or by combining the modes of the POD basis. Examples can be found in [5, 6, 7] for POD-GPR (Gaussian process regression), and in [8] for POD-NN (using a neural network). These methods have the advantage of being very fast and non-intrusive by nature. The latent dynamics, *i.e.* the evolution of the latent variables with respect to time, can also be learnt by recurrent neural networks (*e.g.* Long Short-Term Memory networks, LSTM [4]), see [9, 10, 11] for POD-LSTM and [12, 13, 14, 15] for the combination of LSTM with autoencoders. Alternatively, the latent dynamics can be

[1] Latent variables are unobserved variables defined as functions of observable variables.

modeled with neural ordinary differential equations (NODEs) like in [16, 17].

All of the aforementioned methods are purely *data-driven*: apart from the training data that are generated by a physics model, no physics knowledge is incorporated in the learning process. Learning physics principles from data may require large datasets, especially for complex physics phenomena, which is incompatible with the computational cost of the high-fidelity model generating the training data. Nonetheless, regression-based surrogate models remain useful for the prediction of errors between the true solution and the approximate one [5, 18, 19, 20, 21]. But contrary to many applications of machine learning methods, applications in physics have the particularity of being described by mathematical models that have been developed by experts and researchers for many decades or even centuries. This valuable prior knowledge must be incorporated in AI methods to improve the quality of numerical predictions and compensate the lack of training data. In particular, purely data-driven approaches are likely to make predictions that do not satisfy basic physics principles. To accelerate the constitutive equations solver in solid mechanics, [22] introduces *thermodynamics-based artificial neural networks* (TANNs), which benefit from automatic differentiation [23] to get thermodynamically-consistent predictions satisfying the first and the second laws of thermodynamics. Applications to materials with elastoplastic behaviors show that, on test data out of the range of training data, predictions of TANNs remain very good in comparison with neural network-based surrogates trained for a classical regression task. These results illustrate how important it is to include physics laws in machine learning methods to improve their performances or to reduce the amount of training data that is required. Thermodynamics-aware neural networks have also been studied in [24].

Another seminal paper [25] has defined *physics-informed neural networks* (PINNs) to solve partial differential equations. The neural network takes the position and the time as inputs, and returns the corresponding value of the solution of the boundary-value problem. It is trained by minimizing the mean squared error loss penalized by a regularization term that forces the predictions to satisfy the equations of the boundary-value problem. The regularization term penalizes the squared residuals of the partial differential equations, the boundary conditions and the initial conditions. Variants of this method use the weak formulation [26] or the energy formulation [27] of the partial differential equations being studied. Other approaches use artificial intelligence in multi-fidelity methods, where a physics model makes a first prediction that is improved by a neural network to obtain a high-fidelity prediction. This is called *super-resolution*. This idea has been used in [28] with cycle-consistent generative adversarial networks (CycleGANs [29]).

Projection-based model order reduction [30, 31] is another discipline aiming at the acceleration of numerical simulations without discarding physics equations. The *POD Galerkin method* [2, 32] is a well-known projection-based method that solves partial differential equations in a suitable low-dimensional approximation space. This approximation space is constructed with the POD using high-fidelity training data. *Hyper-reduction* [33] has been introduced by David Ryckelynck in 2005 to maintain the speed of reduced-order models for nonlinear problems involving a lot of internal variables. In this thesis, we will use the POD Galerkin method with the *Empirical Cubature Method* (ECM [34]) for hyper-reduction. These methods have been implemented in a non-intrusive Python code by Fabien Casenave in the *FUI Mordicus* project involving several French companies, including Safran. Details can be found in [35].

1.3 Main contributions

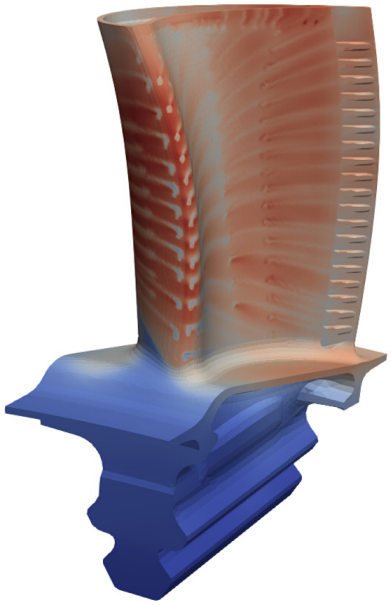

Figure 1.3: Temperature field on the high-pressure turbine blade studied in this thesis.

The objective of this thesis is to study the benefits of machine learning methods for nonlinear model order reduction. The model order reduction community is actively contributing to the assimilation of machine learning advances in computational physics. In particular, the use of AI algorithms is motivated by the study of *non-reducible* problems, that is, problems that cannot be solved accurately in a low-dimensional approximation space. Advection-dominated and wave propagation problems are classic examples of non-reducible problems. In this thesis, we rather focus on elliptic problems with high parameter sensitivity. Among the wide variety of methods dealing with non-reducible problems, *nonlinear manifold ROM methods* [36, 37, 38] generalize the POD Galerkin method by using an autoencoder to solve the equations on a nonlinear manifold. In this thesis, we rather adopt another method based on *clustering* [39] for the construction of *local* reduced-order models (ROMs) [40, 41], giving a dictionary of low-dimensional approximation spaces that are adapted to different regions of the solution manifold. Our work is also inspired by the ideas presented in [42, 43], where a classification algorithm learns to select the most appropriate approximation space for fast and accurate simulations. The novelties introduced in this thesis can be summarized into 4 major contributions:

- The definition of a physics-informed clustering strategy based on a *ROM-oriented dissimilarity measure* for the construction of dictionaries of local ROMs, with an illustration of the sensitivity of local reduced-order models' performances when changing the dissimilarity measure used for clustering. The Euclidean distance in the solution space, although commonly used, is shown to give irrelevant clusters in some specific cases, leading to local ROMs that can even deteriorate the performances of a single global ROM. The ROM-oriented dissimilarity measure has been introduced to remedy this issue. In particular, two solutions have a low dissimilarity if they can be reasonably well approximated in the same low-dimensional approximation space.

We introduce the concept of optimal partition of the solution manifold in terms of normalized Kolmogorov widths, and prove that the optimal partitions can be found by means of a representative-based clustering algorithm using this ROM-oriented dissimilarity measure.

- The definition of an a priori efficiency criterion evaluating the profitability of a dictionary of local ROMs associated to a classifier with respect to a single ROM. This criterion can be evaluated quite early in the training phase, before time-consuming operations. It also helps for the calibration of hyperparameters such as the number of local ROMs in the dictionary.

- The development of a variant of the *mRMR feature selection* algorithm [44, 45] for classification tasks on data coming from numerical simulations. This algorithm reduces the risk of overfitting for high-dimensional classification problems with small training sets by selecting a reduced set of features that are relevant for prediction purposes but not redundant. When dealing with random fields discretized on a mesh, it consists in identifying a few nodes of the mesh that give enough information for prediction.

- The definition of a new *data augmentation* algorithm for classification problems on physics data. It relies on the concept of *pure sets* in a labeled training set for the generation of new labeled examples from convex combinations of elements of the training set. Drawing points from the convex hulls of pure sets reduces the risk of generating new data with wrong labels.

This thesis also introduces 3 additional minor contributions:

- The distinction between low-fidelity and high-fidelity data to address the different needs in terms of the size of the training set for machine learning tasks and for model order reduction.

- The use of surrogate models to replace the Gappy POD [46] when reconstructing full fields from hyper-reduced predictions. The surrogate models take ROM predictions on a reduced-integration domain as inputs and return the coefficients in the corresponding POD basis to get predictions on the whole mesh.

- The demonstration of the applicability of our methodology to a real industrial problem, showing that it can be implemented without bottlenecks for large scale problems.

We have already presented most of these novelties in five papers [47, 48, 49, 50, 51], where our methodology is called *ROM-net*. Some sections of this thesis report are directly taken from these articles. The next chapter introduces machine learning methods that are used in this thesis. Model order reduction methods and strategies for non-reducible problems are presented in a separate chapter. The fourth chapter is the core of this thesis report: it defines our methodology and gives detailed descriptions of the contributions listed above. Finally, the last chapter shows the application of a ROM-net to a real high-pressure turbine blade, whose temperature field can be visualized in Figure 1.3. With 2 local hyper-reduced order models, the ROM-net accelerates a large scale nonlinear mechanical problem with a complex elasto-viscoplastic constitutive law under high parameter sensitivity by a factor

of 636, with error indicators in the order of 1% to 3% on quantities of interest (stress and strain variables). It enables the estimation of uncertainties on the mechanical behavior of the high-pressure turbine blades given uncertainties on the thermal loading.

* *
*

Part II

Machine learning methods

Résumé

Cette partie du mémoire introduit les principales notions d'apprentissage statistique (*machine learning* en anglais) utilisées dans ce travail de thèse. Nous commençons par définir ce qu'est l'apprentissage statistique et le situons par rapport à d'autres thématiques associées, telles que l'intelligence artificielle, la science des données, et l'apprentissage profond. À l'intersection entre statistiques et optimisation, le *machine learning* comprend entre autres des méthodes d'apprentissage supervisé à partir de données labellisées, et d'apprentissage non-supervisé à partir de données brutes. Ces méthodes permettent de construire des modèles prédictifs ou des algorithmes à partir de données d'entraînement, et peuvent aussi servir à extraire de l'information à partir de bases de données. Nous introduisons ensuite quelques concepts de la théorie des probabilités, de l'estimation de paramètres et de la théorie de l'information, afin de donner au lecteur les principales notions élémentaires utilisées en apprentissage statistique. Nous nous intéressons enfin aux opérateurs à noyau ainsi qu'au théorème de Mercer et à la transformée de Karhunen-Loève, dont les applications sont nombreuses dans ce travail via la simulation de processus stochastiques et de champs aléatoires, la décomposition orthogonale aux valeurs propres pour la réduction de modèle, et l'astuce du noyau.

Le chapitre suivant est consacré à la quantification d'incertitudes dans des modèles numériques potentiellement coûteux, et parle également de plans d'expériences numériques et de méthodes de Monte-Carlo. C'est dans ce chapitre que nous définissons les *variabilités non-paramétrées* : un modèle mathématique contient parfois des paramètres aléatoires dont les fluctuations ne peuvent être décrites par un nombre raisonnable de paramètres ; les fluctuations de ces paramètres aléatoires appartenant à un espace de grande dimension sont appelées *variabilités non-paramétrées*. Cette notion souligne un aspect important de ce travail de thèse, puisque nous nous intéressons à des applications en mécanique des milieux continus où les incertitudes portent non pas sur quelques propriétés scalaires des matériaux étudiés, mais sur un champ physique tel qu'un champ de température défini en tout point du système et pour lequel l'utilisateur ne dispose a priori d'aucun modèle paramétrique.

Cette partie du mémoire se termine par un large aperçu des algorithmes d'apprentissage statistique utilisés pendant la thèse. Un premier chapitre est dédié à l'apprentissage non-supervisé et en particulier aux méthodes de réduction de dimension et de partitionnement des bases de données. Un second chapitre présente les principaux algorithmes d'apprentissage supervisé pour la classification et la régression, ainsi que les méthodes d'apprentissage ensembliste permettant de réduire le sur-ajustement (ou surapprentissage). Le risque de surapprentissage est particulièrement important dans les problèmes considérés dans ce mémoire, où les données sont en grande dimension et en nombre limité. Nous verrons qu'il est par ailleurs possible de synthétiser l'information contenue dans ces

données pour une tâche précise grâce à des algorithmes de sélection de variables pertinentes et non-redondantes.

* *
*

Chapter 2

General and theoretical concepts

Abstract: *This chapter introduces basic concepts, definitions and theorems that are used throughout this thesis. After a brief introduction to machine learning, concepts and methods from probability theory, estimation theory and information theory are presented. Probability theory is particularly important for the description of uncertainties in a model and for the understanding of machine learning algorithms. Methods from estimation theory are extensively used in machine learning for parameter estimation from data. Machine learning methods also borrow concepts of information theory for the definition of loss functions for classification problems and for feature selection, for example. Finally, Mercer kernels and the Karhunen-Loève expansion are defined at the end of this chapter for their applications to random field simulation, kernel methods, and, as explained later, to model order reduction.*

Contents

2.1 What is machine learning?

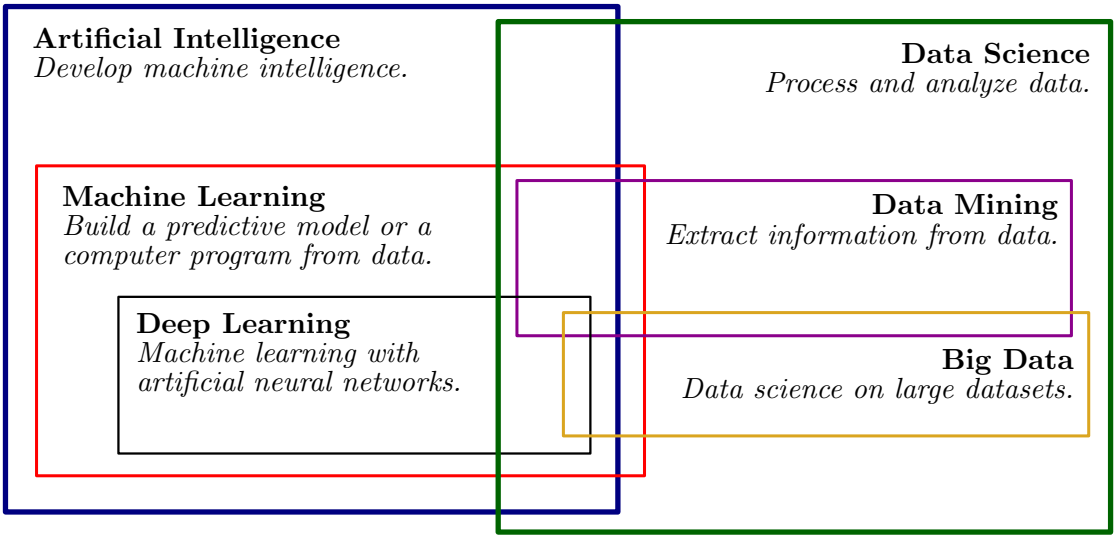

Figure 2.1: Machine learning and related fields.

Artificial intelligence is an interdisciplinary field whose goal is to develop machine intelligence, such as the ability to interact with the environment, collect and interpret data, learn from these data, and make context-adapted decisions. Artificial intelligence is related to computer science, applied mathematics, robotics, data science and machine learning. *Data science* uses mathematical and statistical methods to process, analyze, interpret, summarize and visualize data. *Big data* is a branch of data science dealing with very large datasets, whose development is motivated by the large amount of data that is collected through Internet, for example. *Data mining* is another important branch of data science dedicated to knowledge discovery. It includes all techniques and algorithms aiming at finding hidden structures or patterns and extracting information from databases, such as algorithms for cluster analysis, anomaly detection and association pattern mining.

Machine learning methods combine statistics and mathematical optimization to infer a predictive model from data, or more generally to build a computer program learnt from data that executes specific tasks without being explicitly programmed to perform these tasks. The concept of statistical learning refers to the calibration of the model's parameters using data, by means of an optimization algorithm. Machine learning problems sometimes use data mining algorithms and, conversely, machine learning techniques find many applications in data science. However, the specific goal of machine learning is to use knowledge drawn from data in order to make predictions for new unseen data, whereas data mining focuses on the extraction of knowledge. This leads to the dual concepts of training data, *i.e.* data used to build a predictive model, and test data, *i.e.* data considered in an exploitation phase for which the model will make new predictions (see Figure 2.2). With the recent development of GPUs (Graphics Processing Units) and computer performances, *deep learning* has emerged as a major branch of machine learning, where the predictive models rely on the superposition of simple functionalities in deep architectures. Such models are called *artificial neural networks* or *deep neural networks*, because of some similarities they share with neurons in the human brain. Figure 2.1 shows the intersections between all the fields that are related to machine learning.

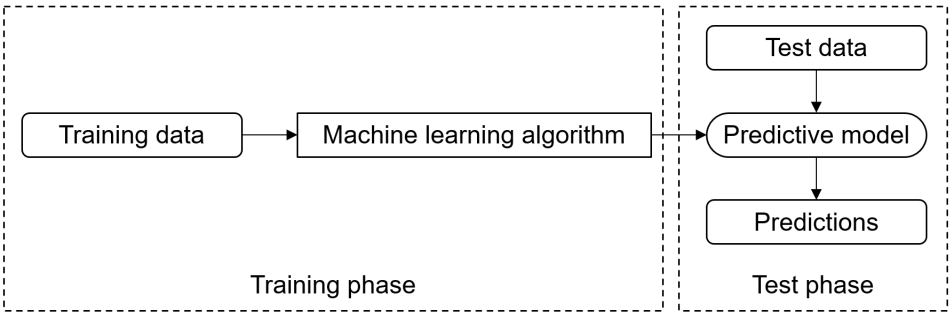

Figure 2.2: Training and test (or exploitation) phases of a machine learning model.

There are three basic learning paradigms:

- In *supervised learning*, every training example x is associated to a label y, and the model must learn the correspondence between the input x and the output y. The output variable can be either categorical (classification problems), or numerical with a continuous set of possible values (regression problems).
- In *unsupervised learning*, the training data are not labeled. Unsupervised learning problems include cluster analysis, where the objective is to divide the data into groups called clusters; dimensionality reduction, which aims at finding a compressed representation of the data; density estimation, consisting in identifying a probability density function that could have generated the training data; and generative modeling, whose goal is to generate new realistic data.
- In *reinforcement learning*, an agent learns to act in an environment through a trial-and-error process guided by a reward system. Typical applications are self-driving cars, autonomous robotics and game playing.

Some unsupervised learning tasks such as clustering and dimensionality reduction can be seen as data mining tasks. However, they are used in machine learning for prediction purposes or for preprocessing the data to be fed into a predictive model. The present thesis borrows algorithms from supervised and unsupervised methods:

- dimensionality reduction is used for the construction of reduced-order models, the compression of simulation data, and the reduction of the number of variables to be processed by some predictive models;
- cluster analysis is used for the identification of groups of data with similar physical or mechanical behaviors, enabling the construction of dictionaries of cluster-specific models;
- classification is used for automatic model recommendation;
- regression is used for the reconstruction of a physical field from localized predictions made by a reduced-order model.

2.2 Basic concepts

2.2.1 Probability theory

A *real-valued random variable* $X : \Theta \to \mathbb{R}$ is a function mapping the outcomes of a random process to real numbers. The function itself is deterministic, but its arguments are randomly chosen from the set Θ of possible outcomes. X is said to have a *probability density function* if there exists an integrable function $p : \mathbb{R} \to \mathbb{R}_+$ such that:

$$\mathbb{P}(X \leq x) = \int_{-\infty}^{x} p(s)ds \tag{2.1}$$

where $\mathbb{P}$ is a probability measure. For dx small enough, $p(x)dx$ can be interpreted as the probability that X returns a value between x and $x+dx$. A probability density function is nonnegative and its integral over $\mathbb{R}$ equals to one. If X is a discrete random variable, the term *probability mass function* is preferred. The probability mass function of a discrete random variable X following a categorical distribution (or multinoulli distribution) is defined by

$$p(x) = \sum_k \mathbb{P}(X = x_k)\delta(x - x_k) \tag{2.2}$$

where the variables x_k denote the discrete values that X can take, and δ is the Dirac delta function ensuring that the integral of p over $\mathbb{R}$ is one:

$$\int_{\mathbb{R}} p(x)dx = \sum_k \mathbb{P}(X = x_k) \quad = 1 \tag{2.3}$$

Let us now consider two real-valued random variables X and Y, having a *joint probability distribution* $p_{X,Y}$. The *marginal distribution* p_X can be calculated from the joint distribution using:

$$p_X(x) = \int_{\mathbb{R}} p_{X,Y}(x, y)dy \tag{2.4}$$

The *conditional distribution* $p_{Y|X}$ is defined as the probability distribution of Y for a given value of X, and reads:

$$p_{Y|X}(y|x) = \frac{p_{X,Y}(x, y)}{p_X(x)} \tag{2.5}$$

The random variables X and Y are *independent* if their joint distribution equals to the product of the marginal distributions.

The *expectation* (or *expected value* or *mean*) of a real-valued random variable X with probability density function p is defined by:

$$\mathbb{E}[X] = \int_{\Theta} X(\theta)d\mathbb{P}(\theta) = \int_{\mathbb{R}} xp(x)dx \tag{2.6}$$

if this integral exists. More generally, the expectation of the random variable $f(X)$ with $f : \mathbb{R} \to \mathbb{R}$ reads:

$$\mathbb{E}[f(X)] = \int_{\mathbb{R}} f(x)p(x)dx \tag{2.7}$$

The expectation operator is linear. The probability distribution is sometimes specified in the notation of the expectation to avoid confusion; in this case, the expectation of

$f(X)$ when X follows the probability distribution p is denoted by $\mathbb{E}_{X\sim p}[f(X)]$. If X is square-integrable, then one can define its *variance* as:

$$\mathrm{Var}(X) = \mathbb{E}\left[(X - E[X])^2\right] = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 \tag{2.8}$$

The *standard deviation* is the square root of the variance. From the definition of the variance, one can prove that $\forall (a,b) \in \mathbb{R}^2,\ \mathrm{Var}(aX+b) = a^2\mathrm{Var}(X)$. The *covariance* of two real-valued random variables X and Y is defined as:

$$\mathrm{Cov}(X,Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y] \tag{2.9}$$

and the *correlation* is obtained by normalizing the covariance:

$$\mathrm{Corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}} \tag{2.10}$$

Due to Cauchy-Schwarz inequality, the correlation satisfies $\mathrm{Corr}(X,Y) \in [-1;1]$. If X and Y are independent, then their covariance (and thus their correlation) is zero, but the converse is false. In other words, independence implies uncorrelatedness. The next definition introduces the concept of *multivariate random variable*:

Definition 2.2.1 (Multivariate random variable)**.** *Given an integer $n \geq 2$, a multivariate random variable (or random vector) is an application $\mathbf{X} : \Theta \to \mathbb{R}^n$ whose coordinates $X_i : \Theta \to \mathbb{R},\ \forall i \in [\![1;n]\!]$ are real-valued random variables. The expectation of $\mathbf{X}$ is the vector defined by:*

$$\mathbb{E}[\mathbf{X}] = (\mathbb{E}[X_1], \mathbb{E}[X_2], ..., \mathbb{E}[X_n]) \tag{2.11}$$

and the covariance matrix $\mathbf{\Gamma} \in \mathbb{R}^{n\times n}$ is defined by:

$$\mathbf{\Gamma} = \mathbb{E}\left[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^T\right] = \mathbb{E}\left[\mathbf{X}\mathbf{X}^T\right] - \mathbb{E}[\mathbf{X}]\,\mathbb{E}[\mathbf{X}]^T \tag{2.12}$$

which gives:

$$\forall (i,j) \in [\![1;n]\!]^2,\ \Gamma_{ij} = \mathrm{Cov}(X_i, X_j) = \mathbb{E}[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])] \tag{2.13}$$

It can be shown that the covariance matrix of a random vector is always symmetric positive semidefinite. The next definitions introduce Gaussian random variables and vectors:

Definition 2.2.2 (Standard normal distribution)**.** *A real-valued random variable X follows the standard normal distribution if its probability density function reads:*

$$\forall x \in \mathbb{R},\ p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{2.14}$$

We write: $X \sim \mathcal{N}(0,1)$.

It can be shown that if $X \sim \mathcal{N}(0,1)$, then its expectation is 0 and its variance is 1.

Definition 2.2.3 (Normal distribution)**.** *A real-valued random variable X follows the normal distribution (or Gaussian distribution) if there exist $U \sim \mathcal{N}(0,1)$ and $(\mu,\sigma) \in \mathbb{R}\times\mathbb{R}_+^*$ such that $X = \mu + \sigma U$. We write $X \sim \mathcal{N}(\mu,\sigma^2)$.*

Property 2.2.4. *If $X \sim \mathcal{N}(\mu, \sigma^2)$, then:*

- *its expectation is:* $\mathbb{E}[X] = \mu$;
- *its variance is:* $\mathrm{Var}(X) = \sigma^2$;
- *its probability density function reads:*

$$\forall x \in \mathbb{R},\ p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{2.15}$$

Definition 2.2.5 (Gaussian random vector)**.** *Let $\mathbf{X} : \Theta \to \mathbb{R}^n$ be a multivariate random variable. $\mathbf{X}$ is a Gaussian random vector if and only if all the linear combinations of its coordinates are normally distributed. In this case, $\mathbf{X}$ is said to follow the multivariate normal distribution and we write $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Gamma})$, where $\boldsymbol{\mu} \in \mathbb{R}^n$ is the mean vector and $\boldsymbol{\Gamma} \in \mathbb{R}^{n\times n}$ is the covariance matrix of $\mathbf{X}$.*

In particular, if $\mathbf{X}$ is a Gaussian random vector, then its coordinates follow the normal distribution. The converse of this property is false.

Property 2.2.6. *Let $\mathbf{X}$ be a Gaussian random vector. The random variables $\{X_i\}_{i\in[\![1;n]\!]}$ are independent if and only if the covariance matrix of $\mathbf{X}$ is diagonal. In other words, for Gaussian random vectors, independence and uncorrelatedness of the coordinates are equivalent.*

Property 2.2.7. *Let $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Gamma})$. $\mathbf{X}$ has a probability density function if and only if its covariance matrix $\boldsymbol{\Gamma}$ is invertible. In this case, the probability density function is given by:*

$$\forall \mathbf{x} \in \mathbb{R}^n,\quad p(x) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\boldsymbol{\Gamma})}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Gamma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.16}$$

To finish this review of basic concepts of probability theory, let us introduce Bayes' theorem:

Theorem 2.2.8 (Bayes' theorem)**.** *Let H (hypothesis) and E (evidence) be two events, with $\mathbb{P}(E) \neq 0$. Then:*

$$\mathbb{P}(H|E) = \frac{\mathbb{P}(E|H)\mathbb{P}(H)}{\mathbb{P}(E)} \tag{2.17}$$

The probability $\mathbb{P}(H)$ is called the *prior probability*, while $\mathbb{P}(H|E)$ is the *posterior probability* giving the probability of the hypothesis H given the observation of the evidence E. Bayes' theorem is used by generative classifiers to compute membership probabilities given class priors and modeled class-conditional probability distributions.

Remark 2.2.9. *For a more detailed introduction to probability theory, we refer the reader to the books [52, 53, 54].*

Notations: The j-th feature of a random vector X will be denoted by X^j. The observations of X (*resp.* X^j) will be denoted by x (*resp.* x^j), or x_i (*resp.* x_i^j) when indexing is necessary, for example when considering training data.

2.2.2 Estimation theory

In statistics, *estimation theory* includes all the techniques for parameter estimation from data. These methods are used for the calibration of the parameters of a model, and are therefore widely used by machine learning algorithms.

Suppose that one wants to model a data-generating distribution, denoted by p and parametrized by θ. The objective is to infer the value of θ from a sample containing n data points x_i, so that these data could have been drawn from the probability distribution $p(.;\theta)$. One way to find a plausible data-generating distribution is to use *maximum likelihood estimation* [55] (MLE). Considering the data points as observations of independent and identically distributed variables following the distribution $p(.;\theta)$, the *likelihood function* is defined as:

$$\theta \mapsto \prod_{i=1}^{n} p(x_i;\theta) \tag{2.18}$$

The maximum likelihood estimate θ_{MLE} is obtained by maximizing the likelihood function or, equivalently, the log-likelihood function:

$$\theta \mapsto \sum_{i=1}^{n} \log\ p(x_i;\theta) \tag{2.19}$$

The maximum likelihood estimate can be computed with the *expectation-maximization algorithm* [56, 55]. *Maximum a posteriori estimation* (MAP) differs slightly from MLE in that it describes the parameter θ as a random variable and introduces a prior distribution q in the function to maximize:

$$\theta \mapsto q(\theta) \prod_{i=1}^{n} p(x_i|\theta) \tag{2.20}$$

MAP estimators are used for example by generative classifiers, which compute the prior probabilities for each class, model class-conditional distribution using MLE, and use Bayes' theorem to compute posterior probabilities defining membership probabilities.

2.2.3 Information theory

Definition 2.2.10 (Differential entropy)**.** *Let X be a random variable with probability density function p_X. The differential entropy of X is defined by:*

$$H(X) = -\mathbb{E}_{X\sim p_X}\left[\log p_X(X)\right] \tag{2.21}$$

Definition 2.2.11 (Kullback-Leibler divergence)**.** *Let p and q be two probability distributions. The Kullback-Leibler divergence (or relative entropy) of p from q is defined by:*

$$D_{KL}(p||q) = \mathbb{E}_{X\sim p}\left[\log \frac{p(X)}{q(X)}\right] \tag{2.22}$$

The Kullback-Leibler divergence is nonnegative and equals to zero if and only if $p = q$ almost everywhere. It is usually used to compare two distributions, but it is not a metric on probability distributions since it is not symmetric and it does not satisfy the triangle inequality. The Kullback-Leibler divergence appears for example as a regularization term in the evidence lower bound (ELBO) defining the objective function to be maximized when training a variational autoencoder (VAE), see [3, 4, 57].

Definition 2.2.12 (Cross-entropy)**.** *Let p and q be two probability distributions. The cross-entropy of q relative to p is:*

$$H(p,q) = -\mathbb{E}_{X\sim p}\left[\log q(X)\right] \tag{2.23}$$

The cross-entropy is commonly used to define loss functions in classification problems. Minimizing the cross-entropy $H(p,q)$ with respect to q is equivalent to minimizing $D_{KL}(p||q)$, since:

$$H(p,q) = H(p) + D_{KL}(p||q) \tag{2.24}$$

Definition 2.2.13 (Mutual information [58], eq. 8.47, p. 251)**.** *Let X and Y be two real-valued random variables with joint probability distribution $p_{X,Y}$ and marginal distributions p_X and p_Y. The mutual information $I(X,Y)$ is defined by*

$$I\left(X,Y\right) = \int_{\mathbb{R}^2} p_{X,Y}(x,y)\log\left(\frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}\right)dxdy \tag{2.25}$$

The mutual information measures the mutual dependence between two random variables, *i.e.* it quantifies the information about Y that can be obtained by observing X. Contrary to correlation coefficients, the information provided by this score function is not limited to linear dependence. The mutual information is nonnegative, symmetric, and equals to zero if and only if the random variables are independent, properties that can be derived from the following equality:

$$I\left(X,Y\right) = D_{KL}(p_{X,Y}||p_Xp_Y) \tag{2.26}$$

The mutual information is commonly used by feature selection algorithms.

Remark 2.2.14. *For more details about information theory, see [58].*

2.3 Mercer kernels

2.3.1 Mercer's theorem and Karhunen-Loève expansion

Let $\mathcal{N}$ be a positive integer and let Ω be a compact set in the Euclidean space $\mathbb{R}^{\mathcal{N}}$, *i.e.* Ω is a closed and bounded subset of $\mathbb{R}^{\mathcal{N}}$ since the dimension $\mathcal{N}$ is finite. A *kernel* is a real-valued function $\kappa : \Omega\times\Omega \to \mathbb{R}$. The following definitions introduce concepts used in Mercer's theorem.

Definition 2.3.1 (Positive semidefinite kernel)**.** *A kernel $\kappa : \Omega\times\Omega \to \mathbb{R}$ is said to be positive semidefinite if it satisfies the two following properties:*

- *κ is symmetric:*
$$\forall(\mathbf{x},\mathbf{x}') \in \Omega\times\Omega, \quad \kappa(\mathbf{x},\mathbf{x}') = \kappa(\mathbf{x}',\mathbf{x}) \tag{2.27}$$
- *for all finite sequences of points $\{\mathbf{x}_i\}_{1\leq i\leq n}$ in Ω and real numbers $\{\lambda_i\}_{1\leq i\leq n}$:*
$$\sum_{i,j=1}^{n}\lambda_i\lambda_j\kappa(\mathbf{x}_i,\mathbf{x}_j) \geq 0 \tag{2.28}$$
i.e. the similarity matrix $(\kappa(\mathbf{x}_i,\mathbf{x}_j))_{1\leq i,j\leq n}$ is positive semidefinite.

Definition 2.3.2 (Mercer kernel)**.** *A continuous positive semidefinite kernel is called a Mercer kernel.*

Definition 2.3.3 (Hilbert-Schmidt integral operator)**.** *Let $\kappa \in L^2(\Omega \times \Omega)$ be a square-integrable kernel. The Hilbert-Schmidt integral operator $\mathcal{L}_\kappa : L^2(\Omega) \to L^2(\Omega)$ associated to κ is the linear operator defined by:*

$$\forall \phi \in L^2(\Omega), \quad \forall \mathbf{x} \in \Omega, \quad \mathcal{L}_\kappa [\phi] (\mathbf{x}) = \int_\Omega \kappa(\mathbf{x}, \mathbf{x}') \phi(\mathbf{x}') d\mathbf{x}' \tag{2.29}$$

Theorem 2.3.4 (Mercer's theorem [59])**.** *Let $\kappa : \Omega \times \Omega \to \mathbb{R}$ be a Mercer kernel, whose Hilbert-Schmidt integral operator is denoted by $\mathcal{L}_\kappa$. The eigenvalues $\{\lambda_i\}_{1 \leq i \leq \infty}$ of $\mathcal{L}_\kappa$ are nonnegative and the corresponding eigenfunctions $\{e_i\}_{1 \leq i \leq \infty}$ form an orthonormal basis of $L^2(\Omega)$. The eigenfunctions associated to nonzero eigenvalues are continuous on Ω. In addition, the kernel satisfies the following equation:*

$$\forall (\mathbf{x}, \mathbf{x}') \in \Omega \times \Omega, \quad \kappa(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i e_i(\mathbf{x}) e_i(\mathbf{x}') \tag{2.30}$$

where the convergence of the above function series is absolute and uniform.

Mercer's theorem has two interesting applications used in this thesis, namely the Karhunen-Loève expansion (or decomposition) and the kernel trick. Before introducing the Karhunen-Loève expansion, let us define the concept of a *random field*:

Definition 2.3.5 (Random field)**.** *A random field on $\Omega \subset \mathbb{R}^{\mathcal{N}}$ is an application $f : \Omega \times \Theta \to \mathbb{R}$ such that, for all $\mathbf{x} \in \Omega$, $f(\mathbf{x}, .) : \Theta \to \mathbb{R}$ is a real-valued random variable.*

Random fields are also known as *random functions*, or *stochastic processes* when $\mathcal{N} = 1$. A *sample path* or *realization* of a random field is an application $f(., \theta) : \Omega \to \mathbb{R}$ obtained for a given $\theta \in \Theta$. The expectation of a random field is an application $\mu : \Omega \to \mathbb{R}$ whose value at $\mathbf{x} \in \Omega$ corresponds to the expectation of $f(\mathbf{x}, .)$. The *(auto)-covariance function* is defined by:

$$\Gamma(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}(f(\mathbf{x}, .), f(\mathbf{x}', .)) \tag{2.31}$$

and the *variance function* is given by $\sigma^2(\mathbf{x}) = \Gamma(\mathbf{x}, \mathbf{x})$. The *(auto)-correlation function* reads:

$$\rho(\mathbf{x}, \mathbf{x}') = \mathrm{Corr}(f(\mathbf{x}, .), f(\mathbf{x}', .)) = \frac{\Gamma(\mathbf{x}, \mathbf{x}')}{\sigma(\mathbf{x}) \sigma(\mathbf{x}')} \tag{2.32}$$

Theorem 2.3.6 (Karhunen-Loève expansion [60, 61])**.** *Let $f : \Omega \times \Theta \to \mathbb{R}$ be a square-integrable random field with continuous covariance function Γ. Then, the covariance function Γ is a Mercer kernel and the random field f reads:*

$$\forall \mathbf{x} \in \Omega, \qquad f(\mathbf{x}, .) = \mathbb{E}[f(\mathbf{x}, .)] + \sum_{i=1}^{\infty} \Upsilon_i \sqrt{\lambda_i} e_i(\mathbf{x}) \tag{2.33}$$

where $\{\lambda_i\}_{1 \leq i \leq \infty}$ and $\{e_i\}_{1 \leq i \leq \infty}$ are respectively the eigenvalues and eigenfunctions of the Hilbert-Schmidt integral operator $\mathcal{L}_\Gamma$ associated to the covariance function Γ, and where $\{\Upsilon_i\}_{1 \leq i \leq \infty}$ are centered mutually uncorrelated random variables with unit variance defined by considering the orthogonal projection in $L^2(\Omega)$ onto the eigenfunctions:

$$\forall i \in \mathbb{N}^*, \qquad \Upsilon_i = \frac{1}{\sqrt{\lambda_i}} \int_\Omega \left(f(\mathbf{x}, .) - \mathbb{E}[f(\mathbf{x}, .)] \right) e_i(\mathbf{x}) \, d\mathbf{x} \tag{2.34}$$

The convergence in Equation (2.33) is in $L^2(\Theta)$ and uniform in Ω.

The Principal Component Analysis (PCA) and the Proper Orthogonal Decomposition (POD) are directly related to the Karhunen-Loève expansion theorem.

2.3.2 Application to random field simulation

The Karhunen-Loève expansion can be used for random field simulation using Gaussian random fields:

Definition 2.3.7 (Gaussian random field [62]). *A Gaussian random field* $f : \Omega \times \Theta \to \mathbb{R}$ *is a random field such that the random vector* $(f(\mathbf{x}_1, .), ..., f(\mathbf{x}_p, .))$ *follows a multivariate normal distribution for any choice of* p *and* $(\mathbf{x}_1, ..., \mathbf{x}_p) \in \Omega^p$.

When considering the Karhunen-Loève expansion of a Gaussian random field, the random variables Υ_i appearing in the decomposition have the additional properties of being independent standard normal random variables [60]. Therefore, a Gaussian random field is entirely defined by its expectation and its covariance function. It can be constructed by defining a mean field μ for the expectation and a Mercer kernel for the covariance function Γ. The first eigenfunctions of the Hilbert-Schmidt integral operator $\mathcal{L}_\Gamma$ corresponding to the largest eigenvalues can be combined with independent standard normal random variables to form a truncated version of the Karhunen-Loève expansion given in Equation (2.33). The covariance function is usually defined as an isotropic function, that is, a function that depends only on the distance $d(\mathbf{x}, \mathbf{x}')$ between the arguments $\mathbf{x}$ and $\mathbf{x}'$. A common example of isotropic covariance function is the *exponential covariance function*:

$$\Gamma(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\left(\frac{d(\mathbf{x}, \mathbf{x}')}{d_0}\right)^\nu\right) \tag{2.35}$$

where d_0 is the *correlation length* and $0 < \nu \leq 2$.

When constructing a random field, one may want it to have some regularity properties, such as continuous realizations with probability one, *i.e.* the probability to draw a continuous field is 1. This is the concept of *sample path continuity*:

Definition 2.3.8 (Sample path continuity [62]). *A random field* $f : \Omega \times \Theta \to \mathbb{R}$ *has continuous realizations (or sample paths) with probability one if* $\mathbb{P}(\mathcal{E}) = 1$ *where* $\mathcal{E}$ *is the event:*

$$\mathcal{E} = \left\{\theta \in \Theta \mid \ \forall \mathbf{x} \in \Omega, \ \lim_{n \to +\infty} \|\mathbf{x}_n - \mathbf{x}\| = 0 \ \Rightarrow \lim_{n \to +\infty} |f(\mathbf{x}_n, \theta) - f(\mathbf{x}, \theta)| = 0\right\} \tag{2.36}$$

The paper [62] gives several conditions under which a random field has continuous realizations with probability one, and makes the conjecture that Gaussian random fields with continuous expectation functions and continuous covariance functions have continuous realizations with probability one (see Conjecture 2.1, page 20 in [62]).

In practice, random fields are discretized on a 2D or 3D point cloud. In this case, the eigenvalue problem of the Hilbert-Schmidt integral operator $\mathcal{L}_\Gamma$ associated to the covariance function Γ is simply replaced by the eigenvalue problem of the covariance matrix $\mathbf{\Gamma}$ defined by $\Gamma_{ij} = \Gamma(\mathbf{x}_i, \mathbf{x}_j)$ where $\mathbf{x}_i$ is the vector containing the coordinates of the i-th sampling point. In this work, the domain Ω is sampled by the nodes of a finite-element mesh.

2.3.3 The kernel trick

The following property is a corollary of Mercer's theorem:

Property 2.3.9 (Representation of Mercer kernels as inner products)**.** *Let Ω be a compact space and let $\kappa : \Omega \times \Omega \to \mathbb{R}$ be a Mercer kernel. Let $\{\lambda_i\}_{1 \leq i \leq \infty}$ and $\{e_i\}_{1 \leq i \leq \infty}$ be the eigenvalues and eigenfunctions of the Hilbert-Schmidt integral operator associated to κ. Let n be defined by:*

$$n = \sup \{i \in \mathbb{N} \mid \lambda_i > 0\} \tag{2.37}$$

The mapping $\varphi : \Omega \to \mathbb{R}^n$ defined by:

$$\forall \mathbf{x} \in \Omega, \ \forall i \in [\![1; n]\!], \quad \varphi_i(\mathbf{x}) = \sqrt{\lambda_i} e_i(\mathbf{x}) \tag{2.38}$$

is continuous and satisfies:

$$\forall (\mathbf{x}, \mathbf{x}') \in \Omega \times \Omega, \quad \kappa(\mathbf{x}, \mathbf{x}') = \left\langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \right\rangle_{l^2} \tag{2.39}$$

where $\langle ., . \rangle_{l^2}$ is the l^2 inner product on $\mathbb{R}^n$.

Remark 2.3.10. *A more general result, known as the Moore-Aronszajn theorem, can be found in [63, 64].*

This property states that any Mercer kernel can be represented by an inner product, which is the origin of the *kernel trick* in kernel methods such as Kernel Ridge Regression, Kernel PCA, and Support Vector Machines (SVMs). Kernels methods include all the methods that only require evaluating inner products between samples. The kernel trick enables transforming linear kernel methods into nonlinear algorithms, by replacing the original inner products by a Mercer kernel. Let us give the example of a support vector classifier, classification algorithm that will be presented later in this thesis. When the classes are not linearly separable, one can apply a feature map φ to the data points in order to get a higher-dimensional representation of the dataset. The key idea is that nonlinear classification problems may be transformed into linear classification problems in higher-dimensional spaces. The transformed feature space $\varphi(\Omega)$ can even be infinite-dimensional, as long as it is an inner product space. The linear support vector classifier can then be applied to the transformed dataset $\{(\varphi(\mathbf{x}_i), y_i)\}_{1 \leq i \leq m}$ where y_i is the label for the i-th example $\mathbf{x}_i$, which simply consists in replacing the dot products $\mathbf{x}_i^T \mathbf{x}_j$ by the inner products $\langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_{l^2}$. However, computing inner products in a high-dimensional space is very expensive. Instead of explicitly choosing a feature map and evaluating these inner products, one can choose a Mercer kernel κ which implicitly defines a feature map φ and directly gives inner products $\langle \varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j) \rangle_{l^2} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$, according to Property 2.3.9. The feature map and the corresponding transformed feature space need not be known: they are implicitly defined via the kernel function, which provides an inexpensive way to evaluate the inner products. Common Mercer kernels include:

- the Gaussian RBF[1] kernel:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||_2^2}{2\sigma^2}\right) \tag{2.40}$$

[1] RBF: Radial Basis Function.

- the polynomial kernel:

$$\kappa(\mathbf{x}, \mathbf{x}') = \left(\gamma_0 + \gamma_1 \mathbf{x}^T \mathbf{x}'\right)^d \tag{2.41}$$

* *
*

Chapter 3

Uncertainty quantification in high-fidelity models

Abstract: *The quantities of interest monitored in numerical simulations depend on the environment of the physical system being modeled, which is usually not exactly known. In some cases, these uncertainties strongly influence simulation results. Uncertainty quantification is necessary to ensure the reliability of the physical system, but it requires calling the model multiple times. When the allowable number of model calls is limited because of its computational complexity, the configurations of the system and its environment for which simulations should be run can be cleverly chosen by design of numerical experiments methods. These concepts will be useful for the application of our methodologies to an industrial test case, where the computational complexity of the high-fidelity model is particularly high and motivates the construction of faster models from well-chosen simulations for uncertainty quantification purposes. This chapter also explains the concept of nonparametrized variabilities describing the type of uncertainties that are studied in this thesis.*

Contents

3.1 The concept of nonparametrized variabilities

Numerical simulations in computational physics involve influential parameters related to the environment of the physical system, its state, and its physical properties. Some of these parameters are uncertain, because of a lack of knowledge, modeling uncertainties, uncertainties due to manufacturing processes, or difficulties to measure some quantities in a physical experiment. They are therefore modeled as random variables. The random fluctuations of these parameters are called *variabilities* in this thesis.

Uncertainty propagation consists in modeling the probabilistic behavior of uncertain influential parameters in a model and propagating these uncertainties in the model to see how outputs (or quantities of interest) are affected. The ultimate goal is to estimate statistical properties of the outputs, such as the moments of their probability distributions (*e.g.* mean, variance), and the accuracy of these estimates can be quantified through confidence intervals. One may also want to calculate the probability for the output to take values larger than a given threshold (*failure probability*), or even to estimate the entire probability density function. Uncertainty quantification is essential for the design of reliable engineering systems, especially because sophisticated physics-based models are now applicable to industrial cases whereas it remains difficult to know the exact behavior of influential parameters related to the environment of the system. Problems with large variabilities on some parameters do not necessarily need uncertainty quantification analyses: these analyses are required when the outputs have a high sensitivity to the variabilities.

The input variabilities may belong to high-dimensional spaces, or even infinite dimensional spaces. This is the case in particular when a parameter corresponds to a function in a Hilbert space. When the model is parametrized by a random field discretized on a mesh, the dimension of the parameter space corresponds to the number of nodes in the mesh. Instead of considering the random field as a parameter, one can see it as a collection of scalar parameters corresponding to the values of the random field at every node in the mesh. However, the number of scalar parameters is then too large to be used efficiently. For instance, for problems with affine dependence with respect to some scalar parameters $X^1, ..., X^p$, *e.g.*:

$$\left(\mathbf{A}_0 + \sum_{k=1}^{p} X^k \mathbf{A}_k \right) \mathbf{u} = \mathbf{b}, \tag{3.1}$$

precomputing the matrices $\{\mathbf{A}_k\}_{0 \leq k \leq p}$ to accelerate the assembly of the operators involved in the problem is beneficial only if the dimension p of the parameter space is low. Correlations between these parameters may enable reducing the effective number of descriptive parameters, but in the general case, no low-dimensional parametrization of the variabilities is available. In this case, we say that the uncertain parameters have *nonparametrized variabilities* (or *generic* variabilities). Therefore, the word *(non)-parametrized* has two different meanings:

- The physics problems that are studied in this thesis are *parametrized* by uncertain influential parameters that can potentially be high-dimensional. The word *parameter* will be adopted to denote these influential parameters whose variabilities are taken as inputs in our methodology.
- The input variabilities are *nonparametrized* when no low-dimensional parametrization is available, or, in other words, when the intrinsic number of independent scalar parameters is too large (say, greater than 100).

The concept of nonparametrized variabilities is introduced in [65], and was already studied in [43] for problems with very high dimensional parameter spaces, although not explicitly mentioned with this specific name. The paper [65] explains that a nonparametrized variability can be seen as a configuration of the system or a scenario, with no explicit parametrized description. The problem when dealing with nonparametrized variabilities is that the operators involved in the model must be assembled for every new simulation. The industrial application presented in this work is an advantageous specific case, although the parameter space is very high-dimensional. Indeed, it is a temperature-dependent problem in solid mechanics, where the variabilities on the temperature field do not require assembling new operators in the exploitation phase of the reduced-order models. The only operation that must be done when considering a new scenario is the evaluation of the new temperature field on the reduced-integration points used for hyper-reduction, which is quasi-instantaneous. These aspects will become clearer in the chapter dedicated to model order reduction. On the contrary, variabilities on pressure loads on the external surface of the physical system would require assembling the new operators for every scenario to take the boundary conditions into account.

In [66, 67], the authors distinguish two types of uncertainties, namely parameter uncertainties (those considered in the present thesis), and model-form uncertainties due to a lack of knowledge of the complex physical phenomena that are necessarily described by imperfect models. The latter has no parametrization, not even in a high dimensional space, which motivates the nonparametric approach developed in [66, 67] to take into account model-form uncertainties. In this sense, the uncertainties on the quantities of interest due to model-form uncertainties are nonparametrized. However, the focus of this thesis is on parameter uncertainties (that can be nonparametrized in the sense explained above), and not on model-form uncertainties.

3.2 Design of numerical experiments

Uncertainty quantification requires a large number of model calls, which may be prohibitive for some real-world problems when the model involves a partial differential equations solver like finite-element or finite-volume solvers. In this case, the model is evaluated on a small sample and the resulting information is used to build a surrogate model that is able to compute approximate outputs in a reduced computation time. The surrogate model can either be a regression model or a reduced-order model. Once trained, it can be evaluated on larger samples to quantify uncertainties on the outputs. Within this framework, the choice of training data for which the true model must be called is a central issue. Given the cost of calling the true model, training points should be well-selected so that they contain as much information as possible to build a predictive surrogate model. This problem is known as *design of computer (or numerical) experiments* [68]. Algorithms for design of experiments (DoE) have been extensively studied for computational problems as well as physical experiments. Monte Carlo sampling [69] is a random sampling technique that suffers from the clumping effect generating holes in the parameter space while some training points are too close to each other. For that reason, Monte Carlo sampling is not appropriate at all when the sample size is limited. To better fill the parameter space, the simplest approach is the factorial design [70] relying on a grid. Although appropriate for the calibration of a linear regression model (possibly with second-order interactions), the number of points in a full factorial experiment grows exponentially with the dimension of

the parameter space for a given number of levels per dimension, as mentioned later in the section dedicated to the curse of dimensionality. This problem is partially solved using fractional factorial design [71], but the resulting training set still has very bad projection properties. Generally speaking, a DoE is expected to have:

- good space-filling properties;
- good projection properties, in the sense that space-filling properties remain good after projections onto lower-dimensional subspaces of the parameter space.

In particular, if clumping appears after projection, then it means that the DoE is not robust to projections. Space-filling properties after projection are important especially for low-dimensional subspaces (dimensions 1 to 3) since, usually, the effective number of influential parameters is small with respect to the dimension of the parameter space and the most significant interaction orders remain low. To build a DoE that has nice space-filling properties, [72] defined two geometrical criteria called *minimax* and *maximin*. Minimax DoE consists in minimizing the maximum distance between a random query point and its closest neighbor among DoE points to avoid having large holes in the parameter space. Maximin DoE rather maximizes the minimum distance between DoE points. Space-filling properties can also be obtained with discrepancy criteria [68]. A discrepancy measure quantifies the deviation of the DoE with respect to the uniform distribution: a uniform filling corresponds to a lower discrepancy.

Concerning projection properties, Latin Hypercube Sampling (LHS, [73]) gives good space-filling properties in one-dimensional subspaces. A DoE on a grid is a LHS if and only if all of the axis-parallel hyperplanes containing a DoE point contain only one DoE point. On a 2D grid, this means that each sample is the only one in its row and column. A LHS can have poor global space-filling properties (*e.g.* samples only on the diagonal in 2D), but ensuring that a DoE is a LHS is a nice way to obtain robustness for projections in dimension 1. Maximin LHS [74] looks for a maximin design among LHS designs. The drawback is that it does not guarantee robustness for projections in subspaces of dimension higher than 1. In 2015, however, [75] introduced Maximum Projection design and Maximum Projection LHS, algorithms that maximize maximin space-filling properties on projections onto subspaces of any dimension, with more weight on lower dimensions.

The aforementioned methods can be slow for high-dimensional parameter spaces or when the number of points in the DoE is large. Quasi-Monte Carlo sampling (*e.g.* with a Sobol' sequence [76]) can be used to generate a fast but suboptimal DoE.

Remark 3.2.1. *A DoE generally tries to spread the samples uniformly. To get samples from a given probability distribution, one can use the inverse transformation method. If U is a random variable following the uniform distribution $\mathcal{U}([0;1])$ on $[0;1]$, and if F denotes the cumulative distribution function of a target distribution p, then the random variable $F^{-1}(U)$ follows the probability distribution p.*

Remark 3.2.2. *Since designs of experiments are not efficient in very high dimensional spaces, they require a low-dimensional parametrization of the variabilities of the uncertain parameters. For problems with nonparametrized variabilities, there is no choice but to generate a DoE for training data using an approximate low-dimensional parametrization. Nonetheless, this DoE can be used to train a machine learning model and/or a reduced-order model that ignores the low-dimensional parametrization and does not use*

it when being called, taking the true variabilities instead in order not to depend on that parametrization. In this way, the trained model can be applied to new nonparametrized variabilities.

3.3 Monte Carlo simulations

Let X denote a random parameter (*i.e.* real-valued random variable, random vector, stochastic process or random field) of the physics problem, and let Z be a quantity of interest whose randomness is induced by the variabilities of X. Although being a deterministic function of X, Z is simply seen as a random variable in this section, which enables forgetting the underlying uncertain parameter X. The focus of this section is on describing the stochastic behavior of Z. The notation Z is used instead of Y for quantities of interest in this thesis, because Y will denote an intermediate variable related to the choice of the reduced-order model for the computation of Z.

Statistical properties to be estimated are generally expressed with integrals. An integral can be seen as the expectation of a function of a random variable, and can then be estimated with the empirical mean (or sample mean) of the function evaluated at random points, which is the main idea behind Monte Carlo methods [77]. The convergence of the Monte Carlo quadrature is ensured by the Strong Law of Large Numbers:

Theorem 3.3.1 (Strong Law of Large Numbers)**.** *Let $(Z_i)_{i\in\mathbb{N}^*}$ be a sequence of independent and identically distributed integrable random variables with expectation μ. Given $n \in \mathbb{N}^*$, let $\overline{Z}_n$ denote the empirical mean defined by:*

$$\overline{Z}_n = \frac{1}{n}\sum_{i=1}^{n} Z_i \tag{3.2}$$

The empirical mean $\overline{Z}_n$ converges almost surely to the expectation μ, that is:

$$\mathbb{P}\left(\lim_{n\to+\infty} \overline{Z}_n = \mu\right) = 1 \tag{3.3}$$

The empirical mean is therefore a consistent estimator of the expectation. This estimator is unbiased, which means that its expected value is the parameter to be estimated. When estimating the variance, one commonly uses the unbiased sample variance that reads:

$$S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(Z_i - \overline{Z}_n\right)^2 \tag{3.4}$$

Monte Carlo methods have a convergence rate of $O(n^{-1/2})$, according to the Central Limit Theorem:

Theorem 3.3.2 (Central Limit Theorem)**.** *Let $(Z_i)_{i\in\mathbb{N}^*}$ be a sequence of independent and identically distributed square-integrable random variables with expectation μ and standard deviation $\sigma > 0$. Then, the random variable defined by:*

$$\frac{\sqrt{n}}{\sigma}\left(\overline{Z}_n - \mu\right) \tag{3.5}$$

converges in law to the standard normal distribution $\mathcal{N}(0,1)$[1].

This convergence rate is rather slow but it has the advantage of being independent of the dimension, which is the reason why Monte Carlo methods are widely used for numerical approximations of high-dimensional integrals. The number of samples n that is required to achieve a desired accuracy level depends on the standard deviation σ. Variance reduction methods such as importance sampling [77] modify the integrand to reduce this constant in order to accelerate convergence or reduce the error for a given n. Another way to accelerate convergence is to resort to quasi-Monte Carlo methods [77] that replace random sequences used in Monte Carlo by quasi-random (or low-discrepancy) sequences. These sequences are deterministic sequences giving a uniform repartition of the samples to avoid the clumping effect of random sequences. Quasi-random sequences can be generated using Sobol' sequences [76], for instance.

The Central Limit Theorem can be used to build asymptotic confidence intervals using confidence intervals of the standard normal distribution. Let ϕ_r denote the quantile of order r of the standard normal distribution $\mathcal{N}(0,1)$. For all $\alpha \in]0;1[$, the interval:

$$I_n = \left[\overline{Z}_n - \phi_{1-\frac{\alpha}{2}}\sqrt{S_n^2/n}; \overline{Z}_n + \phi_{1-\frac{\alpha}{2}}\sqrt{S_n^2/n}\right] \tag{3.6}$$

is an asymptotic confidence interval with confidence level $1-\alpha$ for the expectation μ, that is:

$$\lim_{n\to+\infty} \mathbb{P}(\mu \in I_n) = 1-\alpha \tag{3.7}$$

Monte Carlo simulations simply consist in drawing independent realizations of the random variable X and running the corresponding simulations. The resulting values taken by the output Z are then used to compute estimates of the expectation and the variance and asymptotic confidence intervals, using the Strong Law of Large Numbers and the Central Limit Theorem. As mentioned in the section dedicated to designs of experiments, Monte Carlo sampling is not appropriate when the model complexity limits the number of simulations. In this situation, a first DoE is built in order to generate training data. A surrogate model trained on these data is then used in Monte Carlo simulations to get a large enough number of simulations in a reasonable computation time for uncertainty quantification.

3.4 Uncertainty propagation example in nonlinear solid mechanics

This section provides a simple example in nonlinear solid mechanics, showing the importance of uncertainty propagation. Let us consider the solid body Ω shown on Figure 3.1 subjected to a displacement-controlled loading applied on $\mathcal{S}^u$. Assuming a quasi-static

[1] Pointwise convergence of the cumulative distribution function to the standard normal cumulative distribution function.

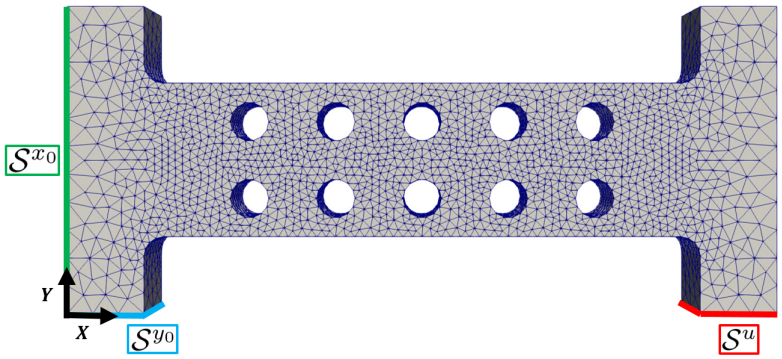

Figure 3.1: Finite-element mesh of the structure (33047 quadratic tetrahedral elements and 54649 nodes).

evolution, equilibrium equations and boundary conditions read:

$$\left\{\begin{array}{rcll} \mathbf{div}(\boldsymbol{\sigma}(\boldsymbol{\xi},t)) &=& \mathbf{0} & \forall t\in[0;1] \quad \forall \boldsymbol{\xi}\in\Omega \\ \mathbf{u}(\boldsymbol{\xi},t).\mathbf{e}_y &=& -u_d(t)\mathbf{e}_y & \forall t\in[0;1] \quad \forall \boldsymbol{\xi}\in\mathcal{S}^u \\ \mathbf{u}(\boldsymbol{\xi},t).\mathbf{e}_x &=& 0 & \forall t\in[0;1] \quad \forall \boldsymbol{\xi}\in\mathcal{S}^{x_0} \\ \mathbf{u}(\boldsymbol{\xi},t).\mathbf{e}_y &=& 0 & \forall t\in[0;1] \quad \forall \boldsymbol{\xi}\in\mathcal{S}^{y_0} \\ \mathbf{u}(\mathbf{0},t).\mathbf{e}_z &=& 0 & \forall t\in[0;1] \\ \boldsymbol{\sigma}(\boldsymbol{\xi},t).\mathbf{n}(\boldsymbol{\xi},t) &=& \mathbf{0} & \forall t\in[0;1] \quad \forall \boldsymbol{\xi}\in\partial\Omega\setminus(\mathcal{S}^u\cup\mathcal{S}^{x_0}\cup\mathcal{S}^{y_0}) \end{array}\right. \tag{3.8}$$

where $\mathbf{u}(\boldsymbol{\xi},t)$ is the displacement field (primal variable), $\boldsymbol{\sigma}(\boldsymbol{\xi},t)$ is the symmetric second-order Cauchy stress tensor, and $\mathbf{n}(\boldsymbol{\xi},t)$ is the outward-pointing normal vector to the outer surface of Ω. The imposed displacement $u_d(t)$ is defined by:

$$u_d(t) = \overline{u}_d(t) + X_t \tag{3.9}$$

where $\overline{u}_d(t)$ is the nominal (or reference) imposed displacement and X_t is a zero-mean Gaussian process defined by an exponential covariance function. This stochastic process models the uncertainties on the mechanical loading that usually exist in complex systems, for example uncertainties due to the vibrations of the neighboring parts. A realization of this stochastic process is given in Figure 3.2. This stochastic process defines non-parametrized variabilities. However, an approximate low-dimensional parametrization of these variabilities can be obtained by truncating the Karhunen-Loève expansion of this stochastic process. This strategy could be used for designs of experiments, for example. In this simple example, we simply run Monte Carlo simulations, so we do not need such parametrizations nor designs of experiments.

The structure is made of an elastoplastic generalized standard material described by the von Mises yield criterion and a nonlinear isotropic hardening law. In the framework of the infinitesimal strain theory, the constitutive equations are:

- Hooke's law (with an isotropic elastic stiffness tensor $\mathbb{C}$):

$$\boldsymbol{\sigma} = \mathbb{C} : (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p) \tag{3.10}$$

- von Mises yield criterion with isotropic hardening:

$$f(\boldsymbol{\sigma}, R) = \sigma_{\text{eq}}(\boldsymbol{\sigma}) - R - \sigma_y \qquad \sigma_{\text{eq}}(\boldsymbol{\sigma}) = \sqrt{\frac{3}{2}\mathbf{s}:\mathbf{s}} \qquad \mathbf{s} = \boldsymbol{\sigma} - \frac{1}{3}\text{tr}(\boldsymbol{\sigma})\mathbf{1} \tag{3.11}$$

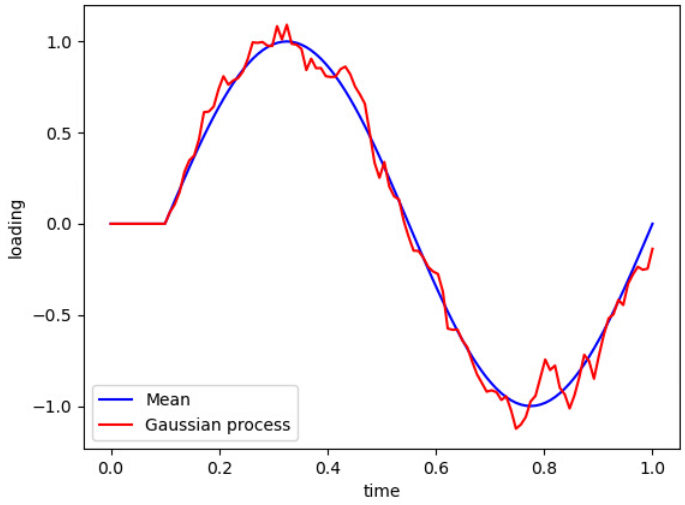

Figure 3.2: Imposed displacement $u_d(t)$ (in red) and nominal loading $\overline{u}_d(t)$.

- nonlinear isotropic hardening law (with p denoting the accumulated plastic strain):

$$R(p) = R_\infty(1 - \exp(-bp)) \tag{3.12}$$

- flow rule for the plastic strain rate tensor:

$$\dot{\boldsymbol{\varepsilon}}^p = \frac{3}{2}\dot{p}\frac{\mathbf{s}}{\sigma_{\text{eq}}(\boldsymbol{\sigma})} \tag{3.13}$$

- Karush-Kuhn-Tucker conditions:

$$\dot{p} \geq 0, \quad f \leq 0, \quad \dot{p}f = 0 \tag{3.14}$$

- Consistency condition for the determination of the plastic multiplier:

$$\dot{p}\dot{f} = 0 \tag{3.15}$$

The quantities of interest Z^1, Z^2 are the mean values of the accumulated plastic strains in two critical zones, namely a zone at the top of the first (from the left) top hole in the structure, and another at the bottom of the first bottom hole. The accumulated plastic strain can be formulated explicitly using Equation (3.13) and the definition of the von Mises stress $\sigma_{\text{eq}}(\boldsymbol{\sigma})$:

$$p(\boldsymbol{\xi}, t) = \int_0^t \sqrt{\frac{2}{3}\,\dot{\boldsymbol{\varepsilon}}^p(\boldsymbol{\xi}, \tau) : \dot{\boldsymbol{\varepsilon}}^p(\boldsymbol{\xi}, \tau)}\; d\tau \tag{3.16}$$

This internal variable is usually involved in ductile fracture criteria such as Rice and Tracey's [78], and is therefore directly associated to the damage state of ductile materials.

The mechanical behavior of the structure is simulated with Zset [79] finite-element software. 30 Monte Carlo simulations are run for different realizations of the stochastic process defining the imposed displacement $u_d(t)$. An example of accumulated plastic strain field in given in Figure 3.3 on the deformed geometry, where deformations have been voluntarily amplified for visualization purposes. Table 3.1 compares the expected

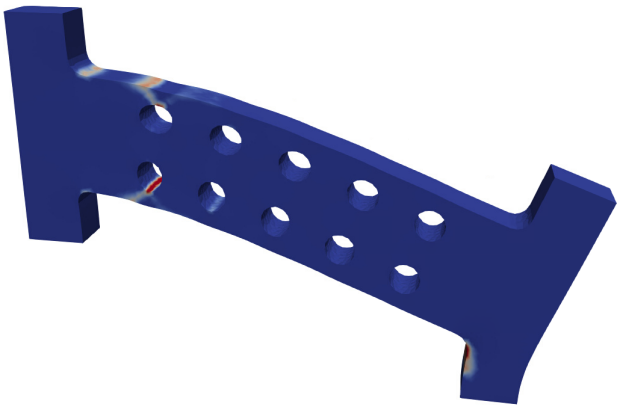

Figure 3.3: Accumulated plastic strain field p.

values of the mean accumulated plastic strains in the two critical zones with their values predicted when solving the mechanical problem without the random term X_t, *i.e.* with the nominal imposed displacement $\overline{u}_d(t)$ only. The bounds of the asymptotic 99% confidence intervals defined in Equation (3.6) are also given. It can be observed that the values of the accumulated plastic strain are underestimated when discarding the random term X_t. Because of nonlinearities, a simulation with the mean loading (nominal imposed displacement) does not give the mean damage level, which illustrates the importance of uncertainty quantification for a reliable design.

Table 3.1: Comparison of the expected values of the accumulated plastic strains averaged over two critical zones and the corresponding values predicted with the mean loading $\overline{u}_d(t)$.

Critical zone	Expected value	99% confidence interval	Nominal value
Lower zone	0.0101	$[0.0086; 0.0116]$	0.0066
Upper zone	0.0116	$[0.0100; 0.0132]$	0.0074

* *
*

Chapter 4

Unsupervised learning

Abstract: *Unsupervised learning deals with unlabeled data and consists in extracting information about their patterns, in the spirit of data mining tasks. It can be used in order to summarize information on a dataset: cluster analysis categorizes samples by finding groups of data sharing some characteristics, and dimensionality reduction models high-dimensional data with a reduced number of latent variables to get a compressed representation. In this thesis, clustering enables identifying groups of possible configurations of a physical system that can be simulated accurately with the same model. Dimensionality reduction is used for the construction of low-dimensional approximation spaces for model order reduction, but also for data preprocessing and visualization purposes.*

Contents

4.1 The curse of dimensionality

Generally speaking, machine learning algorithms are less efficient when the input data belong to a high-dimensional vector space. This issue is known as the *curse of dimensionality* [80]. As shown in [81], under some conditions, the mean relative difference between the smallest and the largest distances from a random query point to points of a high-dimensional dataset becomes negligible as the dimension of the ambient space increases. Therefore, all the concepts and methods relying on a distance (such as nearest neighbor, some clustering algorithms, etc...) become meaningless in high dimension. Of course, in many applications, high-dimensional data have a low intrinsic dimension, meaning that the data points lie in a low-dimensional manifold. In this case, there is no curse of dimensionality. However, in more general cases, working with high-dimensional data is difficult since many more training examples are required to fill a bounded region of the ambient space. Let us give the example of a function defined on a hypercube of dimension d, and let us say that we would like to know the values of this function on a grid with n points per direction. The total number of points in this grid equals to n^d, which becomes too large when d increases. As a result, real datasets in high dimensions cannot give enough information about the influence of variations along each direction of the ambient space.

Another common illustration of the curse of dimensionality is that most of the volume of a hypercube is located around its corners. Indeed, the ratio between the volume of the inscribed hypersphere and the volume of the hypercube tends towards zero when the dimension d increases.

In particular, the curse of dimensionality can be observed in high-dimensional supervised learning tasks where models overfit the training data, and in high-dimensional clustering, where one may find meaningless clusters.

4.2 Dimensionality reduction

Dimensionality reduction consist in finding a compressed representation of a dataset while limiting as much as possible the loss of information. It is particularly useful for the following applications:

- Data visualization;
- Extraction of the hidden structure of data, *i.e.* discovering the main characteristics of the manifold containing the data to get their most salient features;
- Storage of large databases when only the most salient features matter;
- Addressing the curse of dimensionality for high-dimensional supervised learning problems (regression or classification);
- Addressing the curse of dimensionality for high-dimensional cluster analysis by looking for clusters from the low-dimensional representation of the dataset;
- Accelerating numerical simulations (see Model Order Reduction).

The Principal Component Analysis (PCA) belongs to linear dimensionality reduction techniques. It consists in finding a low-dimensional affine subspace minimizing the projection error between the projected data and the original data. The principal axes are determined by solving an eigenvalue problem on the correlation matrix computed after centering the data. The corresponding eigenvalues indicate the variance that is captured, which enables ordering the principal axes and choosing their appropriate number to represent the dataset with a given level of accuracy.

When data lie in a nonlinear manifold, it is sometimes advantageous to apply a nonlinear dimensionality reduction method. Indeed, such methods can unveil latent coordinates that are more adapted to the nonlinear shape of the manifold. For example, when high-dimensional data points form a spiral embedded in a 2D subspace, the radius and the angle might be more relevant than two coordinates associated to a basis of this 2D subspace for the description of the data. Kernel PCA is a nonlinear extension of PCA using the kernel trick. Instead of solving the eigenvalue problem for the (linear) correlation matrix, one considers the eigenvalue problem of a Gram matrix corresponding to an inner product in an unknown higher-dimensional space computed via a Mercer kernel. Theoretically, this is equivalent to mapping the data to a higher-dimensional space and then applying PCA. Once again, the eigenvalues enable selecting an appropriate number of latent coordinates that well represent the data.

More generally, dimensionality reduction can be performed by undercomplete autoencoders. Autoencoders are defined by an encoder $e : \mathbb{R}^{\mathcal{N}} \to \mathbb{R}^n$ and a decoder $d : \mathbb{R}^n \to \mathbb{R}^{\mathcal{N}}$ that are combined to give the autoencoder function $f = d \circ e$. The autoencoder is said to be undercomplete when the dimension n of the latent space is lower than the original dimension $\mathcal{N}$. Intuitively, the encoder compresses the data into the low-dimensional latent space, and the decoder tries to retrieve the original data from the latent coordinates. Autoencoders are usually obtained by training a neural network with this autoencoder architecture, using the mean squared error loss to penalize the reconstruction error. When trained with the mean squared error loss and one single hidden layer with linear activation functions, the autoencoder neural network learns the PCA. Many autoencoder variants exist and can be found in [3, 4]. Using an autoencoder for dimensionality reduction purposes ensures that the low-dimensional representation of the data in the latent space is relevant, in the sense that we know that the decoder can reconstruct the original data from this reduced information.

4.3 Cluster analysis

4.3.1 Clustering algorithms

Cluster analysis belongs to unsupervised learning tasks in data science and has a broad range of applications in data mining, such as image segmentation for medical image computing and object detection, gene expression analysis in bioinformatics, anomaly detection, market segmentation, and community discovery in social network analysis. Cluster analysis is the search of groups (or *clusters*) of similar objects in a database. The choice of the clustering algorithm depends on the underlying motivation and thus on the clusters' topological properties that are expected. One may want to favor compactness or intra-cluster cohesion, between-cluster separation, connectivity, or cluster density. A clustering algorithm is either hierarchical or partitional. Hierarchical cluster analysis aims at finding

a nested series of partitions represented in a dendrogram, obtained by merging or splitting clusters. Hierarchical clustering algorithms include bottom-up agglomerative methods and top-down divisive methods. Many variants exist depending on the definition of the dissimilarity measure between data points and on the linkage criterion quantifying the similarity between two clusters (*e.g.* single linkage, average linkage, complete linkage, Ward). Partitional cluster analysis only finds one partition of the database, with the number of clusters being fixed by the user or automatically chosen according to the structure of the database. Representative-based algorithms use a dissimilarity measure to assign each object of the database to the cluster corresponding to the closest representative object. They include k-means [82], k-medians [83], k-medoids [84], and many variants of k-means. Probabilistic model-based algorithms assume that the data are generated by a mixture model such as the Gaussian mixture model (section 3.2.2 of [39]) and use the expectation-maximization algorithm [56, 55] to find the maximum likelihood estimates of the parameters of the mixture model. Density-based algorithms (*e.g.* DBSCAN [85]) define clusters as high-density zones. This strategy is also used by some grid-based algorithms such as CLIQUE [86] and STING [87] which discretize each feature of the data into a finite number of cells. Graph-based algorithms can detect arbitrarily-shaped clusters by partitioning a similarity graph. They include spectral clustering methods [88] such as NCuts [89]. Finally, deep clustering methods resort to deep learning to find latent representations of the data with features that facilitate clustering, using various network architectures such as multilayer perceptrons, convolutional neural networks, autoencoders, variational autoencoders, generative adversarial networks, and deep belief networks. Reviews and taxonomies of deep clustering can be found in [90, 91]. We refer the reader to the books [39], [83] (chapters 6 and 7), [55] (section 14.3), or articles [92, 93] for more details about clustering algorithms.

4.3.2 K-medoids clustering

In representative-based clustering algorithms, each cluster is associated to a partitioning representative, *i.e.* a reference point that well represents the cluster's members. Representative-based algorithms generally define the clusters thanks to the Voronoi diagram generated by the representatives, which gives clusters with high cohesion. They rely on a dissimilarity measure quantifying the difference between two points in the dataset.

Definition 4.3.1 (Dissimilarity measure)**.** *Let $\mathcal{X}$ be a topological space. A dissimilarity measure on $\mathcal{X}$ is a function $\delta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ such that $\delta(x, x') = \delta(x', x)$ for all $(x, x') \in \mathcal{X}^2$ and $\delta(x, x) = 0$ for all $x \in \mathcal{X}$.*

Definition 4.3.2 (Representative-based clustering)**.** *Let us consider a finite set $\{x_i\}_{1 \leq i \leq m}$ of elements of a topological space $\mathcal{X}$ endowed with a dissimilarity measure δ. For a given integer $K \in [\![2; m]\!]$, representative-based clustering consists in finding K representatives $\{\tilde{x}_k\}_{1 \leq k \leq K} \subset \mathcal{X}$ minimizing the objective function:*

$$\sum_{i=1}^{m} \min_{k \in [\![1;K]\!]} \delta(x_i, \tilde{x}_k) \tag{4.1}$$

The clusters C_k are given by:

$$C_k := \{x_i \mid \delta(x_i, \tilde{x}_k) \leq \delta(x_i, \tilde{x}_l) \ \forall l \in [\![1; K]\!]\} \tag{4.2}$$

When the dissimilarity measure is taken to be the squared Euclidean distance, the optimal representatives are the clusters' means or centroids (see [83], p.162). This problem corresponds to k-means clustering [82], where the cost function in Equation (4.1) corresponds to the within-cluster variance and is related to clusters inertia. Taking the L^1 or Manhattan distance instead as in k-medians would define the optimal representatives as the clusters' component-wise medians, when $\mathcal{X}$ is a finite-dimensional vector space (see [83], p.164). In k-medoids, the representatives must be taken among the elements of the dataset. This restriction is particularly useful when functions of the training examples (such as mean and median) do not make sense or cannot be easily computed. It enables working with any type of data with any dissimilarity measure. The next definitions introduce the k-medoids optimization problem:

Definition 4.3.3 (Binary matrices)**.** *A binary matrix is a matrix whose coefficients are either* 0 *or* 1*. The set of binary matrices of size* $m \times n$ *is denoted by* $\mathcal{B}_{m,n}$*.*

Definition 4.3.4 (K-medoids clustering)**.** *Let us consider a finite set* $\{x_i\}_{1\leq i\leq m}$ *of elements of a topological space* $\mathcal{X}$ *endowed with a dissimilarity measure* δ*. For a given integer* $K \in [\![2; m]\!]$*, let us introduce the set* $\mathcal{Z}_{m,K}$*:*

$$\mathcal{Z}_{m,K} := \left\{ \mathbf{Z} \in \mathcal{B}_{m,K} \mid \sum_{k=1}^{K} z_{ik} = 1 \ \forall i \in [\![1; m]\!] \text{ and } \sum_{i=1}^{m} z_{ik} \geq 1 \ \forall k \in [\![1; K]\!] \right\} \tag{4.3}$$

K-medoids clustering consists in solving the following optimization problem:

$$\mathbf{Z}^* := \underset{\mathbf{Z}\in\mathcal{Z}_{m,K}}{\arg\min} \sum_{k=1}^{K}\sum_{i=1}^{m} z_{ik}\delta(x_i, \tilde{x}_k) \tag{4.4}$$

where the medoids $\tilde{x}_k$ *are given by:*

$$\tilde{x}_k := \underset{j\in[\![1;m]\!]}{\arg\min} \sum_{i=1}^{m} z_{ik}\delta(x_i, x_j) \tag{4.5}$$

Using Equation (4.5)*, the optimization problem given in Equation* (4.4) *can be formulated as follows:*

$$\mathbf{Z}^* := \underset{\mathbf{Z}\in\mathcal{Z}_{m,K}}{\arg\min} \sum_{k=1}^{K} \min_{j\in[\![1;m]\!]} \sum_{i=1}^{m} z_{ik}\delta(x_i, x_j) \tag{4.6}$$

This formulation of the k-medoids problem has similarities with the k-means formulation proposed in [94]. With this formulation, the definition of the clusters C_k in Equation (4.2) is equivalent to:

$$C_k = \{x_i \mid z^*_{ik} = 1\} \tag{4.7}$$

Equation (4.3) defining the set $\mathcal{Z}_{m,K}$ ensures that each point is assigned to one single cluster, and that each cluster contains at least one element. Equation (4.5) defines the medoid of a cluster as its most central member. K-medoids is a combinatorial optimization problem, for which several heuristic approaches have been proposed to find a suboptimal solution at lower cost. The Partitioning Around Medoids (PAM [84] and Chap. 2 of [95]) is the most known algorithm. It iteratively looks for the best swap between nonmedoid points and medoids. Clustering Large Applications (CLARA [96] and Chap. 3 of [95])

applies PAM on different subsamples to reduce the computational complexity of PAM. As explained in Section 11.2.1 of [39], both PAM and CLARA algorithms can be interpreted as graph-searching problems: PAM explores the entire graph of clustering solutions, while CLARA explores a subgraph only. Clustering Large Applications based on Randomized Sampling (CLARANS [97, 98]) only considers a sample of the neighbors of the current graph node at each iteration, which enables searching over the entire graph as in PAM but at lower cost. These three algorithms have been improved recently in [99] in terms of computational complexity. Apart from these approaches, a simple and fast k-medoids algorithm has been proposed by Park and Jun in [100] following the standard implementation of k-means with a Voronoi iteration approach, *i.e.* alternating between a cluster assignment step and updating the medoids with Equation (4.5). Park and Jun's algorithm has a per-iteration complexity of $O(mK)$, whereas PAM has a per-iteration complexity of $O(K(m-K)^2)$, see [101]. However, as explained in [99], this algorithm does not explore as many configurations as PAM does. For small datasets (*i.e.* m in the order of 10^2 to 10^3), it is worth using PAM to find the best configuration. Running PAM on such datasets takes less than a minute, which remains negligible in comparison with other computations involved in our methodology.

Remark 4.3.5. *For all these variants of k-medoids clustering, the dissimilarities $\delta(x_i, x_j)$ are precomputed before looking for clusters.*

* *
*

Chapter 5

Supervised learning

Abstract: *Supervised learning algorithms consist in building predictive models from a training set of input-output examples. They can generally be formulated as an optimization problem, using the empirical risk minimization principle. There are two categories of supervised learning algorithms, namely regression and classification algorithms. Regression is used in this thesis for the reconstruction of full fields from hyper-reduced predictions on reduced-integration domains, and classification plays a major role in our methodology for the recommendation of suitable approximation spaces when solving partial differential equations. Ensemble learning methods enable limiting the risk of overfitting by leveraging different predictive models, which is particularly important when working with small training sets of simulation data.*

Remark 5.0.1. *This chapter includes a few paragraphs taken from our paper [48], with modifications.*

Contents

5.1 Empirical risk minimization

Supervised learning is the task of learning the correspondence between input data X and outputs Y from a training set of input-output pairs $\{(x_i, y_i)\}_{1 \leq i \leq m}$. Supervised machine learning problems fall into two categories: regression problems, for which the outputs take continuous values, and classification problems, consisting in the prediction of categorical labels. In this work, it is assumed that X is a continuous multivariate random variable having a probability density function $p_X : \mathcal{X} \to \mathbb{R}_+$, with $\mathcal{X} \subset \mathbb{R}^{\mathcal{N}}$ where $\mathcal{N}$ is the number of features. In single-label classification problems, the random variable Y is discrete and follows a categorical distribution (or multinoulli distribution) whose probability mass function is defined by

$$\forall y \in \mathbb{R}, \quad p_Y(y) = \sum_{k=1}^{K} \mathbb{P}_Y(k)\delta(y-k) \tag{5.1}$$

where K is the number of categories (or classes), δ is the Dirac delta function, and $\mathbb{P}_Y(k)$ denotes the probability of the event $Y = k$ for a given label $k \in [\![1; K]\!]$. The labeled training data are drawn from the joint probability distribution $p_{X,Y}$, called the *data-generating distribution*. As X is continuous and Y is discrete, $p_{X,Y}$ is a mixed joint density and can be obtained with the formula

$$p_{X,Y}(x, y) = p_Y(y)\ p_{X|Y}(x \mid y) = \sum_{k=1}^{K} \mathbb{P}_Y(k)\delta(y-k)p_{X|Y}(x \mid y) \tag{5.2}$$

with $p_{X|Y}$ being the class-conditional probability distribution.

In the present chapter, we are interested in single-label multiclass problems. Therefore, the classification problem considered here reads: *given an integer $K \geq 2$ and a training set $\{(x_i, y_i)\}_{1 \leq i \leq m} \subset \mathcal{X} \times [\![1; K]\!]$, train a classifier $\mathcal{C}(\ .\ ; \theta) : \mathcal{X} \to [\![1; K]\!]$ to assign any observation $x \in \mathcal{X}$ to the correct class, with θ denoting the parameters of the classifier.* However, reaching the highest possible accuracy on the training set is not the objective to be pursued, as it usually leads to *overfitting*. Indeed, the classifier is supposed to be applied to new unseen data, or *test* data, after the training phase. Therefore, the generalization ability of the classifier is at least as important as its performance on the training set. A classifier with high capacity[1] perfectly fits training data but is very sensitive to noise, leading to high test error and thus overfitting. On the other hand, a classifier with low capacity can produce smaller error gaps between training and test predictions, but such a classifier may not be able to fit the data, which is called *underfitting*. This dilemma is known as the *bias-variance trade-off*: low model capacity leads to high bias, while high model capacity leads to high variance. An illustration of the concepts of overfitting and underfitting is given in Figure 5.1 for regression.

For a given observation $x \in \mathcal{X}$, probabilistic classification algorithms estimate the membership probabilities $\mathbb{P}_{\text{model}}(y \mid x; \theta)$ for each class $y \in [\![1; K]\!]$. The classifier $\mathcal{C}$ returns the index of the class with the highest membership probability:

$$\mathcal{C}(x; \theta) = \underset{y \in [\![1;K]\!]}{\arg\max} \left(\mathbb{P}_{\text{model}}(y \mid x; \theta)\right) \tag{5.3}$$

The parameters θ must be optimized to minimize the *expected risk* $\mathcal{J}(\theta)$ defined by

$$\mathcal{J}(\theta) = \mathbb{E}_{(X,Y)\sim p_{X,Y}}\left[L\left(\mathcal{C}(X; \theta), Y\right)\right] \tag{5.4}$$

[1] Ability to learn classes with complex boundaries, related to model complexity.

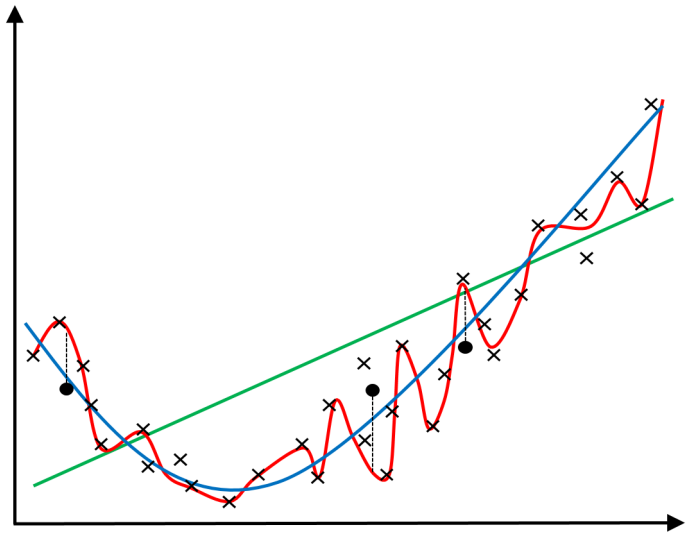

Figure 5.1: Illustration of overfitting (red curve) and underfitting (green curve) in a regression problem. The training points are represented by crosses, while test points are represented by dots.

where L is the per-example loss function quantifying the error between the predicted class $\mathcal{C}(X;\theta)$ and the true class Y. However, as the true data-generating distribution $p_{X,Y}$ is unknown, the expected risk must be estimated by computing the expectation with respect to the empirical distribution $\widehat{p}_{X,Y}$:

$$\widehat{p}_{X,Y}(x,y) = \frac{1}{m}\sum_{i=1}^{m}\delta(x-x_i, y-y_i) \tag{5.5}$$

Therefore, the training process consists in minimizing the *empirical risk*:

$$\widehat{\mathcal{J}}(\theta) = \mathbb{E}_{(X,Y)\sim\widehat{p}_{X,Y}}\left[L\left(\mathcal{C}(X;\theta),Y\right)\right] = \frac{1}{m}\sum_{i=1}^{m}L\left(\mathcal{C}(x_i;\theta),y_i\right) \tag{5.6}$$

This is known as the *empirical risk minimization* (ERM) principle [102]. Common choices for the function L are the hinge loss (defined for multiclass problems in [103]) used by support vector machines (SVMs), and the log loss or negative log-likelihood

$$L\left(\mathcal{C}(x;\theta),y\right) = -\log\ \mathbb{P}_{\text{model}}\left(y \mid x;\theta\right) \tag{5.7}$$

that is widely used for classifiers based on artificial neural networks (ANNs) and for logistic regression. When L is the negative log-likelihood, the objective function $\widehat{\mathcal{J}}(\theta)$ is the cross-entropy loss and the optimal set of parameters θ^* minimizing $\widehat{\mathcal{J}}$ is the maximum likelihood estimator [55]. Variants include the balanced cross-entropy loss to handle class imbalance, and the focal loss [104] which focuses more on misclassified examples. Usually, a regularization term is added to the empirical risk to penalize the model complexity in order to reduce overfitting.

For regression problems, the probability density function of Y is also denoted by p_Y, but Equation (5.1) is no longer valid. The concept of data-generating distribution $p_{X,Y}$ remains valid, although the second equality in Equation (5.2) no longer holds. The ERM principle

and the concept of overfitting are also valid, replacing the classifier $\mathcal{C}(\ .\ ;\theta) : \mathcal{X} \to [\![1; K]\!]$ by a regressor $\mathcal{R}(\ .\ ;\theta) : \mathcal{X} \to \mathbb{R}^K$, where $K = 1$ if Y is a real-valued random variable and $K > 1$ if Y is a K-dimensional random vector. For $K = 1$, the per-example loss function L is rather defined as the squared error $(\mathcal{R}(X;\theta) - Y)^2$ defining the *mean squared error* (MSE) loss, or as the absolute error $|\mathcal{R}(X;\theta) - Y|$ defining the *mean absolute error* (MAE) loss. The next sections introduce the main regression and classification algorithms. Most of the time, supervised learning algorithms have a version for classification and another for regression. Therefore, some algorithms that are presented in the section dedicated to classification have a counterpart for regression (*e.g.* for k-nearest neighbors algorithm, decision trees, support vector machines and artificial neural networks). The choice of presenting these algorithms as regression or classification algorithms is motivated by the way they are used in the test cases presented in this thesis.

5.2 Regression algorithms

5.2.1 Penalized linear regression and kriging

Let $X \in \mathbb{R}^{\mathcal{N}}$ and $Y \in \mathbb{R}^K$ be two multivariate random variables. Linear regression consists in modeling the output variable Y as a linear function of X, *i.e.* $Y \approx WX$ with $W \in \mathbb{R}^{K \times \mathcal{N}}$. An additional bias term can be considered to get an affine function, by adding a coordinate to the vector X and setting it to 1. The MSE loss reads:

$$\widehat{\mathcal{J}}(W) = \frac{1}{m} \sum_{i=1}^{m} ||y_i - W x_i||_2^2 \tag{5.8}$$

Linear regression can be easily extended to nonlinear problems by replacing X by nonlinear functions of X, such as polynomials for polynomial regression. The resulting regression problem is still called *linear regression*, it can be written as a new linear regression problem on a transformed input Z containing the nonlinear functions of the features of X.

When the dimension $\mathcal{N}$ of the input is high, the regression algorithm is prone to overfit training data. To avoid this issue, the model complexity is penalized by adding a regularization term $\Lambda(W)$ to the cost function:

$$W^* = \underset{W \in \mathbb{R}^{K \times \mathcal{N}}}{\arg\min}\ \widehat{\mathcal{J}}(W) + C^{-1}\Lambda(W) \tag{5.9}$$

with C being the inverse of the regularization strength, used to control the trade-off between the regularization term and the MSE loss. Common choices for the regularization term include:

- L^1 regularization, called Lasso:

$$\Lambda(W) = \sum_{i=1}^{K} \sum_{j=1}^{\mathcal{N}} |W_{ij}| \tag{5.10}$$

  which gives sparse coefficients, enabling the automatic selection of the most relevant features.

- L^2 regularization, also known as weight decay:

$$\Lambda(W) = \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1}^{\mathcal{N}} W_{ij}^2 = \frac{1}{2}||W||_F^2 \tag{5.11}$$

  which gives the Ridge Regression algorithm. When different weights are given to the coefficients of the matrix W in $\Lambda(W)$, this is known as Tikhonov regularization.

- Multi-task Lasso:

$$\Lambda(W) = \sum_{j=1}^{\mathcal{N}} \sqrt{\sum_{i=1}^{K} W_{ij}^2} \tag{5.12}$$

  which behaves like Lasso with the additional constraint that the selected features are the same for all the components of Y.

- Elastic net regularization [105], consisting in a weighted average of L^1 and L^2 regularization terms.

Kriging or *Gaussian process regression* [106, 107] is another famous algorithm. It has the advantage of giving confidence intervals for its predictions, which is the reason why kriging is widely used in adaptive refinement procedures, sensitivity analysis, and robust optimization, where it replaces a complicated function that is implicitly defined by a costly numerical simulation [108]. Different correlation functions are used depending on the regularity of the function to be approximated. Common kernels include Matérn kernels with parameter 3/2 for continuous and differentiable functions, and with parameter 5/2 for continuous and twice differentiable functions. For high-dimensional regression problems, Lasso can be applied to select a reduced number of input variables before training a Gaussian process regression model.

5.2.2 Hyperparameters tuning

The parameter C in Equation (5.9) and the other parameters that may be introduced in the regularization term such as the elastic net mixing coefficient are called *hyperparameters*. More generally, for any model and any learning task, a hyperparameter is a parameter that is not calibrated by the optimization algorithm used in the training procedure. These hyperparameters can be optimized using a validation set or with *cross-validation*, instead of being adjusted manually. When using a validation set, the user must train the model several times on the training set for different values of the hyperparameter, and then evaluate the trained models on the validation set. While the predictions on the training set are not accurate enough, new values of the hyperparameter must be tested. But using part of the available data as a validation set is not always appropriate: when working with a limited amount of data, the validation set might be too small to be representative, or too large to let enough examples in the training set. In this case, k-fold cross-validation can be used. It consists in splitting the training set into k subsets. Each possible value for the hyperparameter is tested by training the model k times on $k-1$ subsets and evaluating it on the remaining subset (playing the role of a validation set). The performance criterion is then averaged over the k evaluations, to get a performance measure that is independent of the choice of the validation set. The hyperparameter value giving the best cross-validated performance is kept and the model is trained again with this value for its hyperparameter,

this time on the whole training set. The concept of hyperparameter can be generalized to incorporate not only numerical parameters, but also algorithmic choices such as the optimizer, the kernel for SVMs, and the network architecture for neural networks.

5.3 Classification algorithms

In single-label multiclass classification, the boundaries between classes in the input space are called decision boundaries. Linear classifiers are classification algorithms for which the decision boundaries are defined by linear combinations of the features of X. Linear classifiers are appropriate when the classes are linearly separable in $\mathcal{X}$, which means that the decision boundaries correspond to portions of hyperplanes. Linear classifiers include logistic regression [109, 110, 111], linear discriminant analysis (LDA [55]), and the linear support vector classifier (linear SVM [112, 113]).

Many algorithms exist for nonlinear classification problems, each of them having its own advantages and drawbacks. As a kernel method, the linear SVM is extended to nonlinear classification problems using the *kernel trick* based on Mercer's theorem [59]. Artificial neural networks [114, 115] (see in [116] for a historical review) have become very popular due to their performances in numerous classification contests. Decision trees (e.g., CART algorithm [117]) and naive Bayes classifiers [118, 119] are well-known for their interpretability. Other nonlinear classifiers include the k-nearest neighbors algorithm (kNN [120]) and quadratic discriminant analysis (QDA [55]). In [121], the most common classifiers are compared on eleven binary classification problems. Short reviews of classification algorithms can be found in [122, 123]. This section introduces various classification algorithms that have been used in this thesis.

5.3.1 Generative classifiers

Following Bayes' theorem (Theorem 2.2.8), the posterior probability $\mathbb{P}_{Y|X}(y|x)$ is given by:

$$\mathbb{P}_{Y|X}(y|x) = \frac{p_{X|Y}(x|y)\mathbb{P}_Y(y)}{\sum_{k=1}^{K} \mathbb{P}_Y(k)p_{X|Y}(x|k)} \tag{5.13}$$

Generative classifiers estimate the class priors $\mathbb{P}_Y(k)$ from training data and model the class-conditional distributions $p_{X|Y}(x|k)$ in order to calculate the membership probabilities using Bayes' theorem for any input x. For a given observation x, the predicted label y corresponds to the class with the highest membership probability, which is equivalent to maximizing $p_{X|Y}(x|y)\mathbb{P}_Y(y)$. The decision rule of a generative classifier can then be seen as the maximum a posteriori (MAP) estimation of y, *i.e.* finding the parameter y such that x is a plausible observation drawn from the parametrized distribution $p_{X|Y}(x|y)$, with priors $\mathbb{P}_Y(y)$ on the possible values of y. During training, the class-conditional distributions $p_{X|Y}$ are modeled by classic probability distributions, whose parameters are calibrated using maximum likelihood estimation (MLE).

Quadratic discriminant analysis (QDA [55]) models the class-conditional distributions $p_{X|Y}(x|y)$ with multivariate normal distributions:

$$p_{X|Y}(x|y) = \frac{1}{(2\pi)^{\mathcal{N}/2}\det(\Sigma_y)^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_y)^T\Sigma_y^{-1}(x-\mu_y)\right) \tag{5.14}$$

where μ_y (resp. Σ_y) is the mean (resp. covariance matrix) for class y, estimated on the training set. This model gives quadratic decision boundaries. The Gaussian naive Bayes classifier [118, 119] is a special case of QDA, with all class covariance matrices Σ_y being diagonal. This constraint is referred to as the naive assumption, and amounts to assuming the class-conditional independence of the features X^i, *i.e.*:

$$p_{X|Y}(x|y) = \prod_{i=1}^{\mathcal{N}} p_{X^i|Y}(x^i|y) = \prod_{i=1}^{\mathcal{N}} \frac{1}{\sqrt{2\pi\sigma_{yi}^2}} \exp\left(-\frac{(x^i - \mu_{yi})^2}{2\sigma_{yi}^2}\right) \tag{5.15}$$

Linear discriminant analysis (LDA) is a linear probabilistic classifier obtained by adding the *homoscedasticity assumption* to QDA: all class covariance matrices are identical. This assumption leads to linear decision boundaries. The main advantage of QDA, LDA and Gaussian naive Bayes classifiers is that they have no hyperparameter.

5.3.2 Logistic regression

Definition 5.3.1 (Softmax function)**.** *Let $n \in \mathbb{N}^*$ be a positive integer. The softmax function* $\text{softmax} : \mathbb{R}^n \to \mathbb{R}^n$ *is defined by:*

$$\forall x = \left[x^1, ..., x^n\right] \in \mathbb{R}^n,\ \forall i \in [\![1; n]\!], \quad \text{softmax}_i(x) = \frac{\exp(x^i)}{\sum_{j=1}^n \exp(x^j)} \tag{5.16}$$

The softmax function is commonly used to compute membership probabilities from a vector of scores for multiclass problems. The name *softmax* comes from the fact that the softmax function can be seen as a smooth version of the one-hot encoded arg max function when one score s_k is large enough with respect to the other scores. Indeed, in this case, $\text{softmax}_k(s) \approx 1$ while the other terms are close to zero. The softmax function is used in logistic regression and neural network-based classifiers.

Logistic regression [109, 110] is a linear probabilistic classifier. Its multiclass version is known as the *multinomial logistic regression* [111]. The logarithms of the membership probabilities are modeled with a linear function of the input $x \in \mathbb{R}^{\mathcal{N}}$:

$$\log \mathbb{P}_{\text{model}}\left(k \mid x; W\right) = w_k^T x - \log x_0 \tag{5.17}$$

where $W \in \mathbb{R}^{K \times \mathcal{N}}$ is a matrix whose rows are denoted by $w_k \in \mathbb{R}^{\mathcal{N}}$, and with x_0 being defined as:

$$x_0 = \sum_{k=1}^{K} \exp(w_k^T x) \tag{5.18}$$

so that the sum of the membership probabilities equals to 1. These equations give:

$$\mathbb{P}_{\text{model}}\left(k \mid x; W\right) = \text{softmax}_k\left(Wx\right) \tag{5.19}$$

The parameters W are optimized by minimizing the cross-entropy loss. Usually, a regularization term is added to the cost function. According to [55, 124], LDA is more accurate than logistic regression when the homoscedasticity assumption is satisfied. Nevertheless, as this condition is rarely satisfied, logistic regression is more widely used.

5.3.3 k-nearest neighbors classifier

The k-nearest neighbors classifier [120] belongs to instance-based learning algorithms, which store training data in memory and compare test data with training examples to make predictions. The label for a given observation is obtained by a majority vote between the k nearest training examples in the sense of the Euclidean distance. In weighted nearest neighbors classifiers, the votes are weighted according to the distances from the new observation. A review of weighting schemes can be found in [125].

5.3.4 Tree-based classifiers

Decision trees are supervised learning algorithms modeling the correspondence between inputs and outputs by means of simple decision rules, leading to a recursive partition of the input space. They can handle both categorical and numerical data. Decision trees can solve regression problems (*regression trees*) or classification problems (*classification trees*), and are known to be interpretable in the sense that classification rules can be visualized as paths in a tree structure. Interpretability is a key advantage for applications where understanding the causes leading to a prediction is necessary. For such applications, tree-based algorithms are preferred over black-box models such as deep neural networks or kernel SVM. However, decision trees are generally unstable and usually overfit training data, which is the reason why they are often used in conjunction with an ensemble method such as random forest [126], for instance.

A decision tree is a rooted tree: it has a root node with no incoming edge, all of the other nodes having exactly one incoming edge. An example is given in Figure 5.2, where the root node compares the feature x_1 with a threshold value x_1^t. Nodes with outgoing edges are called internal nodes. Non-internal nodes are called leaves and contain a label. Leaves are represented by ellipses in Figure 5.2. Internal nodes perform a test on a feature of the input data, which splits the input space into two or more regions. Typically, for numerical features, a test consists in comparing a feature to a threshold value. This test corresponds to an internal node with two outgoing edges like in Figure 5.2, splitting the input space with a hyperplane orthogonal to the axis of the feature being tested. For a given observation x, the predicted label is obtained by following a path in the tree: starting from the root node, the path follows edges according to results of tests made by internal nodes, until it reaches a leaf whose label gives the predicted class.

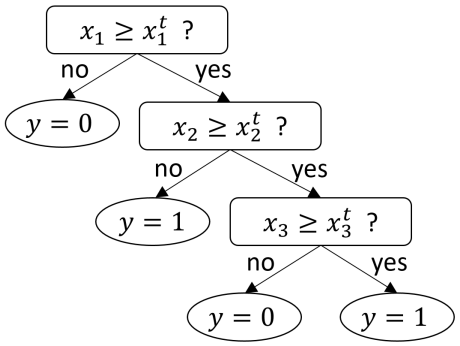

Figure 5.2: Decision tree for a binary classification problem.

Building an optimal decision tree for a given training set is a NP-hard problem. For this reason, many heuristic methods have been developed. Most of them follow a top-down recursive approach, like in CART [117]. This growing procedure is sometimes followed by a pruning procedure to reduce the complexity of the tree and avoid overfitting. The complexity of a tree can be measured via its depth, the number of nodes, or the number of leaves, for example.

5.3.5 Support vector classifiers

Support vector machines (SVMs [112, 113]) are supervised machine learning algorithms which can handle both classification and regression problems. This section focuses on support vector classifiers. A linear support vector classifier, or linear SVM, can solve classification problems where classes are linearly separable. The linear SVM is a non-probabilistic classifier, *i.e.* it does not compute membership probabilities to make decisions. SVMs are introduced in this section for binary classification problems. In this context, the labels y_i are redefined to take values in $\{-1; 1\}$.

Hard-margin classifier for linearly separable classes

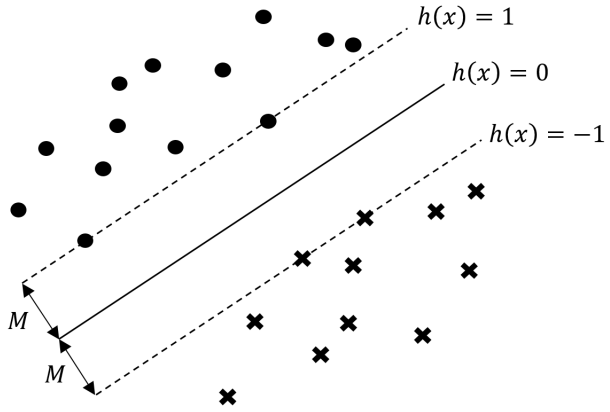

Figure 5.3: Decision boundary of a hard-margin classifier.

Before introducing the linear SVM and kernel SVM, let us introduce a simplified version of the linear SVM known as the *hard-margin classifier* [55]. When dealing with linearly separable classes, there is an infinite number of hyperplanes separating the two classes. For a given hyperplane defined by the affine equation $h(x) = 0$ with:

$$h(x) = w^T x + w_0 \tag{5.20}$$

and $(w, w_0) \in \mathbb{R}^{\mathcal{N}} \times \mathbb{R}$, the *margin* M is defined by:

$$M = \min_{x \in \mathbb{R}^{\mathcal{N}}} \min_{i \in [\![1;m]\!]} \{||x - x_i||_2,\ h(x) = 0\} = \min_{i \in [\![1;m]\!]} \frac{|h(x_i)|}{||w||_2} \tag{5.21}$$

The hyperplane maximizing the margin is taken as the decision boundary of the hard-margin classifier when the two classes are perfectly linearly separable, see Figure 5.3. The

labels 1 and -1 can be inverted if necessary so that this hyperplane satisfies $y_i h(x_i) > 0$ for all $i \in [\![1; m]\!]$. In addition, w and w_0 can be chosen so that:

$$\min_{i \in [\![1;m]\!]} y_i h(x_i) = 1 \tag{5.22}$$

which implies that $M = ||w||_2^{-1}$. Therefore, to find the hyperplane maximizing the margin, one must solve the following optimization problem:

$$\min_{(w,w_0) \in \mathbb{R}^{\mathcal{N}} \times \mathbb{R}} ||w||_2 \tag{5.23}$$

subject to:

$$y_i h(x_i) \geq 1 \quad \forall i \in [\![1; m]\!] \tag{5.24}$$

Soft-margin classifier

In general, the two classes are not perfectly linearly separable. Even if this is the case, the hard-margin classifier is too sensitive to outliers. This is why linear SVM relies on the concept of *soft margin*, allowing some points to be between the two hyperplanes $h(x) = 1$ and $h(x) = -1$. Given a hyperparameter $C > 0$, the primal formulation of the optimization problem solved by the linear SVM is:

$$\min_{(w,w_0,\zeta) \in \mathbb{R}^{\mathcal{N}} \times \mathbb{R} \times \mathbb{R}^m} \frac{1}{2} w^T w + C \sum_{k=1}^{m} \zeta_k \tag{5.25}$$

under the constraints:

$$\forall i \in [\![1; m]\!], \; y_i h(x_i) \geq 1 - \zeta_i \qquad \text{and} \qquad \zeta_i \geq 0 \tag{5.26}$$

The constraints on the slack variables ζ_i can be expressed with the hinge loss:

$$\zeta_i \geq \max(0; 1 - y_i h(x_i)) \tag{5.27}$$

which shows the link between Equation (5.25) and the empirical risk minimization of the hinge loss with a L^2 regularization on the weights w with regularization strength C^{-1}. One way to solve this optimization problem is to solve its Lagrangian dual with a quadratic programming algorithm, see [55]. Points located on the hyperplanes $h(x) = 1$ and $h(x) = -1$ are called *support vectors*. It can be shown that the solution w is a combination of the support vectors only [55].

Several extensions of SVMs to multiclass problems have been proposed, like in [103]. As a kernel method, the SVM classifier can be applied to nonlinear problems thanks to the *kernel trick*. According to [127], SVMs are popular for classification problems on small datasets and thanks to their strong theoretical background, contrasting with the development of neural networks based on extensive experimentation before coming to theoretical results. As said in [128], SVMs find the global minimum during training, while neural networks suffer from the existence of many local minima. For some applications, SVMs proved to be as competitive as or even more competitive than neural networks, see [127, 129, 130]. However, neural networks have become more popular during the last decade thanks to the rapid improvement of computer performances and to the extensive use of GPUs, enabling working with deep architectures to address complex problems.

5.3.6 Artificial neural networks

Network architecture

Artificial neural networks (ANNs) form a class of functions that can be represented by an architecture made of layers of neurons with an alternation between linear operations and nonlinear activation functions. The first ANN trained for a supervised learning task was the *perceptron*, invented by Rosenblatt in 1958 [131]. Later, hidden layers of neurons have been progressively added between the input and the output layers of ANNs to solve nonlinear problems, which led to the concepts of *deep* neural networks (DNNs) and *deep learning*. The earliest examples of DNNs in the 1960s can be found in [114, 115]. Since then, complex architectures have been explored for many different problems not limited to supervised learning tasks: recurrent neural networks (RNNs) for the processing of sequential data such as time series or texts (*e.g.* Long Short-Term Memory networks, LSTM), autoencoders for nonlinear dimensionality reduction, U-nets for image segmentation, variational autoencoders (VAEs) and generative adversarial networks (GANs) for the generation of synthetic data, etc... For detailed descriptions of the various deep learning architectures, we refer the reader to the books [3] and [4]. In this thesis, only feedforward architectures have been used. A historical review of artificial neural networks is proposed in [116]. With many outstanding results in international competitions in computer vision since 2009, deep learning has become very popular during the last decade with the development of very deep architectures trained on GPUs.

Feedforward neural networks

Feedforward neural networks are artificial neural networks whose connections between neurons go from the input layer to the output layer without forming loops in the hidden layers (see Figure 5.4). Formally, a feedforward neural network can be defined as follows:

$$\left\{ \begin{array}{rcll} x^{(0)} & = & x & \\ x^{(l)} & = & \Phi^{(l)}\left(x^{(l-1)}\right) & \forall l \in [\![1; L]\!] \end{array} \right. \tag{5.28}$$

where $L \in \mathbb{N}^*$ is the number of layers (excluding the input layer). If $n^{(l)}$ denotes the number of neurons in the l-th layer, then the functions $\Phi^{(l)}$ are defined on $\mathbb{R}^{n^{(l-1)}}$ and return vectors in $\mathbb{R}^{n^{(l)}}$. For a K-class classification problem, the output layer is generally defined using the softmax function to get membership probabilities:

$$\Phi^{(L)}\left(x^{(L-1)}\right) = \text{softmax}\left(W x^{(L-1)}\right) \tag{5.29}$$

with $W \in \mathbb{R}^{K \times n^{(L-1)}}$. No bias term is added before applying the softmax function, since softmax is translation-invariant.

Multilayer perceptrons

A *multilayer perceptron* (MLP) is a particular type of feedforward neural networks with fully-connected layers, meaning that the value computed at any neuron of any layer l can be a function of all the neurons of layer $l-1$. The most simple MLPs consist in stacking affine transformations and nonlinear activation functions. Additional functionalities may

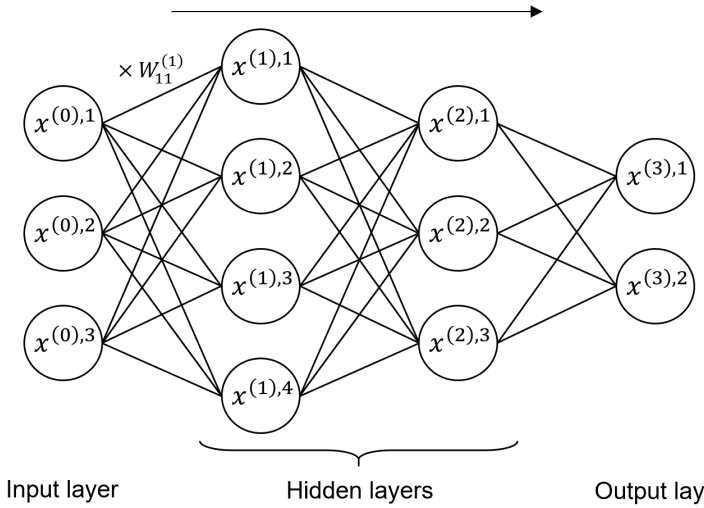

Figure 5.4: Example of feedforward neural network with fully-connected layers (multilayer perceptron). Each connection is weighted by a coefficient of the matrices $W^{(l)}$. Each neuron, represented by a circle, carries a scalar value. The contributions whose connections converge towards the same neuron are added and fed into a nonlinear activation function, optionally with an additional bias term.

be added in the architecture, such as dropout [132] and batch normalization [133] for example. Without these additional functionalities, a MLP can be defined by:

$$\Phi^{(l)}\left(x^{(l-1)}\right) = \varphi^{(l)}\left(W^{(l)}x^{(l-1)} + b^{(l)}\right) \tag{5.30}$$

with $W^{(l)} \in \mathbb{R}^{n^{(l)} \times n^{(l-1)}}$, $b^{(l)} \in \mathbb{R}^{n^{(l)}}$, and $\varphi^{(l)} : \mathbb{R}^{n^{(l)}} \to \mathbb{R}^{n^{(l)}}$ of the form:

$$\forall z = [z^1, ..., z^{n^{(l)}}]^T \in \mathbb{R}^{n^{(l)}},\ \varphi^{(l)}(z) = [\widehat{\varphi}^{(l)}(z^1), ..., \widehat{\varphi}^{(l)}(z^{n^{(l)}})]^T \tag{5.31}$$

except for $l = L$ when considering classification problems. In the equation above, $\widehat{\varphi}^{(l)}$ is an activation function, the most common choice being the ReLU function [4] $z \mapsto \max(0, z)$. An illustration of a simple MLP with $L = 3$ layers (2 hidden layers and an output layer) is given in Figure 5.4.

Remark 5.3.2. *A MLP classifier made of a single fully-connected layer ($L = 1$) with the softmax activation function for $\Phi^{(1)}$ and trained with the cross-entropy loss is equivalent to the multinomial logistic regression algorithm.*

Radial basis function networks

A *radial basis function network* (RBFN) is a feedforward neural network such that:

$$\Phi^{(l)}\left(x^{(l-1)}\right) = W^{(l)}\rho^{(l)}\left(x^{(l-1)}, c^{(l)}\right) \tag{5.32}$$

with $W^{(l)} \in \mathbb{R}^{n^{(l)} \times \tilde{n}^{(l)}}$. The function $\rho^{(l)} : \mathbb{R}^{n^{(l-1)}} \times \mathbb{R}^{n^{(l-1)}} \to \mathbb{R}^{\tilde{n}^{(l)}}$ is a vector function made of radial basis functions:

$$\left(\rho^{(l)}\left(\chi, c^{(l)}\right)\right)_i = \widehat{\rho}^{(l)}\left(||\chi - c^{(l)}[i]||_2\right) \tag{5.33}$$

with $c^{(l)}$ being the list of centers $c^{(l)}[i] \in \mathbb{R}^{n^{(l-1)}}$. A common choice for $\widehat{\rho}^{(l)}\left(||\chi - c^{(l)}[i]||_2\right)$ is the Gaussian RBF:

$$\widehat{\rho}^{(l)}\left(||\chi - c^{(l)}[i]||_2\right) = \exp\left(-\gamma||\chi - c^{(l)}[i]||_2^2\right) \tag{5.34}$$

The centers are usually determined with k-means clustering algorithm on the training data, while the coefficient γ for the Gaussian RBF function is given by heuristic methods. These parameters can also be optimized with gradient descent, like the weights $W^{(l)}$ of the fully-connected layers. Using radial basis functions instead of simple activation functions imply that it is generally not necessary to use many RBF layers. In fact, RBFN are usually shallow, with only one RBF layer and one or more fully-connected layers. The idea behind RBFN is similar to SVM: the RBF layer gives a richer representation of the input, on which the classification problem may be easier. As explained in [4], when trained with the hinge loss, RBFNs can be seen as generalized kernel SVMs.

Convolutional neural networks

Convolutional neural networks (CNNs) are commonly used to process images. The first CNN can be found in [134]. Common CNN architectures for classification tasks generally involve several blocks made of a convolution layer, batch normalization, an activation layer and a pooling layer, followed by fully-connected layers. 2D convolution filters are applied on images to compute feature maps (or channels) containing increasingly high-level features as we progress in the hidden convolution layers, while pooling layers reduce the amount of information to be analyzed in each feature map. Then, the feature maps of the last convolution layer are flattened to be fed into fully-connected layers that analyze the high-level features to predict the output. A CNN can be seen as a first purely-convolutional network processing useful representations of the input data, followed by a multilayer perceptron. The next paragraphs introduce a few functionalities of CNNs. For simplicity, the input is assumed to have a tensorial representation $x \in \mathbb{R}^{c \times n_1 \times n_2}$ like images, where c is the number of channels ($c = 3$ for colored images, *i.e.* one channel for each primary RGB color), and n_1 and n_2 are the number of pixels along the horizontal and vertical directions. In fact, in this thesis, CNNs will be applied to 3D fields defined on a mesh by projecting them onto a 3D grid with $n_1 \times n_2 \times n_3$ voxels. The following equations will therefore be used with an additional dimension to process data in $\mathbb{R}^{c \times n_1 \times n_2 \times n_3}$, where c will correspond to the number of components of the physical field.

Convolution operator. Let us consider tensors $x \in \mathbb{R}^{c \times n_1 \times n_2}$ and $w \in \mathbb{R}^{c_w \times c \times f_1 \times f_2}$ with $f_i \leq n_i$. The tensor $w \star x \in \mathbb{R}^{c_w \times m_1 \times m_2}$ with $m_i = n_i - f_i + 1$ is defined by:

$$(w \star x)[i,j,k] = \sum_{\alpha=1}^{c} \sum_{\beta=1}^{f_1} \sum_{\gamma=1}^{f_2} w[i,\alpha,\beta,\gamma]\, x[\alpha, j+\beta-1, k+\gamma-1] \tag{5.35}$$

Padding function. Let $p \in \mathbb{N}^2$. $\Phi_{\text{pad}}(.;p) : \mathbb{R}^{c \times n_1 \times n_2} \to \mathbb{R}^{c \times (n_1+2p_1) \times (n_2+2p_2)}$ is defined by:

$$\Phi_{\text{pad}}(x;p)[i,j,k] = \begin{cases} x[i, j-p_1, k-p_2] & \text{if } (j,k) \in [\![1+p_1; n_1+p_1]\!] \times [\![1+p_2; n_2+p_2]\!] \\ 0 & \text{else} \end{cases} \tag{5.36}$$

Stride function. Let $s \in \mathbb{N}^2$. $\Phi_{\text{stride}}(.;s) : \mathbb{R}^{c \times n_1 \times n_2} \to \mathbb{R}^{c \times m_1 \times m_2}$ with:

$$m_i = \left\lfloor \frac{n_i - 1}{s_i} \right\rfloor + 1 \tag{5.37}$$

is defined by:

$$\Phi_{\text{stride}}(x;s)\,[i,j,k] = x\,[i, 1 + (j-1)s_1, 1 + (k-1)s_2] \tag{5.38}$$

2D convolution layer. $\Phi_{\text{conv}}(.;W,b,p,s) : \mathbb{R}^{c \times n_1 \times n_2} \to \mathbb{R}^{c_W \times m_1 \times m_2}$ with:

$$m_i = \left\lfloor \frac{n_i + 2p_i - f_i}{s_i} \right\rfloor + 1 \tag{5.39}$$

and $p \in \mathbb{N}^2,\ s \in \mathbb{N}^2,\ b \in \mathbb{R}^{c_W \times m_1 \times m_2},\ W \in \mathbb{R}^{c_W \times c \times f_1 \times f_2}$. The integer c_W is the number of filters (or feature maps), defining the number of channels of the output. The 2D convolution layer is defined by:

$$\Phi_{\text{conv}}(x;W,b,p,s) = b + \Phi_{\text{stride}}\left(W \star \Phi_{\text{pad}}(x;p);s\right) \tag{5.40}$$

2D pooling layer. $\Phi_{\text{pool}}(.;\phi_{\text{pool}},f,p,s) : \mathbb{R}^{c \times n_1 \times n_2} \to \mathbb{R}^{c \times m_1 \times m_2}$ with:

$$m_i = \left\lfloor \frac{n_i + 2p_i - f_i}{s_i} \right\rfloor + 1 \tag{5.41}$$

is defined by:

$$\Phi_{\text{pool}}(x;\phi_{\text{pool}},f,p,s) = \Phi_{\text{stride}}\left(\phi_{\text{pool}}\left(\Phi_{\text{pad}}(x;p);f\right);s\right) \tag{5.42}$$

with $\phi_{\text{pool}}(.;f) : \mathbb{R}^{c \times n_1 \times n_2} \to \mathbb{R}^{c \times (n_1 - f_1 + 1) \times (n_2 - f_2 + 1)}$. Common pooling operations include:

- Max pooling:

$$\phi_{\text{pool}}^{\max}(x;f)\,[i,j,k] = \max_{\beta \in [\![1;f_1]\!]} \max_{\gamma \in [\![1;f_2]\!]} x\,[i, j+\beta-1, k+\gamma-1] \tag{5.43}$$

- Average pooling:

$$\phi_{\text{pool}}^{\text{avg}}(x;f)\,[h,i,j] = \frac{1}{f_1 f_2} \sum_{\beta=1}^{f_1} \sum_{\gamma=1}^{f_2} x\,[i, j+\beta-1, k+\gamma-1] \tag{5.44}$$

Training a deep neural network

Training a DNN for a supervised learning task requires optimizing its parameters θ in order to minimize the empirical risk. To do this, one needs to compute the gradient of the empirical risk:

$$\nabla_\theta \widehat{\mathcal{J}}(\theta) = \mathbb{E}_{(X,Y) \sim \widehat{p}_{X,Y}}\left[\nabla_\theta L\left(\mathcal{C}(X;\theta),Y\right)\right] = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L\left(\mathcal{C}(x_i;\theta),y_i\right) \tag{5.45}$$

Computing the expected value of the gradient of the per-example loss function is very expensive, because it requires calling the model and evaluating gradients on the whole

training set. Instead, the gradient $\nabla_\theta \widehat{\mathcal{J}}(\theta)$ can be approximated by evaluating the sum in Equation (5.45) on a *batch* containing a small number of randomly drawn training examples. This is the idea behind the *minibatch stochastic gradient descent* (SGD) optimization method, which updates the parameters θ with gradient descent using minibatch approximations of the gradient. One *epoch* corresponds to the multiple iterations that are made so that the DNN has seen the entire training set. For more details, see [3]. In practice, there exist other stochastic optimization methods that are more sophisticated than minibatch SGD, such as Adam [135] for instance. Independently from the choice of the optimizer, the gradients $\nabla_\theta L\left(\mathcal{C}(x_i;\theta), y_i\right)$ are computed with the *backpropagation algorithm* (see section 1.3 of [4]), a special case of *automatic differentiation* [23] (also known as *auto-diff* or *algorithmic differentiation*). Contrary to numerical differentiation (finite difference approximation) and symbolic differentiation (manipulating expressions), automatic differentiation computes efficiently the exact derivatives of a function thanks to the chain rule for the differentiation of composition functions, idea that is at the origin of differentiable programming.

5.4 Ensemble learning

Ensemble learning aims at creating a meta-estimator from several base estimators (or learners). Combining the predictions of different models generally results in more robust predictions and reduces overfitting. In addition, using an ensemble method replaces the task of finding a single very accurate model by the task of building an effective meta-estimator from several models with lower accuracies.

5.4.1 Voting and averaging

Generally speaking, a group of experts is more reliable than a single expert. A single expert can make mistakes, whereas a group of experts benefits from the opinion of each of his members to make a decision that is approved by the majority. This is the idea behind *voting ensembles*, where a *voting estimator* is built from several estimators. For classification tasks, the simplest method is based on majority vote (*hard voting*): the predicted class is the class that is most frequently predicted by the classifiers in the ensemble. Some classifiers can make mistakes, but their mistakes have a lower impact when combining predictions of different classifiers. A voting classifier is less sensitive to noise and outliers in training data because the vote tends to smooth the decision boundaries, which reduces overfitting. When forming an ensemble, it is important to train classifiers with different characteristics so that their weaknesses can be compensated by the strengths of the other classifiers. One can use different classification algorithms for example, but combining them does not automatically increase the performances on test data, especially when some classifiers are much less accurate than others. When training a neural network, it is common to recycle different networks that have been trained with different architectures and hyperparameters to gain a few percents of accuracy. This practice does not require more efforts than training a single neural network, since reaching a high accuracy generally requires trying many different architectures and hyperparameters settings. Using a voting classifier enables recycling the best models instead of discarding all of them except the one with the highest accuracy.

This idea can also be used for regression tasks, where the final prediction is obtained by averaging the outputs of the regressors in the ensemble. If the errors made by each classifier form a sequence of independent and identically distributed random variables with expectation 0, then their average converges almost surely to zero as the number of classifiers increases, according to the Strong Law of Large Numbers. Averaging predictions also works for classification tasks when using probabilistic classifiers. *Soft voting* consists in averaging the membership probabilities predicted by the probabilistic classifiers in the ensemble, and returning the class with the highest averaged membership probability. These techniques are known as *ensemble averaging* [136].

The winners of numerous deep learning challenges used ensemble averaging to improve their predictions (see for example [137, 138, 139]). We also noticed the benefits of ensemble averaging in [47], where using soft voting on an ensemble made of 12 deep neural networks with accuracies ranging from 63.05% to 73.75% enabled reaching an accuracy of 80% on test data.

5.4.2 Other ensemble methods

Bagging or *bootstrap aggregating* [140] consists in training several instances of a base learner on different training sets. These training sets, called bootstrap samples, are built from the original one by random sampling with replacement. Hence, some training examples may appear several times in a given bootstrap sample, or not appear at all. Training the same learner on different bootstrap samples gives an ensemble of models with different strengths. The predictions of these models are aggregated to improve the accuracy of the final prediction, either by a majority vote or by averaging membership probabilities. The generalization error of a bagging estimator can be estimated by the *out-of-bag error*. The out-of-bag error is evaluated by calculating the prediction error for each base estimator on the training examples that do not belong to the bootstrap sample of the estimator. The *random subspace method* [141], also known as *feature bagging*, works like classical bagging except that the training sets are obtained by randomly sampling the features instead of the training examples. This method is generally used when the number of features is larger than the number of samples. Random forests [126] combine bagging and feature bagging to build an ensemble of decision trees.

In the real world, groups of experts sometimes give a report of their discussions and conclusions to another person that has the responsibility to make the final decision. This person does not need to be an expert: he only needs to analyze the opinion of the group of experts and make a sensible decision in consequence. Similarly, in machine learning, the predictions of several pre-trained classifiers can be fed into another simpler classifier (*e.g.* logistic regression) that returns the final prediction. This is called *stacked generalization* or *stacking* ([142, 55]).

All the aforementioned methods use models trained independently. In contrast, *boosting methods* train different models sequentially. The most common boosting algorithm is AdaBoost (adaptive boosting [143, 144]). The algorithm iteratively trains a weak learner (typically a shallow decision tree) to build a boosted estimator whose prediction consists of a weighted average of the weak learner's predictions. During training, the $i+1$-th weak learner focuses more on training examples for which the i-th weak learner made mistakes. *Gradient boosting* [145, 146, 147, 148] generalizes boosting by interpreting it as an optimization problem with a specific loss function, and by applying gradient descent to an

arbitrary differentiable loss function to define the weak learners. When applied with decision trees as weak learners, gradient boosting algorithms are also called *gradient-boosted decision trees*.

5.5 Classification in computational physics

Classification problems can be encountered in various disciplines such as handwritten text recognition [149], document classification [150], and computer-aided diagnosis in the medical field [151], among many others. In numerical analysis, classification algorithms are getting increasingly more attention for the selection of efficient numerical models that can predict the behavior of a physical system with very different states or under various configurations of its environment [42, 43, 152, 153, 154, 155, 156]. In this case, the class labels are used to identify the models.

Applications to turbulence modeling in computational fluid dynamics can be found in [153, 156]. In large eddy simulations (LES; see in [157]), the Navier–Stokes equations are filtered to avoid resolving small-scale turbulent structures whose effects are taken into account either by sub-grid scale models (explicit LES closures) or via the dissipation induced by numerical schemes (implicit LES). In [153], sub-grid statistics obtained from direct numerical simulations enable training a fully-connected deep neural network to switch between different explicit LES closures at any point of the grid. This classifier is reused in [156], this time for switching between different numerical schemes in implicit LES. In both cases, the classifier is used to increase the accuracy of numerical predictions.

The idea of locally switching between different simulation strategies can also be found in [152] for the multiscale modeling of composite materials. In the multilevel finite-element method (FE^2 [158]), the quantities of interest at every integration point of the macroscopic finite-element mesh are given by a microscopic finite-element computation of an elementary cell representing the material's microstructure. The multi-fidelity surrogate model presented in [152] relies on two surrogate models replacing the microscopic finite-element model: a reduced-order model taken from [159] and an artificial neural network based regression model. At each integration point of the macroscopic mesh, the classifier (a fully-connected network) analyzes the effective strains and predicts whether the error of the regression model would be acceptable, enabling the selection of either the purely data-driven regression model or the more sophisticated physics-driven ROM. This time, automatic model recommendation by a classifier is used to adapt the model complexity and reduce the computation time.

In [154, 155], optimal classification trees (OCTs [160]) are used as *model selectors* in a data-driven physics-based digital twin of an unmanned aerial vehicle (UAV). The OCTs enable the update of the digital twin according to sensor data by selecting a model from a predefined model library. In this context, the training procedure for the classifier corresponds to an inverse problem. Indeed, training examples are generated by running simulations with all the models in the library and evaluating their predictions at the sensors' locations. Therefore, for a given model $y \in [\![1; K]\!]$, the data x are obtained by means of numerical simulations performed with y. This corresponds to the forward mapping. The classifier must learn the inverse mapping giving y as a function of x. In this example, data labeling is straightforward: the label of a training example x is given by the index y of the model which was used to generate x. It is also noteworthy

that generating training examples is not too expensive, because numerical simulations are performed with reduced-order models obtained by the Static-Condensation Reduced-Basis-Element method (SCRBE [161, 162, 163, 164]). In this application, automatic model recommendation gives the UAV the ability to dynamically evaluate its flight capability and replan its mission accordingly.

Another example of classifier used to accelerate numerical simulations can be found in [42]. Contrary to the works in [154, 155], the data labeling procedure relies on the clustering of simulation data. In this framework, the model library is made of cluster-specific Discrete Empirical Interpolation Method (DEIM) [165] models that are faster than the high-fidelity model. The high-fidelity model computes a prediction u_i for each input x_i in the database $\{x_i\}_{1\leq i\leq m}$, resulting in a dataset $\{u_i\}_{1\leq i\leq m}$ on which a clustering algorithm is applied. The predicted variable u is the discretization of a continuous field on a finite-element mesh, thus living in a high-dimensional space. To avoid the so-called *curse of dimensionality* [80], a DEIM-based feature selection technique is used before applying k-means clustering [82]. Alternatively, the clusters can be obtained with a variant of k-means using the DEIM residual as clustering criterion. Then, for a given training example x_i, the class label y_i is defined by the index of the cluster that u_i is assigned to. In the exploitation phase, when dealing with test data, the best DEIM model is selected by a nearest neighbor classifier. The input data given to the classifier are either parameters of the problem or the variable u obtained at the previous time increment. A similar methodology is described in [43], where the concept of model library is termed *model dictionary*, which is the terminology adopted in this thesis. The model dictionary is made of hyper-reduced-order models [33], and the input data $\{x_i\}_{1\leq i\leq m}$ are images of a mechanical experiment. The dimensionality of simulation data is reduced by Principal Component Analysis (PCA) before using k-means clustering. A convolutional neural network [166] is trained to return class labels without computing the intermediate variable u in order to avoid time-consuming operations. This classifier is an approximation of the *true classifier* $\mathcal{K}$ returning the correct label for any input x.

5.6 Feature selection based on mutual information

5.6.1 Introduction to feature selection

When classification data are high-dimensional, dimensionality reduction techniques can be applied to reduce the amount of information to be analyzed by the classifier. For classification problems where the dimension of the input data is higher than the number of training examples, dimensionality reduction is crucial to avoid overfitting. In addition, when considering physical fields discretized on a mesh, the dimension of the input space can reach 10^6 to 10^8 for industrial problems. In such cases, the input data are too hard to manipulate, which dramatically slows down the training process for the classifier and thus restrains the exploration of the hyperparameters space, as it requires multiple runs of the training process with different values for the hyperparameters. Applying data augmentation techniques to increase the number of examples in the training set is also impossible, as it would cause memory problems. Therefore, dimensionality reduction is recommended not only for reducing the risk of overfitting, but also for facilitating the training phase and enabling data augmentation.

Feature selection [167] aims at decreasing the number of features by selecting a subset of the original features. It differs from *feature extraction*, where new features are created from the original ones (e.g., Principal Component Analysis (PCA), and more generally encoders taken from undercomplete autoencoders [3]). Feature selection can be seen as applying a mask to a high-dimensional random vector to get a low-dimensional random vector containing the most relevant information. It is preferred over autoencoders when interpretability is important [168]. Furthermore, contrary to undercomplete autoencoders trained with the mean squared error loss, most feature selection algorithms do not intend to find reduced features enabling the reconstruction of the input: features are selected for the purpose of predicting class labels, which makes these algorithms more goal-oriented for supervised learning tasks.

Among the existing feature selection algorithms, univariate filter methods consist in computing a score for each feature and ranking the features according to their scores. The score measures how relevant a feature is for the prediction of the output variable. If N_f is the target number of features, then the N_f features with the highest scores are selected, and the others are discarded. The major drawback of univariate filter methods is that they do not account for relations between the selected features. The resulting set of selected features may then contain redundant features. To address this issue, the *minimum redundancy maximum relevance* (mRMR) algorithm [44, 45] tries to find a trade-off between relevance and redundancy.

Remark 5.6.1. *Feature selection is not required for high-dimensional data represented by second-order or third-order tensors, obtained for example by projecting a 2D or 3D field on a voxel grid. Indeed, dedicated learning algorithms have been developed to circumvent the problem of their dimension, namely convolutional neural networks (CNNs).*

5.6.2 mRMR feature selection

We recall that a projection π is a linear map satisfying $\pi \circ \pi = \pi$, with $\circ$ denoting function composition. It is entirely defined by its kernel and its image, which are complementary: given two complementary vector subspaces V_1 and V_2, there is a unique projection π whose kernel is V_1 and whose image is V_2, namely, the projection onto V_2 along V_1. For more details about projections, see in [169], pages 385 to 388. Let us now give a formal definition of a *feature selector*:

Definition 5.6.2 (Feature selector)**.** *Let V be a finite-dimensional real vector space. Given a basis $\mathcal{B} = (e_i)_{1 \leq i \leq \dim(V)}$ of V and a set of integers $S \subset [\![1; \dim(V)]\!]$, the feature selector $\pi_{S,\mathcal{B}} : V \to V$ is the projection whose image is* $\operatorname{span}\left(\{e_i\}_{i \in S}\right)$ *and whose kernel is* $\operatorname{span}\left(\{e_i\}_{i \in [\![1;\dim(V)]\!] \setminus S}\right)$.

When the choice of the basis $\mathcal{B}$ is obvious, the notation $\pi_{S,\mathcal{B}}$ is simply replaced by π_S. In practice,

$$\forall (\lambda_i)_{1 \leq i \leq \dim(V)} \in \mathbb{R}^{\dim(V)}, \quad \pi_S \left(\sum_{i=1}^{\dim(V)} \lambda_i e_i \right) = \sum_{i \in S} \lambda_i e_i \tag{5.46}$$

Therefore, from a numerical point of view, one can interpret the feature selector as linear map $\pi_S : V \to \operatorname{span}\left(\{e_i\}_{i \in S}\right)$, which enables reducing the size of the vector representing $\pi_S(x)$ for $x \in V$. In this way, applying a feature selector π_S to a vector of $\mathbb{R}^{\mathcal{N}}$ consists in

masking its features whose indexes are not in S, which gives a reduced vector in $\mathbb{R}^{|S|}$ where $|S|$ denotes the number of elements in S. Feature selection algorithms build the set S by searching for the most relevant features for the prediction of the output variable Y. For this purpose, the mutual information (see Definition 2.2.13) can be used to quantify the degree of the relationship between variables. Given Equation (5.2) and Definition 2.2.13, the mutual information between a feature X^i of X and the categorical output Y reads:

$$I\left(X^i, Y\right) = \sum_{k=1}^{K} \mathbb{P}_Y(k) \int_{x^i \in \mathbb{R}} p_{X^i|Y}(x^i|k) \log\left(\frac{p_{X^i|Y}(x^i|k)}{p_{X^i}(x^i)}\right) dx^i \tag{5.47}$$

The mutual information can be used to quantify the redundancy of a set of features S with cardinality $|S|$ and its relevance for predicting Y:

Definition 5.6.3 (Relevance [45], eq. 4, p. 2)**.** *Let $X = (X^i)_{1 \leq i \leq \mathcal{N}}$ be a multivariate random variable, and let Y be a discrete random variable. The relevance of a reduced set $S \subset [\![1; \mathcal{N}]\!]$ of features of X for predicting Y is defined by*

$$D(S, Y) = \frac{1}{|S|} \sum_{i \in S} I(X^i, Y) \tag{5.48}$$

Definition 5.6.4 (Redundancy [45], eq. 5, p. 2)**.** *Let $X = (X^i)_{1 \leq i \leq \mathcal{N}}$ be a multivariate random variable. The redundancy of a reduced set $S \subset [\![1; \mathcal{N}]\!]$ of features of X is defined by*

$$R(S) = \frac{1}{|S|^2} \sum_{i,j \in S^2} I(X^i, X^j) \tag{5.49}$$

The *minimum redundancy maximum relevance* (mRMR) algorithm [44, 45] builds the set S by maximizing $D(S, Y) - R(S)$, which is a combinatorial optimization problem. For this type of optimization problem, a brute-force search is intractable, because the number of solution candidates is too large. Instead, mRMR searches for a sub-optimal solution by following a greedy approach. First, the feature having the highest mutual information with the label variable Y is selected. Then, the algorithm follows an incremental procedure: given the set S_{k-1} obtained at iteration $k-1$, form the set S_k such that

$$S_k = S_{k-1} \cup \left\{ \underset{i \in [\![1; \mathcal{N}]\!] \setminus S_{k-1}}{\arg\max} \left(I(X^i, Y) - \frac{1}{k-1} \sum_{j \in S_{k-1}} I(X^i, X^j) \right) \right\} \tag{5.50}$$

This incremental procedure stops when k reaches the target number of features N_f. A review of feature selection algorithms based on mutual information can be found in [170].

* *
*

Part III

Nonlinear model order reduction

Résumé

La plupart des phénomènes physiques peuvent être modélisés à l'aide d'équations mathématiques, en particulier par des équations aux dérivées partielles. La résolution numérique de ces équations permet de simuler le phénomène physique, pratique devenue courante en ingénierie pour compléter les informations tirées de campagnes expérimentales. Nous disposons aujourd'hui de modèles sophistiqués permettant de simuler le comportement de systèmes complexes faisant intervenir différents phénomènes physiques couplés. Néanmoins, la précision de ces modèles est altérée par le manque de connaissance de l'environnement exact dans lequel le système étudié évolue, ainsi que par les incertitudes sur l'état et les propriétés réels de ce système. Quantifier les incertitudes sur un résultat de simulation numérique est donc primordial pour optimiser un produit en phase de conception et s'assurer de sa fiabilité. La quantification d'incertitudes requiert généralement de lancer un grand nombre de simulations pour différentes configurations possibles du système et de son environnement, ce qui est impossible pour certaines applications industrielles pour lesquelles la durée d'une seule simulation est trop importante pour envisager de lancer des milliers de calculs.

La réduction de modèle permet de réduire la complexité et le temps des calculs en fournissant une solution approchée aux équations du modèle. Contrairement aux méthodes de régression qui se substituent aux modèles physiques, les méthodes de réduction de modèle par projection identifient un espace d'approximation de faible dimension permettant de résoudre de manière approchée les équations du modèle par projection de Galerkin. L'espace d'approximation est construit à partir de données d'entraînement générées par un modèle haute-fidélité, et est donc adapté au problème physique étudié, contrairement à l'espace d'approximation générique utilisé par le modèle haute-fidélité (méthode des éléments finis ou des volumes finis, par exemple). La décomposition orthogonale aux valeurs propres (POD pour *Proper Orthogonal Decomposition*) est couramment utilisée comme méthode de réduction de dimension linéaire pour construire cet espace d'approximation afin de réduire le nombre de degrés de liberté à déterminer. Pour des modèles basés sur des lois de comportement non-linéaires faisant intervenir plusieurs variables internes, il est parfois nécessaire d'avoir recours à une méthode d'hyper-réduction en complément de la POD afin d'accélérer les calculs de manière efficace.

Un problème décrit par une équation aux dérivées partielles est dit réductible si la suite des épaisseurs de Kolmogorov décroît rapidement lorsque la dimension de l'espace d'approximation augmente. L'épaisseur de Kolmogorov quantifie l'erreur que l'on ferait en calculant une solution approchée dans un espace d'approximation de dimension donnée. Certains problèmes sont très sensibles aux variations des paramètres incertains et peuvent générer des solutions vivant dans une variété différentielle trop complexe pour permettre une décroissance rapide des épaisseurs de Kolmogorov. Ces problèmes sont non-réductibles

et ont fait l'objet de nombreuses recherches dans la communauté de la réduction de modèle. Parmi les méthodes issues de ces travaux de recherche, l'utilisation d'un dictionnaire de modèles réduits locaux permet d'avoir recours à une collection d'espaces d'approximation associés à différentes régions de la variété contenant les solutions possibles. C'est la méthode retenue dans ce travail de thèse.

* *
*

Chapter 6

Projection-based model order reduction

Abstract: *Projection-based model order reduction methods consist in finding a low-dimensional approximation space where an approximate solution of a partial differential equation can be computed in a reasonable computation time. For complex nonlinear problems, hyper-reduction enables using a reduced number of integration points in a finite-element mesh. Model order reduction and hyper-reduction are essential for applications requiring fast simulations, but also for the reduction of energy consumption [171] in problems involving many numerical simulations such as in uncertainty quantification and design optimization. This chapter introduces the Proper Orthogonal Decomposition [1, 2] that is widely used for the construction of suitable approximation spaces, and presents the Empirical Cubature Method [34], a hyper-reduction technique that is used in the industrial application of this thesis.*

Remark 6.0.1. *This chapter includes some paragraphs taken from our papers [47, 49], with some modifications.*

Contents

6.1 Parametrized partial differential equations

Let us consider a physics problem described by the following parametrized differential equation:

$$\mathcal{D}(u; x) = 0 \tag{6.1}$$

where u is the primal variable belonging to a Hilbert space $\mathcal{H}$ whose inner product is denoted by $\langle ., . \rangle_{\mathcal{H}}$, x denotes the parameters of the problem, and $\mathcal{D}$ is an operator involving a differential operator and operators for initial conditions and/or boundary conditions. Equation (6.1) can be a system of ordinary differential equations (ODEs) or partial differential equations (PDEs) depending on the physics problem. Let us assume that this physics problem is well-posed in the sense of Hadamard, that is to say that there exists a unique solution $u(x)$ for any parameter x, and that this solution changes continuously with x. Let us introduce the set $\mathcal{X}$ of all the possible parameters x. The *solution manifold* $\mathcal{M}$ is defined by:

$$\mathcal{M} := u(\mathcal{X}) = \{u(x) \mid x \in \mathcal{X}\} \tag{6.2}$$

When a probability distribution is defined on $\mathcal{X}$ to describe the stochastic behavior of the parameter x with a random variable, Equation (6.1) becomes a *stochastic* (partial) differential equation. Stochastic PDEs are extensively studied in uncertainty quantification.

6.2 Model order reduction techniques

Model order reduction [30, 31] is a discipline in numerical analysis consisting in replacing a computationally expensive high-fidelity model by a fast reduced-order model (ROM) to calculate approximate solutions of some complex physics equations. A ROM can be either a data-driven metamodel (or surrogate model) calibrated with a regression algorithm, or a physics-based model obtained by numerical methods such as the Proper Generalized Decomposition (PGD [172, 173]), the Reduced Basis method (RBM [174, 175]), and the POD Galerkin method [2, 32], among others. It is generally used for parametrized equations whose solution must be known for different points in the parameter space. As in machine learning, a model order reduction procedure starts by a *training phase* (or *offline stage*) where the ROM is built from some training data. The ROM is then used on test data in an *exploitation phase* (or *online stage*). In the training phase, high-fidelity solutions, called *snapshots*, are computed with the high-fidelity model for different points of the parameter space to get a sampled representation of the solution manifold. The model order reduction algorithm analyzes these snapshots to learn how the solution is affected by parameter variations. Contrary to usual machine learning problems, the amount of training data is limited because the high-fidelity model giving snapshots is time-consuming and costly. The selection of relevant points in the parameter space can be optimized to ensure that the snapshots are representative of the behavior of the solution, like in the greedy approach of the Reduced Basis method where an a posteriori error estimator is used to select snapshots. Given the cost of computing snapshots in the training phase, a ROM is profitable only if it is extensively used in the exploitation phase. In this thesis, we will focus on *projection-based model order reduction* (e.g. POD Galerkin, Reduced Basis method) where the approximate solution is obtained by solving the physics equations with the Galerkin method on a well-chosen reduced-order basis (ROB).

Following the terminology introduced in [35], the training phase of a projection-based model order reduction method has 3 main steps:

- *Data generation:* snapshots are computed with the high-fidelity model and provide information about how the physical system reacts to changes of the parameter x;
- *Data compression:* a ROB is constructed by looking for a hidden low-rank structure in the snapshots, using dimensionality reduction techniques such as the POD;
- *Operator compression:* operator compression includes all the operations that guarantee the efficiency of the reduced-order model in the exploitation phase. It includes pre-computations of some integral terms facilitating the assembly of the reduced problem in the online stage[1], but also the use of a hyper-reduction method to guarantee the reduction of the computational cost for complex nonlinear problems.

The next sections introduce methods for data compression and operator compression.

6.3 Data compression

6.3.1 The Proper Orthogonal Decomposition

It is now assumed that Equation (6.1) defines a parametrized partial differential equation whose solution $u(x)$ for a given point $x \in \mathcal{X}$ in the parameter space is a function of space and time defined on $\Omega \times [0; t_f]$, with $\Omega \subset \mathbb{R}^\alpha$, $\alpha = 1$, 2 or 3 and $t_f \in \mathbb{R}_+^*$. Most of the time, the Hilbert space $\mathcal{H}$ is a subspace of the Lebesgue space $L^2(\Omega \times [0; t_f])$ of square-integrable functions. However, parameters x and time t can be considered together in a variable χ called *generalized parameters* living in the set $\tilde{\mathcal{X}} = \mathcal{X} \times [0; t_f]$. Therefore, the solution $u(\chi)$ belongs to the space $L^2(\Omega)$. The Stiefel manifold $V(N, L^2(\Omega))$ represents the set of all orthonormal N-frames in $L^2(\Omega)$. For two square-integrable functions f and g, the notation $\langle f, g \rangle_{L^2(\Omega)}$ stands for the $L^2(\Omega)$ inner product $\int_\Omega f(\boldsymbol{\xi}) g(\boldsymbol{\xi}) d\boldsymbol{\xi}$. The following definition gives a theoretical continuous definition of the proper orthogonal decomposition (POD [1, 2]):

Definition 6.3.1 (POD basis)**.** *Let $u : \tilde{\mathcal{X}} \to L^2(\Omega)$. A POD basis $\{\psi_k^*\}_{1 \leq k \leq N} \in V(N, L^2(\Omega))$ of order $N \in \mathbb{N}^*$ of u is a solution of the following optimization problem:*

$$\{\psi_k^*\}_{1 \leq k \leq N} \in \underset{\{\psi_k\}_{1 \leq k \leq N} \in V(N, L^2(\Omega))}{\arg\min} \int_{\chi \in \tilde{\mathcal{X}}} ||u(\chi) - \sum_{k=1}^{N} \langle u(\chi), \psi_k \rangle_{L^2(\Omega)} \psi_k||_{L^2(\Omega)}^2 \, d\chi \quad (6.3)$$

The sum in Equation (6.3) is the proper orthogonal decomposition of order N of u. When $N \to +\infty$, the approximation error given by the minimum of the cost function in Equation (6.3) tends towards zero (Theorem 4 in [176]). The proper orthogonal decomposition is related to the Karhunen-Loève decomposition introduced in Equation 2.33.

[1] This technique is particularly relevant for linear and nonlinear problems with affine parameter dependence.

Let $\mathcal{H}$ be a Hilbert space with an orthonormal basis $\{e_i\}_{i\in\mathbb{N}^*}$, and $A : \mathcal{H} \to \mathcal{H}$ a linear operator. We define the Hilbert-Schmidt function $\Lambda_{HS(\mathcal{H})}$ as:

$$\Lambda_{HS(\mathcal{H})}(A) := \sqrt{\sum_{i=1}^{\infty} ||A(e_i)||_{\mathcal{H}}^2} \tag{6.4}$$

which can potentially take infinite values. A linear operator A on a Hilbert space $\mathcal{H}$ is a *Hilbert-Schmidt operator* if $\Lambda_{HS(\mathcal{H})}(A)$ is finite. As shown in [177] (Chapter VIII, Theorem 2.3), the set $HS(\mathcal{H})$ of all Hilbert-Schmidt operators on $\mathcal{H}$ is a Hilbert space with respect to the following inner product:

$$\langle A, B\rangle_{HS(\mathcal{H})} := \sum_{i=1}^{\infty} \langle A(e_i), B(e_i)\rangle_{\mathcal{H}} \tag{6.5}$$

The Hilbert-Schmidt function $\Lambda_{HS(\mathcal{H})}$ is actually the norm induced by this inner product, and corresponds to the Frobenius norm for matrices when the vector space $\mathcal{H}$ is finite-dimensional. We now use the more conventional notation $||A||_{HS(\mathcal{H})} := \Lambda_{HS(\mathcal{H})}(A)$ for the Hilbert-Schmidt norm. The Hilbert-Schmidt inner product and norm are independent of the choice of the basis $\{e_i\}_{i\in\mathbb{N}^*}$, see Proposition 9.18 in Chapter 9 of [178], which will be useful for proofs of some properties of the dissimilarity measure introduced in this thesis. The POD is highly related to the theory of Hilbert-Schmidt operators. In [179, 180], it is shown that the POD optimization problem is equivalent to finding the optimal approximation of a Hilbert-Schmidt operator related to u by a finite rank operator in the Hilbert-Schmidt norm. The POD basis functions can also be obtained from the eigenfunctions of the Hilbert-Schmidt integral operator [2] $\mathcal{R}_u$:

$$L^2(\tilde{\mathcal{X}}) \ni \varphi \mapsto \mathcal{R}_u(\varphi) \in L^2(\tilde{\mathcal{X}}), \tag{6.6}$$

where $\mathcal{R}_u(\varphi)$ is defined by:

$$\tilde{\mathcal{X}} \ni \chi \mapsto \mathcal{R}_u(\varphi)(\chi) := \int_{\chi'\in\tilde{\mathcal{X}}} \left\langle u(\chi), u(\chi')\right\rangle_{L^2(\Omega)} \varphi(\chi')d\chi' \in \mathbb{R} \tag{6.7}$$

In this work, we keep the explicit distinction between the time t and the parameters $x \in \mathcal{X}$ rather than working on the generalized parameters $\chi \in \tilde{\mathcal{X}}$, because we do not consider the time as a clustering variable. Nonetheless, spatio-temporal functions $f \in L^2(\Omega \times [0; t_f])$ are considered as trajectories $(f(., t))_{t\in[0;t_f]}$ in the Hilbert space $L^2(\Omega)$. In other words, such functions are seen as functions defined on Ω and parametrized by the time. For this reason, the manifold $\mathcal{M}$ is rather defined by:

$$\mathcal{M} := \{u(x)(., t) \mid x \in \mathcal{X},\ t \in [0; t_f]\} \tag{6.8}$$

and the approximation spaces are subspaces of $L^2(\Omega)$, leading to an approximate solution expressed as a time-dependent linear combination of basis functions defined on Ω.

In practice, we are given a finite set of m points x_i of the parameter space $\mathcal{X}$, for which high-fidelity solutions $u(x_i)$ are computed in a high-dimensional approximation space whose dimension is denoted by $\mathcal{N}$. These solutions, called snapshots, provide information about the behavior of the physical system and give a sampled version of the solution manifold. The POD is applied as a linear dimensionality reduction technique, processing this information to build a ROB that can be used to accelerate future numerical simulations for new parameters.

Definition 6.3.2 (POD basis construction)**.** *Given an integer $N \leq \mathcal{N}$, a POD basis $\{\psi_k^*\}_{1\leq k\leq N} \in V(N, L^2(\Omega))$ is computed from the snapshots $\{u(x_i)\}_{1\leq i\leq m}$ as a solution of the following optimization problem:*

$$\{\psi_k^*\}_{1\leq k\leq N} \in \underset{\{\psi_k\}_{1\leq k\leq N}\in V(N,L^2(\Omega))}{\arg\min} \sum_{i=1}^{m} ||u(x_i) - \sum_{k=1}^{N} \langle u(x_i), \psi_k\rangle_{L^2(\Omega)}\psi_k||^2_{L^2(\Omega\times[0;t_f])} \quad (6.9)$$

The uniqueness of the POD basis is obtained by specifying a construction algorithm, such as the Snapshot POD [181, 182] or the singular value decomposition (SVD) for instance. By construction, the subspace spanned by the ROB minimizes the projection errors of the snapshots $u(x_i)$. The optimality of the POD basis is discussed and illustrated in [183]. In practice, when using a numerical procedure to solve Equation (6.1), for example the finite-element method with a time-stepping scheme, the coordinates of the snapshots in the finite-element basis are stored in columns in a matrix $\mathbf{Q} \in \mathbb{R}^{\mathcal{N}\times mn_t}$ called snapshots matrix, with n_t being the number of time steps. The coordinates of the POD modes ψ_k are given in the N first columns of the matrix $\mathbf{M}^{-1/2}\mathbf{V}$, where $\mathbf{M} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ is the finite-element mass matrix and $\mathbf{V} \in \mathbb{R}^{\mathcal{N}\times\text{rank}(\mathbf{Q})}$ is the matrix of left singular vectors in the SVD of the snapshots matrix $\mathbf{Q}$, when indexing the singular values in decreasing order. The decay rate of the singular values of the snapshots matrix $\mathbf{Q}$ is related to the behavior of the sequence of Kolmogorov widths of the PDE given in Equation (6.1). It enables evaluating the reducibility of the physics problem. When computing a POD basis for a variable defined at integration points rather than the finite-element mesh nodes, for the purpose of applying Gappy POD after hyper-reduced simulations, the POD modes are simply given by the N first left singular vectors in the SVD of the corresponding snapshots matrix.

Remark 6.3.3 (Singular value decomposition (SVD))**.** *The SVD of the snapshots matrix $\mathbf{Q} \in \mathbb{R}^{\mathcal{N}\times mn_t}$ reads:*

$$\mathbf{Q} = \tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{W}}^T = \mathbf{V}\mathbf{\Sigma}\mathbf{W}^T \quad (6.10)$$

where $\tilde{\mathbf{\Sigma}} = diag(\sigma_1, ..., \sigma_r, 0, ..., 0) \in \mathbb{R}^{\mathcal{N}\times mn_t}$ and $\mathbf{\Sigma} = diag(\sigma_1, ..., \sigma_r) \in \mathbb{R}^{r\times r}$, with $r = \text{rank}(\mathbf{Q})$. The diagonal terms $\sigma_1, ..., \sigma_r$ are positive and correspond to the nonzero singular values of $\mathbf{Q}$. The columns of $\tilde{\mathbf{V}} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ (resp. $\tilde{\mathbf{W}} \in \mathbb{R}^{mn_t\times mn_t}$) contain the left singular vectors (resp. the right singular vectors). The matrices $\tilde{\mathbf{V}}$ and $\tilde{\mathbf{W}}$ are orthogonal: $\tilde{\mathbf{V}}^T\tilde{\mathbf{V}} = \tilde{\mathbf{V}}\tilde{\mathbf{V}}^T = \mathbf{I}_{\mathcal{N}}$ and $\tilde{\mathbf{W}}^T\tilde{\mathbf{W}} = \tilde{\mathbf{W}}\tilde{\mathbf{W}}^T = \mathbf{I}_{mn_t}$. The matrix $\mathbf{V} \in \mathbb{R}^{\mathcal{N}\times r}$ (resp. $\mathbf{W} \in \mathbb{R}^{mn_t\times r}$) contains the r first columns of the matrix $\tilde{\mathbf{V}}$ (resp. $\tilde{\mathbf{W}}$) associated to the r nonzero singular values. The left singular vectors of $\mathbf{Q}$ correspond to the eigenvectors of the matrix $\mathbf{Q}\mathbf{Q}^T$ since $\mathbf{Q}\mathbf{Q}^T\mathbf{V} = \mathbf{V}\mathbf{\Sigma}^2$. They can also be obtained with the formula $\mathbf{V} = \mathbf{Q}\mathbf{W}\mathbf{\Sigma}^{-1}$, where the right singular vectors are given by the eigenvectors of $\mathbf{Q}^T\mathbf{Q}$ since $\mathbf{Q}^T\mathbf{Q}\mathbf{W} = \mathbf{W}\mathbf{\Sigma}^2$. Note that the columns of $\mathbf{M}^{-1/2}\mathbf{V}$ are orthogonal for the L^2 inner product:

$$(\mathbf{M}^{-1/2}\mathbf{V})^T\mathbf{M}(\mathbf{M}^{-1/2}\mathbf{V}) = \mathbf{V}^T\mathbf{V} = \mathbf{I}_r. \quad (6.11)$$

Remark 6.3.4 (Snapshot POD)**.** *The Snapshot POD consists in: (i) computing the snapshot correlation matrix $\mathbf{C}$ with the formula $C_{ij} = \langle u(\chi_i), u(\chi_j)\rangle_{L^2(\Omega)}$, $1 \leq i, j \leq mn_t$, (ii) retaining the eigenvalue/eigenvector pairs of $\mathbf{C}$ associated to the N highest eigenvalues: (λ_i, ζ_i), $1 \leq i \leq N$, and (iii) recombining them with the snapshots to create the ROB: $\psi_i = \lambda_i^{-1/2}\sum_{k=1}^{mn_t} u(\chi_k)\zeta_{ik}$, $1 \leq i \leq N$. Note that we have used the generalized parameters χ for the equations of the Snapshot POD, in order to simplify the numbering of the mn_t snapshots provided by m simulations with n_t time steps each.*

6.3.2 The POD Galerkin method

This section shows how to solve the PDE by Galerkin projection onto the approximation space spanned by the POD basis. Let us consider, for example, a physics problem that can be solved by the finite-element method, such as in solid mechanics or heat transfer. Let us introduce the finite-element shape functions $\{\phi_i\}_{1\leq i\leq\mathcal{N}}$. The finite-element approximation $u_h(x)$[2] of the primal variable $u(x)$ is:

$$\forall x \in \mathcal{X}, \quad \forall t \in [0; t_f], \quad \forall \boldsymbol{\xi} \in \Omega, \qquad u_h(x)(\boldsymbol{\xi}, t) = \sum_{i=1}^{\mathcal{N}} u_{h,i}(x,t)\phi_i(\boldsymbol{\xi}) \tag{6.12}$$

The finite-element solution $u_h(x)(.,t_n)$ computed at the n-th time step can be represented by a vector $\mathbf{U}_n(x) \in \mathbb{R}^{\mathcal{N}}$ containing its coordinates $u_{h,i}(x,t_n)$ in the finite-element basis $\{\phi_i\}_{1\leq i\leq\mathcal{N}}$. For simplicity, the dependence with respect to x is omitted in this section. This numerical solution can be obtained with the Newton-Raphson algorithm, an iterative procedure based on the linearization of the weak formulation of the PDE. The resulting linear system to be solved for the k-th iteration at the n-th time increment reads:

$$\mathbf{J}_n^{(k)} \delta\mathbf{U}_n^{(k)} = -\mathbf{R}_n^{(k)} \tag{6.13}$$

where $\mathbf{J}_n^{(k)} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ is the Jacobian matrix (also called global tangent stiffness matrix in mechanics), $\mathbf{R}_n^{(k)} \in \mathbb{R}^{\mathcal{N}}$ is the vector of residuals, and $\delta\mathbf{U}_n^{(k)} \in \mathbb{R}^{\mathcal{N}}$ is the correction applied to the vector of increments of the primal variable defined by:

$$\Delta\mathbf{U}_n^{(k)} = \Delta\mathbf{U}_n^{(k-1)} + \delta\mathbf{U}_n^{(k)} \tag{6.14}$$

with $\Delta\mathbf{U}_n^{(0)} = \mathbf{0}$. When the convergence criterion $||\mathbf{R}_n^{(k)}|| \leq \epsilon_{NR}||\mathbf{F}_n^{\text{ext}}||$ is satisfied for a given $k = k^*$ with ϵ_{NR} being the tolerance of the Newton-Raphson algorithm and $\mathbf{F}_n^{\text{ext}}$ being the vector of external forces, the solution at the n-th time increment is defined as:

$$\mathbf{U}_n = \mathbf{U}_{n-1} + \Delta\mathbf{U}_n^{(k^*)} \tag{6.15}$$

Equation (6.13) is the high-dimensional linear system deriving from the high-fidelity model.

Projection-based model order reduction consists in searching an approximation of the high-fidelity solution in a low-dimensional subspace $\mathcal{V}_{ROM} \subset \text{span}\{\phi_i\}_{1\leq i\leq\mathcal{N}}$ adapted to the current physics problem. This subspace is spanned by an appropriate reduced-order basis $\{\psi_i\}_{1\leq i\leq N}$, with N being very small compared to $\mathcal{N}$. The reduced-order approximation of the primal variable reads:

$$\forall t \in [0; t_f], \quad \forall \boldsymbol{\xi} \in \Omega, \qquad u_{ROM}(\boldsymbol{\xi}, t) = \sum_{i=1}^{N} \gamma_i(t)\psi_i(\boldsymbol{\xi}) \tag{6.16}$$

where $\{\gamma_i\}_{1\leq i\leq N}$ are the reduced coordinates which can be stored in a vector $\boldsymbol{\gamma} \in \mathbb{R}^N$. The coordinates of the modes $\{\psi_i\}_{1\leq i\leq N}$ in the finite-element basis $\{\phi_i\}_{1\leq i\leq\mathcal{N}}$ are stored in columns in a matrix $\mathbf{V} \in \mathbb{R}^{\mathcal{N}\times N}$ called reduction matrix. Hence:

$$\forall i \in [\![1; N]\!], \quad \forall \boldsymbol{\xi} \in \Omega, \qquad \psi_i(\boldsymbol{\xi}) = \sum_{j=1}^{\mathcal{N}} V_{ji}\phi_j(\boldsymbol{\xi}) \tag{6.17}$$

[2]The index h generally refers to a parameter controlling the mesh size.

After the Galerkin projection of the governing equations onto $\mathcal{V}_{ROM}$, the reduced linear system to solve at each iteration of the Newton-Raphson algorithm in the reduced-order model (ROM) is then:

$$\mathbf{V}^T\mathbf{J}_n^{(k)}\mathbf{V}\delta\boldsymbol{\gamma}_n^{(k)} = -\mathbf{V}^T\mathbf{R}_n^{(k)} \tag{6.18}$$

6.4 Operator compression

6.4.1 Hyper-reduction

For a nonlinear model, the Jacobian matrix must be updated at every iteration. Hence, the products $\mathbf{V}^T\mathbf{J}_n^{(k)}\mathbf{V}$ and $\mathbf{V}^T\mathbf{R}_n^{(k)}$ in Equation (6.18) must be computed at every iteration. As explained in [184], if ω is the average number of nonzero entries in a row of the Jacobian matrix, then the product $\mathbf{V}^T\mathbf{J}_n^{(k)}\mathbf{V}$ involves $2\mathcal{N}N(\omega + N)$ floating-point operations. The product $\mathbf{V}^T\mathbf{R}_n^{(k)}$ requires $2N\mathcal{N}$ floating-point operations. In spite of the reduction in terms of number of degrees of freedom, the computational cost of these products still scales linearly with the dimension $\mathcal{N}$ of the high-fidelity model. In addition, all the entries of the Jacobian and the residual vector must be evaluated to compute these products, and the cost of the local integration of the constitutive laws is non-negligible for nonlinear models involving many internal variables. Because of this additional complexity, POD bases do not efficiently diminish the computational cost of the nonlinear problem.

Hence, a second reduction stage is necessary for these nonlinear problems, this time in terms of integration points. This is called *hyper-reduction*, or *operator compression* according to the terminology introduced in [35]. *Hyper-reduction* was initially the name of a method developed by David Ryckelynck in [33], but this term has been extended to refer to all the methods proposing a second reduction stage. Hyper-reduction methods include the empirical interpolation method (EIM, [185]), the missing point estimation (MPE, [186]), the a priori hyperreduction (APHR, [33]), the best point interpolation method (BPIM, [187]), the discrete empirical interpolation method (DEIM, [165]), the Gauss-Newton with approximated tensors (GNAT, [188]), the enery-conserving sampling and weighting (ECSW, [189]), the empirical cubature method (ECM, [34]), and the linear program empirical quadrature procedure (LPEQP, [190]). Hyper-reduction techniques implicitly assume that the physics model is based on *local* constitutive laws. A constitutive model is *local* if its equations evaluated at a given point $\boldsymbol{\xi} \in \Omega$ only involve variables evaluated at $\boldsymbol{\xi}$. The constitutive equations are ordinary differential equations in time, and can be solved pointwise. Consequently, some integration points can be removed without preventing the integration of the constitutive laws on the remaining points. This is no longer possible when the constitutive equations include effects of the neighborhood, which is the case for contact problems [191] and for some *nonlocal* models that are used in damage mechanics to avoid mesh dependency [192, 193].

6.4.2 The Empirical Cubature Method

Instead of computing the Jacobian matrix and the residual vector before solving Equation (6.18), the products $\mathbf{V}^T\mathbf{J}_n^{(k)}\mathbf{V}$ and $\mathbf{V}^T\mathbf{R}_n^{(k)}$ can be evaluated directly by computing integrals involving the POD modes. But contrary to the shape functions, the POD modes do not have local supports, and therefore these integrals must be computed over the whole

domain Ω with a generic cubature formula involving all the integration points of the finite-element mesh:

$$\int_{\Omega} f(\boldsymbol{\xi}) d\boldsymbol{\xi} \approx \sum_{i=1}^{N_G} \omega_i f(\boldsymbol{\xi}_i^G) \tag{6.19}$$

where $f : \Omega \to \mathbb{R}$ is an integrand, $\{\omega_i\}_{i=1}^{N_G}$ is the set of integration weights, and $\left\{\boldsymbol{\xi}_i^G\right\}_{i=1}^{N_G}$ is the set of Gauss points (or integration points). Evaluating the integrand at every integration point can be expensive, the number of Gauss points N_G being proportional to the dimension $\mathcal{N}$ of the high-fidelity model. Values of the integrand must be updated at every iteration and every Gauss point, which requires calling the constitutive equations solver many times.

The idea of the ECM proposed in [34] is to determine a small subset of integration points $\left\{\hat{\boldsymbol{\xi}}_i^G\right\}_{i=1}^{n_G}$ and their associated weights $\{\hat{\omega}_i\}_{i=1}^{n_G}$ such that the integration error is minimized, with n_G being orders of magnitude smaller than N_G. The cubature formula becomes:

$$\int_{\Omega} f(\boldsymbol{\xi}) d\boldsymbol{\xi} \approx \sum_{i=1}^{n_G} \hat{\omega}_i f(\hat{\boldsymbol{\xi}}_i^G) \tag{6.20}$$

The integration weights $\{\hat{\omega}_i\}_{i=1}^{n_G}$ have to be positive in order to preserve the spectral properties of the Jacobian matrix, namely symmetry and positive definiteness. The fact that the spectral properties of the Jacobian are preserved guarantees the stability of the hyper-reduced model, which is not the case in some nodal vector approximation approaches such as the EIM. The optimal reduced-integration points and their corresponding weights are the solution of a combinatorial optimization problem involving $\binom{N_G}{n_G}$ nonnegative least square problems. In practice, one looks for a sub-optimal solution with a heuristic approach, such as the *Nonnegative orthogonal matching pursuit* [194] like in [35]. Following the terminology of the APHR [33], the set formed by the elements containing the reduced-integration points is called the *reduced-integration domain (RID)* and is denoted by Ω_R.

6.4.3 Dual variable reconstruction

In hyper-reduced simulations, the constitutive equations are solved only at the reduced-integration points. Consequently, the dual variables (or internal variables) are known on the RID only, whereas the primal variable is naturally defined on the whole mesh because it is a linear combination of the POD modes. The Gappy POD [46] can reconstruct the field of a dual variable given its values on the RID, using a POD basis inferred from the snapshots for this dual variable. Let $z \in L^2(\Omega)$ be a dual variable, known on the RID Ω_R only, and let $\{\psi_i^z\}_{1 \leq i \leq N^z}$ be a POD basis for this variable. The Gappy POD method consists in solving the following optimization problem:

$$\min_{\boldsymbol{\lambda} \in \mathbb{R}^{N^z}} ||z - \sum_{i=1}^{N^z} \lambda_i \psi_i^z||_{L^2(\Omega_R)}^2 \tag{6.21}$$

in order to find the optimal coefficients $\{\lambda_i\}_{1 \leq i \leq N^z}$ in the POD basis. When one wants to retrieve the whole history of z over time, this optimization problem must be solved for each time step.

* *

*

Chapter 7

Non-reducible problems

Abstract: *In some cases, the solution manifold of a parametrized partial differential equation cannot be embedded into a low-dimensional approximation space. Such problems are non-reducible, which means that classical model order reduction fails to compute accurate solutions in a reasonable computation time. The non-reducibility of a problem is related to the behavior of its Kolmogorov widths, which slowly decay as the dimension of the approximation space increases. This chapter introduces the definitions of Kolmogorov widths and gives an overview of strategies that have been developed to accelerate the computation of non-reducible problems. The interpretation of the normalized Kolmogorov width in terms of angles is particularly important for the methodology developed in this thesis.*

Remark 7.0.1. *This chapter uses paragraphs taken from our paper [49], with some modifications.*

Contents

7.1 Kolmogorov widths

Approximate solutions of Equation (6.1) can be obtained by solving the PDEs on a finite-dimensional subspace $\mathcal{H}_N \in \mathrm{Gr}(N, \mathcal{H})$ spanned by a ROB, where the Grassmannian $\mathrm{Gr}(N, \mathcal{H})$ is the set of all N-dimensional subspaces of $\mathcal{H}$. The best approximation of the solution in $\mathcal{H}_N$ for a given parameter x is the orthogonal projection $\pi_{\mathcal{H}_N}(u(x))$ of the theoretical solution $u(x)$ onto the approximation space:

$$\pi_{\mathcal{H}_N}(u(x)) = \underset{v \in \mathcal{H}_N}{\arg\min} \, ||u(x) - v||_{\mathcal{H}} \tag{7.1}$$

with $||.||_{\mathcal{H}}$ denoting the norm induced by the inner product of the Hilbert space $\mathcal{H}$.

The Kolmogorov N-width is defined by:

$$d_N(\mathcal{M}) := \inf_{\mathcal{H}_N \in \mathrm{Gr}(N,\mathcal{H})} \sup_{u \in \mathcal{M}} \inf_{v \in \mathcal{H}_N} ||u - v||_{\mathcal{H}} = \inf_{\mathcal{H}_N \in \mathrm{Gr}(N,\mathcal{H})} \sup_{u \in \mathcal{M}} ||u - \pi_{\mathcal{H}_N}(u)||_{\mathcal{H}} \tag{7.2}$$

and quantifies how well the solution manifold $\mathcal{M}$ can be approximated by searching approximate solutions in a N-dimensional subspace of $\mathcal{H}$. The Kolmogorov N-width corresponds to the worst projection error on the best N-dimensional approximation space. A variant of this definition is introduced in [195]:

$$\begin{aligned} d'_N(u, p_X) &:= \left(\inf_{\mathcal{H}_N \in \mathrm{Gr}(N,\mathcal{H})} \int_{x \in \mathcal{X}} p_X(x) \, ||u(x) - \pi_{\mathcal{H}_N}(u(x))||^2_{\mathcal{H}} \, dx \right)^{1/2} \\ &= \left(\inf_{\mathcal{H}_N \in \mathrm{Gr}(N,\mathcal{H})} \mathbb{E}_{X \sim p_X} \left[||u(X) - \pi_{\mathcal{H}_N}(u(X))||^2_{\mathcal{H}} \right] \right)^{1/2} \end{aligned} \tag{7.3}$$

where X is a random variable representing the parameters whose probability density function is $p_X : \mathcal{X} \to \mathbb{R}_+$ and whose observations are denoted by x. Equation (7.3) gives the smallest expected squared projection error that can be obtained when using a N-dimensional approximation space. For a fixed solution manifold $\mathcal{M}$, the sequences $(d_N(\mathcal{M}))_{N \in \mathbb{N}^*}$ and $(d'_N(u, p_X))_{N \in \mathbb{N}^*}$ are decreasing, which means that approximation errors get lower when increasing the dimension of the approximation space. Instead of considering the absolute projection error when defining the Kolmogorov N-width, one can use the relative projection error, which leads to the following definition:

Definition 7.1.1 (Normalized Kolmogorov N-width)**.** *Let $N \in \mathbb{N}^*$. If $\mathcal{M}$ contains at least one nonzero element, the normalized Kolmogorov N-width of the manifold $\mathcal{M}$ in the ambient Hilbert space $\mathcal{H}$ is defined by:*

$$\tilde{d}_N(\mathcal{M}) := \inf_{\mathcal{H}_N \in \mathrm{Gr}(N,\mathcal{H})} \sup_{u \in \mathcal{M} \setminus \{0\}} \inf_{v \in \mathcal{H}_N} \frac{||u - v||_{\mathcal{H}}}{||u||_{\mathcal{H}}} \tag{7.4}$$

Property 7.1.2 (Inequalities on Kolmogorov widths)**.** *If $\mathcal{M}$ is bounded and contains at least one nonzero element, then:*

$$\forall N \in \mathbb{N}^*, \quad d_N(\mathcal{M}) \leq \sup_{v \in \mathcal{M}} ||v||_{\mathcal{H}} \, \tilde{d}_N(\mathcal{M}) \tag{7.5}$$

Proof. The boundedness of $\mathcal{M}$ implies the existence of $\sup_{v \in \mathcal{M}} ||v||_{\mathcal{H}}$, thus for all $u \in \mathcal{M} \setminus \{0\}$:

$$||u - \pi_{\mathcal{H}_N}(u)||_{\mathcal{H}} \leq \sup_{v \in \mathcal{M}} ||v||_{\mathcal{H}} \frac{||u - \pi_{\mathcal{H}_N}(u)||_{\mathcal{H}}}{||u||_{\mathcal{H}}} \tag{7.6}$$

which implies Equation (7.5). □

Let $\measuredangle_{\mathcal{H}}(u,v) \in [0;\pi/2]$ denote the angle between two nonzero elements u and v of $\mathcal{H}$:

$$\measuredangle_{\mathcal{H}}(u,v) := \arccos\left(\frac{|\langle u,v\rangle_{\mathcal{H}}|}{||u||_{\mathcal{H}}||v||_{\mathcal{H}}}\right) \tag{7.7}$$

and let $\measuredangle_{\mathcal{H}}(u,\mathcal{V}) \in [0;\pi/2]$ be the angle between u and a subspace $\mathcal{V} \subset \mathcal{H}$:

$$\measuredangle_{\mathcal{H}}(u,\mathcal{V}) := \inf_{v\in\mathcal{V}} \measuredangle_{\mathcal{H}}(u,v) \tag{7.8}$$

The normalized Kolmogorov N-width is related to the largest angle between elements of the solution manifold and the approximation space:

Property 7.1.3. *Let $N \in \mathbb{N}^*$, and suppose that $\mathcal{M}$ contains at least one nonzero element. Then:*

$$\tilde{d}_N(\mathcal{M}) = \inf_{\mathcal{H}_N\in \mathrm{Gr}(N,\mathcal{H})} \sup_{u\in\mathcal{M}\setminus\{0\}} \sin\measuredangle_{\mathcal{H}}(u,\mathcal{H}_N) \tag{7.9}$$

Proof. Given that $\measuredangle_{\mathcal{H}}(u,\mathcal{H}_N) \in [0;\pi/2]$, the sine of the angle $\measuredangle_{\mathcal{H}}(u,\mathcal{H}_N)$ satisfies:

$$\begin{aligned}\sin\measuredangle_{\mathcal{H}}(u,\mathcal{H}_N) &= |\sin\measuredangle_{\mathcal{H}}(u,\mathcal{H}_N)| \\ &= \sqrt{1-\cos^2\measuredangle_{\mathcal{H}}(u,\mathcal{H}_N)} \\ &= \sqrt{1-\cos^2\inf_{v\in\mathcal{H}_N}\arccos\left(\frac{|\langle u,v\rangle_{\mathcal{H}}|}{||u||_{\mathcal{H}}||v||_{\mathcal{H}}}\right)}\end{aligned} \tag{7.10}$$

and, since the function $\alpha \to \sqrt{1-\cos^2\alpha}$ is increasing on the interval $[0;\pi/2]$:

$$\sin\measuredangle_{\mathcal{H}}(u,\mathcal{H}_N) = \inf_{v\in\mathcal{H}_N}\sqrt{1-\cos^2\arccos\left(\frac{|\langle u,v\rangle_{\mathcal{H}}|}{||u||_{\mathcal{H}}||v||_{\mathcal{H}}}\right)} = \inf_{v\in\mathcal{H}_N}\sqrt{1-\frac{\langle u,v\rangle^2_{\mathcal{H}}}{||u||^2_{\mathcal{H}}||v||^2_{\mathcal{H}}}} \tag{7.11}$$

Furthermore:

$$\frac{||u-\pi_{\mathrm{span}(\{v\})}(u)||^2_{\mathcal{H}}}{||u||^2_{\mathcal{H}}} = \frac{||u-\langle u,\frac{v}{||v||_{\mathcal{H}}}\rangle_{\mathcal{H}}\frac{v}{||v||_{\mathcal{H}}}||^2_{\mathcal{H}}}{||u||^2_{\mathcal{H}}} = 1-\frac{\langle u,v\rangle^2_{\mathcal{H}}}{||u||^2_{\mathcal{H}}||v||^2_{\mathcal{H}}} \tag{7.12}$$

where $\pi_{\mathrm{span}(\{v\})}(u)$ is the orthogonal projection of u on $\mathrm{span}(\{v\})$. Using Equation (7.12) in Equation (7.11) yields:

$$\sin\measuredangle_{\mathcal{H}}(u,\mathcal{H}_N) = \inf_{v\in\mathcal{H}_N}\frac{||u-\pi_{\mathrm{span}(\{v\})}(u)||_{\mathcal{H}}}{||u||_{\mathcal{H}}} = \inf_{v\in\mathcal{H}_N}\inf_{w\in\mathrm{span}(\{v\})}\frac{||u-w||_{\mathcal{H}}}{||u||_{\mathcal{H}}} \tag{7.13}$$

Finally, given that $\bigcup_{v\in\mathcal{H}_N}\mathrm{span}(\{v\}) = \mathcal{H}_N$:

$$\sin\measuredangle_{\mathcal{H}}(u,\mathcal{H}_N) = \inf_{v\in\mathcal{H}_N}\frac{||u-v||_{\mathcal{H}}}{||u||_{\mathcal{H}}} \tag{7.14}$$

which ends the proof. □

7.2 Strategies for non-reducible problems

For some problems, the Kolmogorov width slowly decays when increasing the dimension N of the approximation space. For these *non-reducible* problems, the dimension N of the approximation space giving a sufficiently small Kolmogorov width is generally too high to enable the fast computation of approximate solutions. Qualitatively, the solution manifold $\mathcal{M}$ covers too many independent directions to be embedded in a low-dimensional subspace. To address this issue, several techniques have been developed:

- Problem-specific methods tackle the difficulties of some specific physics problems that are known to be non-reducible, such as advection-dominated problems which have been largely investigated, for instance in [196, 197, 198].
- Online-adaptive model reduction methods update the ROM in the exploitation phase by collecting new information online as explained in [199], in order to limit extrapolation errors when solving the parametrized governing equations in a region of the parameter space that was not explored in the training phase. The ROM can be updated for example by querying the high-fidelity model when necessary for basis enrichment [33, 200, 201, 65, 202]. Other methods propose enrichment procedures that do not require solving the equations with the high-fidelity model, whose complexity scales linearly with ([203, 204]) or is independent of ([205]) the dimension of the high-fidelity model.
- ROM interpolation methods [206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217] use interpolation techniques on Grassmann manifolds or matrix manifolds to adapt the ROM to the parameters considered in the exploitation phase by interpolating between two precomputed ROMs.
- Dictionaries of basis vector candidates enable building a parameter-adapted ROM in the exploitation phase by selecting a few basis vectors. This technique is presented in [218, 219] for the Reduced Basis method.
- Dictionaries of ROMs rely on the construction of several local ROMs adapted to different regions of the solution manifold. These local ROMs can be obtained by partitioning the time interval [220, 221], the parameter space [220, 222, 223, 42, 202, 154, 155], or the solution space [40, 41, 42, 224, 43, 47, 225]. Local ROMs have been used both with the Reduced Basis method and the POD Galerkin method. In the same vein as online-adaptive model reduction methods, local ROBs can be adapted online using for example a low-rank SVD update method, as in [41, 224] when switching from one local ROB to another or in [202] when an error indicator detects extrapolation errors. This concept of local ROMs should not be confused with another type of local (or localized) ROMs described in [226], where the ROMs are associated to subdomains of the computational domain, in the spirit of domain decomposition techniques.
- Nonlinear manifold ROM methods [36, 37, 38] learn a nonlinear embedding and project the governing equations onto the corresponding approximation manifold, by means of a nonlinear function mapping a low-dimensional latent space to the solution space. This function is the decoder of an undercomplete autoencoder trained with the mean squared error loss to compress the snapshots and reconstruct them from

their compressed representations. In this way, the nonlinear manifold is approximated with one single nonlinear ROM. Classical linear ROMs are obtained when the autoencoder has only one hidden-layer with linear activation functions. In this case, the decoder simply returns a linear combination of the POD modes.

This thesis focuses on dictionaries of ROMs, where the solution manifold is partitioned to get a collection of subsets $\mathcal{M}_k \subset \mathcal{M}$ that can be covered by a dictionary of low-dimensional subspaces, enabling the use of linear ROMs. If $\{\mathcal{M}_k\}_{k\in[\![1;K]\!]}$ is a partition of $\mathcal{M}$, then:

$$\forall k \in [\![1;K]\!],\ \forall N \in \mathbb{N}^*, \quad d_N(\mathcal{M}_k) \leq d_N(\mathcal{M}) \tag{7.15}$$

which is also valid when replacing d_N by d'_N or $\tilde{d}_N$. For a given number K of subsets, two partitions can be compared on the basis of the ratios $d_N(\mathcal{M}_k)/d_N(\mathcal{M})$. The idea of clustering training data to define local ROBs traces back to the work of D. Amsallem, K. Washabaug, M.J. Zahr and C. Farhat in 2012, published in [40, 41] and validated on nonlinear problems in computational fluid dynamics and fluid-structure-electric interactions. In these papers, the set of snapshots is partitioned with k-means clustering to define K clusters represented by their means $\{\overline{u}_k\}_{1\leq k\leq K}$. One local ROB is computed for each cluster using the POD. In the exploitation phase, given the solution at the i-th time increment, one looks for the closest mean $\overline{u}_k$ in terms of the norm $||.||_{L^2(\Omega)}$ and computes the state of the solution at the $i+1$-th time increment with the corresponding local ROB. This technique has been used more recently in a hyper-reduction framework in [224, 225].

* *
*

Part IV

ROM-nets

Résumé

Cette partie est le coeur du mémoire, dans lequel nous présentons nos contributions à la réduction d'ordre de modèle non-linéaire par l'utilisation de méthodes d'apprentissage statistique. La méthode que nous proposons est appelée *ROM-net*, ou plus précisément *ROM-net basé sur un dictionnaire*, et a déjà été décrite dans nos cinq articles [47, 48, 49, 50, 51]. Les chapitres qui suivent reprennent d'ailleurs en grande partie des sections de ces papiers, avec quelques modifications et éléments supplémentaires. De manière générale, un ROM-net est un algorithme d'apprentissage statistique qui adapte le modèle d'ordre réduit (ROM) utilisé en fonction de l'état et de l'environnement du système physique étudié. Un ROM-net permet de prendre en compte une large gamme de configurations du système et d'en simuler le comportement de manière approchée plus rapidement qu'un modèle physique haute-fidélité, tout en conservant les équations du modèle. Les configurations (ou scénarios) étudiées correspondent potentiellement à des variabilités non-paramétrées.

Un ROM-net basé sur un dictionnaire comprend un dictionnaire de ROMs locaux et un algorithme de classification, ou classifieur. Le dictionnaire de ROMs locaux permet d'adapter le choix du modèle (et donc de l'espace d'approximation pour la résolution des équations) en fonction de la configuration du système. Chaque modèle du dictionnaire est adapté à une famille particulière de configurations du système. Le classifieur, quant à lui, sert de système de recommandation automatique de modèle : il analyse la configuration du système et sélectionne le meilleur modèle du dictionnaire pour effectuer des simulations rapides et précises. La phase d'apprentissage d'un ROM-net basé sur un dictionnaire se décompose comme suit:

- **Analyse du système :** Un modèle haute-fidélité est utilisé pour prédire le comportement physique du système pour un certain nombre de configurations différentes. Pour réduire le coût de cette étape, il est possible de considérer une version simplifiée du problème physique. Dans le cas d'application présenté à la fin du mémoire pour le calcul de durée de vie d'une aube de turbine dans un turboréacteur avec incertitudes sur le chargement thermique, le problème cible consiste à prédire le comportement mécanique de l'aube pour un grand nombre de cycles de chargement, un cycle correspondant à un vol. L'analyse du système consiste ici à simuler, pour différents chargements thermiques possibles, le comportement mécanique lors du tout premier cycle de chargement. L'information fournie par les résultats de ces simulations numériques dites simplifiées est insuffisante pour prédire une durée de vie, mais permet néanmoins d'appréhender le comportement de l'aube lorsque le chargement thermique varie. Les résultats des simulations numériques simplifiées sont appelés *snapshots simplifiés*.

- **Partitionnement des données de simulation :** Les simulations simplifiées per-

mettent d'identifier des familles (ou *clusters*) de configurations conduisant à des comportements physiques similaires. Il s'agit de l'étape de partitionnement des données (ou *clustering* en anglais). De cette manière, K clusters sont identifiés, où K est un entier choisi par l'utilisateur. Le clustering est dit *physique*, car le partitionnement de la base de données de configurations possibles se fait à l'aide de données physiques issues des simulations numériques simplifiées.

- **Construction du dictionnaire de modèles réduits locaux :** Pour chacun des K clusters, nous utilisons les méthodes classiques de réduction d'ordre de modèle pour construire un modèle réduit adapté aux membres du cluster en question.

- **Entraînement du classifieur :** Le classifieur est entraîné à recommander le meilleur modèle du dictionnaire pour une configuration donnée, grâce aux données d'entraînement labellisées lors de l'étape du clustering.

Après une présentation générale de la méthodologie, nous présentons les algorithmes développés et les solutions proposées pour optimiser les deux éléments principaux d'un ROM-net basé sur un dictionnaire, à savoir le dictionnaire de ROMs locaux et le classifieur.

Bien que couramment utilisée, la distance euclidienne entre des données issues de simulations n'est pas toujours pertinente pour faire du clustering en vue de construire un dictionnaire de ROMs locaux. Cela est expliqué en détails et illustré dans les chapitres qui suivent. Nous proposons donc une nouvelle mesure de dissimilarité adaptée à la réduction de modèle, qui joue le rôle de distance dans la procédure de clustering. Cette dissimilarité est définie de manière à regrouper dans un même cluster les données de simulation pouvant être approchées en utilisant le même espace d'approximation. Nous donnons quelques éléments théoriques permettant de relier cette dissimilarité à la distance de Hilbert-Schmidt entre deux opérateurs de projection ainsi qu'aux épaisseurs de Kolmogorov. Enfin, nous introduisons la notion de partition optimale de la variété des solutions, et démontrons que ces partitions optimales sont les solutions du problème de clustering des k-médoïdes avec la dissimilarité que nous avons proposée, donnant ainsi une justification à nos choix méthodologiques. L'efficacité de notre méthode de clustering est illustrée sur un problème de thermique non-réductible en 1D, où les performances du dictionnaire de ROMs locaux obtenu dépassent significativement les performances de dictionnaires obtenus par des méthodes alternatives. Nous définissons également un critère d'efficacité a priori d'un ROM-net basé sur un dictionnaire, permettant d'évaluer le gain potentiel apporté par un ROM-net par rapport à un unique ROM global. Ce critère est calculable assez tôt dans la phase d'entraînement, avant les étapes les plus coûteuses, et constitue donc un moyen efficace d'évaluation du compromis entre le coût et la complexité de la mise en place d'un ROM-net d'une part, et du gain en précision et en temps de calcul d'autre part. Ce critère d'efficacité fournit aussi une méthode pratique pour la calibration de certains hyperparamètres tels que le nombre de modèles du dictionnaire ou la dimension des espaces d'approximation.

Après avoir optimisé notre stratégie de construction du dictionnaire de ROMs locaux, nous nous intéressons plus particulièrement à l'entraînement du classifieur pour la recommandation automatique de modèles. Contrairement aux problèmes de classification usuels en machine learning, le problème de classification rencontré lors de la mise en place d'un ROM-net à base de dictionnaire combine trois difficultés majeures liées à la nature et au coût des données d'entraînement issues de simulations numériques. Lorsque le classifieur

doit analyser un champ physique discrétisé sur un maillage, la dimension des données à manipuler est proportionnelle à la dimension du modèle haute-fidélité, et peut donc facilement atteindre quelques millions voire plusieurs dizaines de millions sur certains cas industriels. En plus d'être de grande dimension, les données d'entraînement sont nécessairement en nombre limité en raison du coût de la procédure de labellisation des données par clustering physique, puisque cette procédure fait intervenir des simulations numériques. La grande dimension des données d'entraînement et leur nombre limité sont les deux difficultés principales qui, combinées, accentuent le risque de surapprentissage du classifieur. Le caractère physique de ces données pose une difficulté supplémentaire, puisque bon nombre d'algorithmes d'augmentation de données ne permettent pas de conserver le caractère physique des données ou ne sont tout simplement pas applicables à des champs discrétisés sur un maillage. Nous proposons donc un nouvel algorithme d'augmentation de données adapté aux problèmes de classification sur des données issues de simulations numériques. Par ailleurs, pour réduire la quantité d'informations à analyser pour une seule donnée et ainsi accélérer l'entraînement du classifieur et réduire la mémoire utilisée lors de l'augmentation de données, nous proposons une variante d'un algorithme de sélection de variables pertinentes et non-redondantes tenant compte du caractère physique de nos données pour identifier plus rapidement un jeu de variables efficace pour la classification. Les algorithmes d'augmentation de données et de sélection de variables sont évalués sur un cas test, en combinaison avec une grande variété d'algorithmes de classification usuels.

* *
*

Chapter 8

Preliminaries about ROM-nets

Abstract: *The use of ROM dictionaries introduces the need for a model selection method that identifies the most suitable model in the dictionary. In [40, 41, 224, 225], the local ROM is selected by finding the closest cluster representative from the current state of the solution with the Euclidean distance. When model selection is not straightforward and slows down the simulation process, one can use a classifier to learn the model selection task and enable fast model recommendation in the exploitation phase. Dictionaries of ROMs with automatic model recommendation made by a classifier can be found in [43, 42, 47, 154, 155]. To our knowledge, the idea of combining physics-informed clustering for the definition of local ROMs with a classifier for model recommendation came from the pioneering works of Peherstorfer, Butnaru, Willcox, and Bungartz on the Localized Discrete Empirical Interpolation Method (LDEIM [42], 2014) and of Nguyen, Barhli, Muñoz and Ryckelynck on computer vision [43] in 2018. This chapter introduces a few definitions and concepts that are further developed in the next chapters. In particular, it presents the dictionary-based ROM-net, methodology that we have developed and improved in this thesis from the works of [42] and [43].*

Remark 8.0.1. *This chapter incorporates paragraphs taken from our papers [47, 49], with some modifications.*

Contents

8.1 ROM-nets

Our objective is to predict a quantity of interest Z via the computation of a primal variable u that belongs to a reduced approximation space and satisfies nonlinear physics equations depending on a stochastic input X. Let $\mathcal{X}$ denote the set of input variabilities and let $\mathcal{Z}$ represent the set containing the quantity of interest. In structural mechanics, Z can represent a damage field, the von Mises stress in a zone of interest, or the displacement of a specific point in the structure, while X can stand for material constants, boundary conditions, geometrical parameters, a X-ray computed tomography scan characterizing the microstructure, images of defects, or even a three-dimensional field defined on the domain Ω such as a temperature field, residual stresses, or heterogeneous material parameters. The input X is described by a random variable because it contains the uncertainties on the physical system under study: when considering polycrystalline materials, X-ray computed tomography scans could be used to study macroscopic properties under microstructural variabilities such as grains' sizes, shapes and orientations. We refer the reader to [227, 228] for more details on finite-element modeling based on X-ray computed tomography scans. In the industrial application presented at the end of this thesis, X is the finite-element discretization of a temperature field whose variabilities evolve in $L^2(\Omega)$. These stochastic variabilities may be related to turbulence in a fluid-structure interaction with a high-Reynolds-number fluid flow. In aircraft engines, the temperature field in high-pressure turbine blades results from a complex turbulent flow coming from the combustion chamber. An example of fluid flow in a combustion chamber can be found in [229], where a time stable reduced-order model is built for the simulation of 3D unsteady turbulent and incompressible flow in a fuel injection system and in the primary zone of a combustion chamber in an aircraft engine.

Although X can be generated by a parametric stochastic model, it is assumed that we have no prior knowledge of the underlying model. Therefore, the proposed methodology is suitable for nonparametrized input variabilities which can represent uncertainties on the environment of the physics problem. This feature is required when the method is trained on data simulated by a parametric model, but applied to real data with unknown distributions obtained from experimental measures or from a more complex model. Applying an algorithm on data drawn from a distribution that is different from the one that generated training data is called *domain adaptation*. It is a common practice in *transfer learning*. It is recalled that the expression *nonparametrized variabilities* is adopted here to underline that the stochastic model generating X is unknown and, as a consequence, no low-dimensional parametrization describing X exactly is available. Typically, the dimension of the input X scales with the number of degrees of freedom in the high-fidelity model, when X describes a field discretized on a mesh.

When the input $X \in \mathcal{X}$ is modified, the primal variable u evolves on a manifold $\mathcal{M}$. In some situations, it is complicated to build a relevant reduced-order model giving accurate predictions for the primal variable on the whole manifold. In such cases, predictions on the quantity of interest Z are inaccurate since they derive from the behavior of the primal variable. The reduced-order model must be adapted to the input to capture nonlinearities.

Let us introduce the notation $V(\mathcal{H})$ representing the set of all the possible reduced-order models, where a reduced-order model is defined by a reduced-order basis (basis of a vector subspace of the Hilbert space $\mathcal{H}$) and, optionally, by some parameters related to a hyper-reduction algorithm. Given two sets $\mathcal{A}$ and $\mathcal{B}$, the notation $\mathcal{B}^{\mathcal{A}}$ represents the set

of functions $f : \mathcal{A} \to \mathcal{B}$. Let us now give the definitions of a *projection-based reduced-order solver* and a *ROM-net*:

Definition 8.1.1 (Projection-based reduced-order solver)**.** *Let us consider a physics problem parametrized by $X \in \mathcal{X}$. Let $Z \in \mathcal{Z}$ be a quantity of interest of this physics problem. A projection-based reduced-order solver is an operator $\mathcal{S} : V(\mathcal{H}) \to \mathcal{Z}^{\mathcal{X}}$ taking a reduced-order model $m \in V(\mathcal{H})$ as an input and returning a predictor $\mathcal{S}[m] : \mathcal{X} \to \mathcal{Z}$ for the quantity of interest. The operator $\mathcal{S}[m]$ uses Galerkin projection onto a reduced-order basis to compute an approximate solution of the governing equations, and computes the quantity of interest associated to this approximate solution. Given $X \in \mathcal{X}$, the quantity of interest Z can be approximated by:*

$$\tilde{Z} := \mathcal{S}[m](X) \tag{8.1}$$

Definition 8.1.2 (ROM-net)**.** *Let us consider a physics problem parametrized by $X \in \mathcal{X}$, where a quantity of interest $Z \in \mathcal{Z}$ can be predicted by a projection-based reduced-order solver $\mathcal{S} : V(\mathcal{H}) \to \mathcal{Z}^{\mathcal{X}}$. A ROM-net $\mathcal{R} : \mathcal{X} \to V(\mathcal{H})$ is a machine learning algorithm returning a reduced-order model $\mathcal{R}(X) \in V(\mathcal{H})$ adapted to the input $X \in \mathcal{X}$. Given $X \in \mathcal{X}$, the quantity of interest Z can be approximated by:*

$$\tilde{Z} := \mathcal{S}[\mathcal{R}(X)](X) \tag{8.2}$$

Contrary to metamodeling, using a reduced-order model $\mathcal{R}(X)$ enables satisfying homogeneous Dirichlet boundary conditions and solving the constitutive equations at least at some specific points if hyper-reduction is used. Hence, a ROM-net provides a hybrid approach mixing physics-based modeling and machine learning. It is noteworthy that the definition of the quantity of interest remains quite flexible after the training of a ROM-net. In solid mechanics for instance, the definition of the damage indicator of an uncoupled damage model can be changed without restarting the training phase. Using physics knowledge through reduced-order models also facilitates transfer learning: indeed, in the exploitation phase, parameters that are not included in the variable X can also be slightly modified, as long as their variabilities do not induce highly nonlinear variations of the outputs. Solving physics equations is supposed to give more robustness to parameter changes.

Remark 8.1.3. *The Definition 8.1.2 of a ROM-net has been slightly modified with respect to the original definition we gave in [47]. Indeed, in [47], the definition specifies that a ROM-net is a deep learning algorithm. It has been chosen to change this definition to make it more general in order to include other related works.*

8.2 Dictionary-based ROM-nets

When the solution manifold $\mathcal{M}$ is embedded in a low-dimensional vector space, one can construct a single global reduced-order model in order to compute approximate solutions of the physics problem for different points in the parameter space $\mathcal{X}$. When the solution manifold $\mathcal{M}$ is not embedded in a low-dimensional vector space, using one single global reduced-order model would result in either time-consuming or inaccurate reduced simulations, depending on the number of modes selected in the reduced-order basis. By

partitioning the parameter space $\mathcal{X}$, one can define a dictionary of local reduced-order models which enables approximating $\mathcal{M}$ by several affine subspaces, as mentioned in Section 7.2. Clustering algorithms can be used to split the set $\mathcal{X}$ into distinct clusters. Inputs belonging to the same cluster lead to solutions which can be predicted with the same local reduced-order model because of their proximity on the manifold $\mathcal{M}$.

The dictionary of local reduced-order models contains $K \geq 2$ cluster-specific reduced-order models. Hence, for a given input $X \in \mathcal{X}$, one must identify the corresponding cluster to select the most appropriate reduced-order model.

Definition 8.2.1 (Dictionary of reduced-order models)**.** *Given an integer $K \geq 2$, an injective function $\mathcal{D}_K : [\![1; K]\!] \to V(\mathcal{H})$ is called a dictionary of reduced-order models of dimension K, or K-ROM-dictionary.*

Definition 8.2.2 (Dictionary-based ROM-net)**.** *Let us consider a physics problem parametrized by $X \in \mathcal{X}$, where a quantity of interest $Z \in \mathcal{Z}$ can be predicted by a projection-based reduced-order solver $\mathcal{S} : V(\mathcal{H}) \to \mathcal{Z}^{\mathcal{X}}$. Given an integer $K \geq 2$, a classifier $\mathcal{C}_K : \mathcal{X} \to [\![1; K]\!]$ and a K-ROM-dictionary $\mathcal{D}_K : [\![1; K]\!] \to V(\mathcal{H})$, a dictionary-based ROM-net $\mathcal{R}_K$ is defined by:*

$$\forall X \in \mathcal{X}, \qquad \mathcal{R}_K(X) = \mathcal{D}_K \circ \mathcal{C}_K(X) \tag{8.3}$$

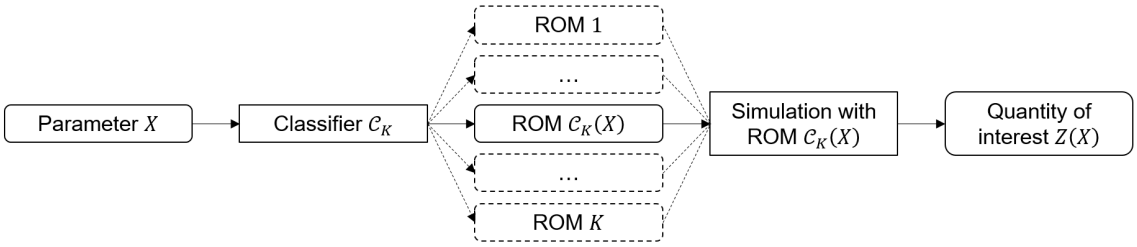

Figure 8.1: Exploitation phase of a dictionary-based ROM-net. K local ROMs are combined with a classifier $\mathcal{C}_K$ for automatic model recommendation.

Figure 8.1 illustrates the concept of dictionary-based ROM-nets. The classifier $\mathcal{C}_K$ solves a multiclass classification problem to recommend a suitable reduced-order model from the dictionary. Nevertheless, as the classes are given by a clustering algorithm, one could wonder why a classifier is used. When using a representative-based clustering algorithm with dissimilarity measure δ, each cluster $\mathcal{X}_k$ is represented by a center $\tilde{x}_k$. In theory, one could compute the dissimilarities between the new input x and all the clusters' representatives $\tilde{x}_k$, and then select the cluster with the smallest dissimilarity $\delta(x, \tilde{x}_k)$. However, this procedure is not reasonable when repeated many times, because of the computation time required to evaluate the dissimilarities. Indeed, as further explained later, dissimilarity measures that are suitable for model order reduction applications may involve numerical simulations. The clustering procedure of dictionary-based ROM-nets is actually a physics-informed clustering procedure. Hence, the time saving obtained by model order reduction would be counterbalanced by the time-consuming operations required for model selection. The *perfect classifier* defined by:

$$\mathcal{K}_K(X) = \underset{k \in [\![1;K]\!]}{\arg\min}(\delta(X, \tilde{x}_k)) \tag{8.4}$$

is too expensive because it is based on numerical simulations. When using the ROM-net, the perfect classifier $\mathcal{K}_K$ is replaced by the approximate (or *real*) classifier $\mathcal{C}_K$ to bypass the computations required for model recommendation.

8.3 Overview of the training procedure

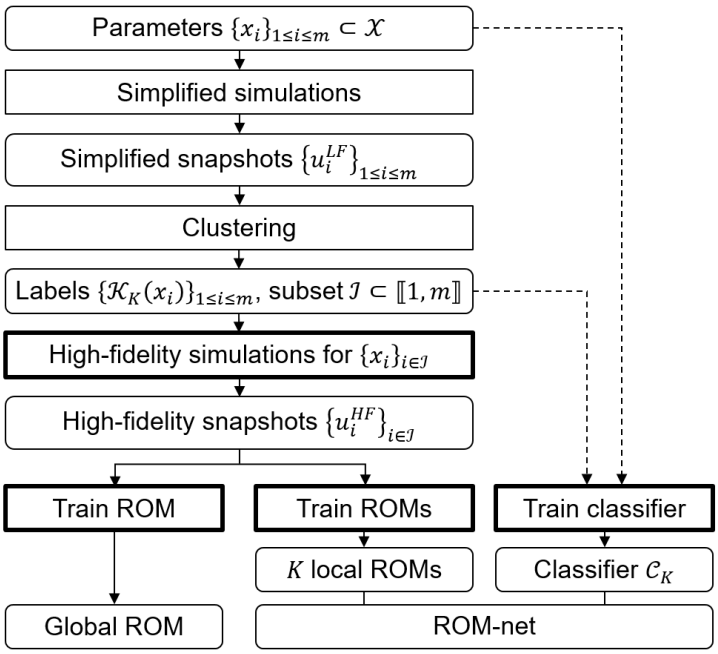

Figure 8.2: Training phases of a dictionary-based ROM-net and a global ROM with a physics-informed clustering strategy.

A dictionary-based ROM-net consists in a dictionary of local ROMs and a classifier acting as a model selector, which enables the automatic adaptation of the ROM to the state and the environment of the physical system. The ROM-net's classifier (*real classifier* denoted by $\mathcal{C}_K$ where K is the number of local ROMs) approximates the theoretical *perfect classifier* $\mathcal{K}_K$ returning the index of the best local ROM for a given point in the parameter space. For simplicity, the phrases *dictionary-based ROM-nets* and *ROM-nets* will be used interchangeably.

Figure 8.2 gives the main steps of the training phase of a dictionary-based ROM-net and draws a comparison with the construction of a global ROM benefiting from the ROM-net's physics-informed cluster analysis. The dictionary of local ROMs is built from clusters given by a physics-informed clustering procedure. First, a simplified version of the physics problem is solved for each input example of the training database. The simplified physics problem must be less computationally demanding than the target problem. In particular, it can be solved with a coarse mesh to reduce the dimension of the approximation space. The simplified simulations provide what we call *simplified snapshots*: these snapshots cannot be exploited to build ROMs, but they give information about how the physical system reacts to parameter changes. The clustering algorithm finds clusters from the information contained in these simplified snapshots. In light of the clustering results, one must identify a few relevant training examples for which the target problem is solved to get *high-fidelity snapshots*, that is, snapshots that well represent the solution manifold and can then be used for the construction of the local ROMs. The different needs in terms of training data for reduced-order modeling and machine learning can be seen through these two families of snapshots: the clustering and classification algorithms use information related to the simplified snapshots to get a sufficiently large training set, while the ROMs

use a limited number of high-fidelity snapshots in order to learn to make predictions in a physics problem. This distinction between these two types of simulation data is essential when considering complex problems with many degrees of freedom. The three major differences between our work and the seminal works of [43] and [42] are the use of simplified simulations, the clustering strategy with a new ROM-oriented dissimilarity measure, and an a priori efficiency criterion introduced hereinafter. In addition, high-dimensional nonparametrized variabilities are considered in this thesis. It is noteworthy that, among the four variants of the LDEIM, the parameter-based LDEIM with clustering of snapshots (section 4.2. of [42]) is the one that shares the more similarities with our work, since it applies clustering on simulation data and uses the parameters as inputs for the classifier.

When using a dictionary-based ROM-net, a natural question arises:

- Which dissimilarity measure and clustering algorithm should be used for model order reduction purposes?

After clustering, the training phase of the dictionary-based ROM-net still includes expensive steps corresponding to boxes with thick lines in Figure 8.2, namely the computation of the high-fidelity snapshots, the construction of the local ROMs (which can involve a hyper-reduction algorithm), and the training of a classifier for automatic model recommendation. Therefore, an evaluation criterion is needed in order to assess the quality of the clusters before continuing the ROM-net's training phase, *i.e.* right after the clustering step, see Figure 8.2. This criterion should enable the evaluation of the profitability of the ROM-net and the tuning of clustering hyperparameters, using the simplified snapshots only. Put briefly, in addition to the aforementioned question, we must also address the following issues:

- Is it possible to define a simple practical method to select good hyperparameters (number of clusters, number of POD modes, number of high-fidelity snapshots)?
- Can one define an efficiency criterion indicating whether it is worth continuing the ROM-net's training after the clustering step? This efficiency criterion would enable choosing between a ROM-net and a single global ROM by evaluating the balance between the benefits of using a ROM-net and its training cost.

These questions are answered in the two next chapters.

* *
*

Chapter 9

Physics-informed clustering procedure

Abstract: *This chapter introduces a physics-informed clustering strategy for the construction of dictionaries of local ROBs. Physics-informed cluster analysis consists in clustering the parameter space of the parametrized physics problem under study using knowledge about the behavior of the physical system. In particular, physics-informed representative-based clustering algorithms rely on dissimilarity measures involving physical quantities obtained when solving the physics problem. In other words, clusters in the parameter space are implicitly defined as the preimages of clusters found in a database of numerical simulation results. The dissimilarity measure introduced in this chapter is designed for model order reduction purposes, and can be computed either on the solution or on a quantity of interest.*

Note: The work presented in this chapter follows up the internship of Ali Ketata, that I had the pleasure of supervising at Safran in 2020.

Remark 9.0.1. *This chapter is taken from our papers [49, 51], with some modifications.*

Remark 9.0.2. *As explained in Section 6.3.1 introducing the POD, the time t is not included in the parameter x for time-dependent problems, because it is not considered as a clustering variable. A simulation for a given point x in the parameter space gives a solution u that provides several snapshots in time for the construction of a reduced-order model.*

Contents

9.1 Drawbacks of the Euclidean distance

When using a clustering algorithm to partition the solution manifold, the quality of the partition is related to the choice of the clustering method and the dissimilarity measure δ used to group similar solutions on the manifold. Among physics-informed clustering strategies, *i.e.* strategies incorporating simulation data to compute dissimilarities, [40, 41, 224, 43, 225] used k-means with Euclidean distances in the solution space or in a subspace of the solution space found by PCA, we used the Grassmann distance between subspaces spanned by the trajectories of the solutions in [47] (idea that was also used later in [230] and in our paper [48]), and [42] proposed working on the governing equations' nonlinear term, using either a variant of k-means with the DEIM [165] residual as clustering criterion or k-means on a low-dimensional representation of the governing equations' nonlinear term obtained by a DEIM-based feature selection. It is recalled that k-means is a representative-based clustering algorithm equipped with the Euclidean distance, and that changing this distance leads to other clustering methods. The Local Decomposition Method [6] also relies on a physics-informed clustering strategy even though no dissimilarity measure is used, because a Gaussian mixture model is applied to shock sensors computed from the field of a quantity of interest, which enables separating subsonic and transonic flows in computational fluid dynamics.

As suggested by Equation (7.9) of Property 7.1.3, the dissimilarity measure should be defined as a function of the angle between elements of the solution manifold in order to focus on the shape of the fields $u \in \mathcal{M}$ rather than their intensities. In this way, clustering would efficiently decrease projection errors by limiting the maximum angular deviations within clusters. The Euclidean distance $||u - v||_{\mathcal{H}}$ used in [40, 41, 224, 43, 225] does not always ensure the reduction of projection errors. Indeed, the solution manifold can contain solutions that are relatively close in terms of the Euclidean distance but distributed in many different directions of the space $\mathcal{H}$. On the other hand, having a subset $\mathcal{M}_k$ with a large diameter in terms of the Euclidean distance is not a problem if it is embedded in a low-dimensional space, as indicated by Property 7.1.3. Let us suppose that the solution manifold contains two elements u and v having disjoint supports $\text{supp}(u)$ and $\text{supp}(v)$ and such that there exists a large real number λ such that λu is still in the solution manifold (see Figure 9.1). The elements u and λu are aligned in the same direction and could then be obtained with the same 1-dimensional approximation space. However, if λ is large enough, the distance $||u - \lambda u||_{\mathcal{H}}$ can be very large with respect to $||u - v||_{\mathcal{H}}$. In this case, it is possible to assign u and v to the same cluster while assigning λu to another, whereas u and λu are aligned along a direction that is orthogonal to v. For these reasons, the Euclidean distance does not seem to be adapted, except if the number K of clusters is large enough to get very local subsets $\mathcal{M}_k$ with restricted angular deviations.

A more natural and straightforward approach would consist in clustering the parameter space $\mathcal{X}$ to define the subsets $\mathcal{M}_k = u(\mathcal{X}_k)$ for each cluster $\mathcal{X}_k$. This strategy may not be

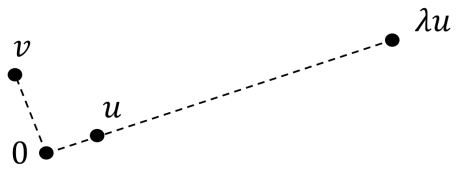

Figure 9.1: Clustering with the Euclidean distance would assign u and v to the same cluster and λu to another, whereas u and λu could be computed with the same 1D approximation space.

appropriate when u is a nonlinear function of the parameters $x \in \mathcal{X}$. The physics of the underlying problem can also generate situations where small changes of the parameters in some directions of the parameter space totally modifies the shape of the solution in a nonlinear way, while large variations in other directions of the parameter space only imply linear variations. An example is given in [47], where it is shown that clusters identified in the parameter space give subsets $\mathcal{M}_k$ spreading all over the solution manifold $\mathcal{M}$. To avoid this issue, it is preferable to apply a physics-informed clustering strategy by partitioning the solution manifold directly with an appropriate dissimilarity measure δ. Contrary to [40, 41, 224, 225], the method presented in this thesis does not enable changing the selected local ROB according to the solution's state in time-dependent problems. Nevertheless, we introduce a dissimilarity measure that improves the quality of the local ROBs on problems where the Euclidean distance $||.||_{\mathcal{H}}$ in the solution space fails to define good approximation spaces.

9.2 The dissimilarity measure

9.2.1 Definitions and general properties

Clustering is the task of splitting a database into several clusters of similar data points. For many clustering approaches, the degree of similarity between two points is quantified by a dissimilarity measure. This section introduces the dissimilarity measure used in this thesis and gives some of its properties. It is important to stress that this dissimilarity is computed from the simplified snapshots given by the simplified simulations. Hence, in this section, the notation $u \in L^2(\Omega \times [0; t_f])$ represents a simplified snapshot.

Definition 9.2.1 (Principal angles between subspaces)**.** *Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be two subspaces of $L^2(\Omega)$. The principal angles or canonical angles $\theta_k(\mathcal{V}_1, \mathcal{V}_2) \in [0; \pi/2]$ between $\mathcal{V}_1$ and $\mathcal{V}_2$ are defined by:*

$$\forall k \in \mathbb{N}^*, \quad \theta_k(\mathcal{V}_1, \mathcal{V}_2) := \measuredangle(v_1^k, v_2^k) \tag{9.1}$$

where the angle $\measuredangle := \measuredangle_{L^2(\Omega)}$ is measured in $L^2(\Omega)$ (see Equation (7.7)), and where the vectors $v_1^k \in \mathcal{V}_1$ and $v_2^k \in \mathcal{V}_2$ are given by the following sequence of optimization problems:

$$\left\{ \begin{array}{rcl} (v_1^1, v_2^1) & \in & \underset{(v_1,v_2)\in\mathcal{V}_1\times\mathcal{V}_2}{\arg\min} \ \measuredangle(v_1, v_2) \\ (v_1^{k+1}, v_2^{k+1}) & \in & \arg\min \left\{ \measuredangle(v_1, v_2) \,\middle|\, v_j \in \mathcal{V}_j \cap \left(\mathrm{span}(\{v_j^i\}_{1\leq i\leq k})\right)^{\perp}, \ j \in \{1;2\} \right\} \end{array} \right. \tag{9.2}$$

with the notation $\mathcal{V}^{\perp}$ denoting the orthogonal complement of $\mathcal{V} \subset L^2(\Omega)$ in $L^2(\Omega)$.

In practice, when the spaces $\mathcal{V}_1$ and $\mathcal{V}_2$ are finite-dimensional, it can be shown (see Theorem 1 of [231]) that the principal angles are given by:

$$\forall k \in [\![1; \min(\dim(\mathcal{V}_1), \dim(\mathcal{V}_2))]\!], \quad \theta_k(\mathcal{V}_1, \mathcal{V}_2) = \arccos \sigma_k \tag{9.3}$$

with $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{\min(\dim(\mathcal{V}_1),\dim(\mathcal{V}_2))}$ being the singular values of the matrix $\mathbf{C}(\mathcal{V}_1, \mathcal{V}_2) \in \mathbb{R}^{\dim(\mathcal{V}_1)\times\dim(\mathcal{V}_2)}$ defined by:

$$C_{ij}(\mathcal{V}_1, \mathcal{V}_2) := \langle \psi_i^{(1)}, \psi_j^{(2)} \rangle_{L^2(\Omega)} \tag{9.4}$$

where the functions $\psi_i^{(1)}$ (resp. $\psi_j^{(2)}$) form an orthonormal basis of $\mathcal{V}_1$ (resp. $\mathcal{V}_2$). The vector $\boldsymbol{\theta}(\mathcal{V}_1, \mathcal{V}_2)$ denotes the vector containing the principal angles between the spaces $\mathcal{V}_1$ and $\mathcal{V}_2$.

Definition 9.2.2 (n-dimensional elementary basis)**.** *Let $u \in L^2(\Omega\times[0;t_f])$ and $n \in [\![1;\mathcal{N}]\!]$. The n-dimensional elementary basis associated to u is the orthonormal n-frame $\Psi_n(u) \in V(n, L^2(\Omega))$ obtained by solving the POD minimization problem given in Equation (6.9) with the Snapshot POD algorithm, using the trajectory of u over time as a snapshot.*

Definition 9.2.3 (n-dimensional elementary approximation space)**.** *Let $u \in L^2(\Omega\times[0;t_f])$ and $n \in [\![1;\mathcal{N}]\!]$. The n-dimensional elementary approximation space $\mathcal{V}_n(u) \in \mathrm{Gr}(n, L^2(\Omega))$ is the subspace spanned by $\Psi_n(u)$.*

In Definition 9.2.2, the POD basis $\Psi_n(u)$ is used for clustering only, it is not supposed to be used for numerical simulations since it is computed from simplified snapshots. Qualitatively, the subspace $\mathcal{V}_n(u)$ spanned by this POD basis is the best n-dimensional approximation space for the trajectory of $u(.,t)$ in $L^2(\Omega)$, that is to say:

$$\mathcal{V}_n(u) = \underset{\mathcal{V}_n \in \mathrm{Gr}(n,L^2(\Omega))}{\arg\min} \int_0^{t_f} \inf_{v\in\mathcal{V}_n} ||u(.,t) - v||^2_{L^2(\Omega)} \, dt \tag{9.5}$$

Definition 9.2.4 (Chordal distance between subspaces [232], p. 140, Section 2)**.** *Let $\mathcal{H}$ be a Hilbert space, and n, m be two integers with $n \leq m$. The chordal distance between subspaces $\mathcal{V}_1 \in \mathrm{Gr}(n, \mathcal{H})$ and $\mathcal{V}_2 \in \mathrm{Gr}(m, \mathcal{H})$ is defined by:*

$$d_c(\mathcal{V}_1, \mathcal{V}_2) := ||\sin \boldsymbol{\theta}(\mathcal{V}_1, \mathcal{V}_2)||_2 = \left(\sum_{k=1}^{n} \sin^2 \theta_k(\mathcal{V}_1, \mathcal{V}_2) \right)^{1/2} \tag{9.6}$$

Definition 9.2.5 (Sine dissimilarity between functions)**.** *Given $n \in [\![1;\mathcal{N}]\!]$, the sine dissimilarity $\tilde{\delta}_n$ between functions u and v in $L^2(\Omega \times [0;t_f])$ is defined by:*

$$\tilde{\delta}_n(u, v) := d_c(\mathcal{V}_n(u), \mathcal{V}_n(v)) \tag{9.7}$$

Let us now recall the definition of the orthogonal projection $\pi_{\mathcal{H}_n} : L^2(\Omega) \to L^2(\Omega)$ on a n-dimensional subspace $\mathcal{H}_n$ of $L^2(\Omega)$, with an orthonormal basis $\{\psi_k\}_{1\leq k\leq n}$:

$$\forall u \in L^2(\Omega), \quad \pi_{\mathcal{H}_n}(u) = \sum_{k=1}^{n} \langle u, \psi_k \rangle_{L^2(\Omega)} \psi_k \tag{9.8}$$

Property 9.2.6 (Sine dissimilarity and L^2 projection errors)**.** *For all $n \in [\![1; \mathcal{N}]\!]$, the sine dissimilarity is symmetric and satisfies:*

$$\forall (u,v) \in L^2(\Omega \times [0;t_f])^2, \quad \tilde{\delta}_n(u,v) = \left(\sum_{i=1}^{n} ||\psi_i(u) - \pi_{\mathcal{V}_n(v)}(\psi_i(u))||^2_{L^2(\Omega)} \right)^{1/2} \tag{9.9}$$

with $\pi_{\mathcal{V}_n(v)}$ denoting the orthogonal projection on $\mathcal{V}_n(v)$ and where the functions $\psi_i(u) \in L^2(\Omega)$ for $i \in [\![1;n]\!]$ are the vectors of the elementary basis $\Psi_n(u)$.

Proof. Let us first develop the square of the right-hand side of Equation (9.9), denoted by $f_n(u,v)^2$, using Equation (9.8), the bilinearity of the L^2 inner product and the orthonormality of the bases $\Psi_n(u)$ and $\Psi_n(v)$:

$$\begin{aligned}
f_n(u,v)^2 &= \sum_{i=1}^{n} ||\psi_i(u)||^2_{L^2(\Omega)} - 2\sum_{i=1}^{n}\sum_{j=1}^{n} \langle \psi_i(u), \psi_j(v) \rangle^2_{L^2(\Omega)} \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \langle \psi_i(u), \psi_j(v) \rangle_{L^2(\Omega)} \langle \psi_i(u), \psi_k(v) \rangle_{L^2(\Omega)} \langle \psi_j(v), \psi_k(v) \rangle_{L^2(\Omega)} \\
&= n - 2\sum_{i=1}^{n}\sum_{j=1}^{n} \langle \psi_i(u), \psi_j(v) \rangle^2_{L^2(\Omega)} \\
&\quad + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \langle \psi_i(u), \psi_j(v) \rangle_{L^2(\Omega)} \langle \psi_i(u), \psi_k(v) \rangle_{L^2(\Omega)} \delta_{jk} \\
&= n - \sum_{i=1}^{n}\sum_{j=1}^{n} \langle \psi_i(u), \psi_j(v) \rangle^2_{L^2(\Omega)}
\end{aligned} \tag{9.10}$$

where δ_{jk} is the Kronecker delta function. Let $\mathbf{C} \in \mathbb{R}^{n \times n}$ be the matrix whose entries are the inner products $\langle \psi_i(u), \psi_j(v) \rangle_{L^2(\Omega)}$. Its SVD reads $\mathbf{C} = \mathbf{V} \cos \mathbf{\Theta}\ \mathbf{W}^T$ where $\mathbf{\Theta}$ is a diagonal matrix containing the principal angles $\theta_k(\mathcal{V}_n(u), \mathcal{V}_n(v))$, and where $\mathbf{V}$ and $\mathbf{W}$ are orthogonal matrices. Then:

$$\begin{aligned}
f_n(u,v)^2 &= n - \mathrm{tr}(\mathbf{C}^T\mathbf{C}) \\
&= n - \mathrm{tr}\left(\mathbf{W} \cos(\mathbf{\Theta})^T \mathbf{V}^T \mathbf{V} \cos(\mathbf{\Theta}) \mathbf{W}^T \right) \\
&= n - \mathrm{tr}\left(\mathbf{W}^T\mathbf{W} \cos(\mathbf{\Theta})^T \mathbf{V}^T \mathbf{V} \cos(\mathbf{\Theta}) \right) \\
&= n - \mathrm{tr}\left(\cos(\mathbf{\Theta})^T \cos(\mathbf{\Theta}) \right) \\
&= n - \sum_{k=1}^{n} \cos^2 \theta_k(\mathcal{V}_n(u), \mathcal{V}_n(v)) \\
&= \sum_{k=1}^{n} 1 - \cos^2 \theta_k(\mathcal{V}_n(u), \mathcal{V}_n(v)) \\
&= \sum_{k=1}^{n} \sin^2 \theta_k(\mathcal{V}_n(u), \mathcal{V}_n(v)) \\
&= \tilde{\delta}_n(u,v)^2
\end{aligned} \tag{9.11}$$

These equations remain true when exchanging u and v, which ends the proof. □

Property 9.2.7 (Sine dissimilarity and Hilbert-Schmidt distance)**.** *For all* $n \in [\![1;\mathcal{N}]\!]$*, for all* $(u,v) \in L^2(\Omega \times [0;t_f])^2$*, the sine dissimilarity satisfies:*

$$\tilde{\delta}_n(u,v) = \frac{1}{\sqrt{2}}||\pi_{\mathcal{V}_n(u)} - \pi_{\mathcal{V}_n(v)}||_{HS(L^2(\Omega))} \tag{9.12}$$

Proof. Since the Hilbert-Schmidt inner product on $HS(L^2(\Omega))$ does not depend on the choice of the orthonormal basis of $L^2(\Omega)$, let us choose a basis that is relevant for calculations. For $u \in L^2(\Omega \times [0;t_f])$, the n-dimensional elementary basis $\Psi_n(u)$ is completed with an orthonormal basis of the orthogonal complement of $\mathcal{V}_n(u)$ in $L^2(\Omega)$. The resulting orthonormal basis of $L^2(\Omega)$ is denoted by $\{\psi_k(u)\}_{k\in\mathbb{N}^*}$, where the n first basis vectors are those of the basis $\Psi_n(u)$. Let us now expand the term $||\pi_{\mathcal{V}_n(u)} - \pi_{\mathcal{V}_n(v)}||^2_{HS(L^2(\Omega))}$:

$$||\pi_{\mathcal{V}_n(u)} - \pi_{\mathcal{V}_n(v)}||^2_{HS(L^2(\Omega))} = ||\pi_{\mathcal{V}_n(u)}||^2_{HS(L^2(\Omega))} + ||\pi_{\mathcal{V}_n(v)}||^2_{HS(L^2(\Omega))} - 2\langle \pi_{\mathcal{V}_n(u)}, \pi_{\mathcal{V}_n(v)} \rangle_{HS(L^2(\Omega))} \tag{9.13}$$

Using the definition of the Hilbert-Schmidt inner product given by Equation (6.5), one has:

$$\begin{aligned} \langle \pi_{\mathcal{V}_n(u)}, \pi_{\mathcal{V}_n(v)} \rangle_{HS(L^2(\Omega))} &= \sum_{i=1}^{\infty} \langle \pi_{\mathcal{V}_n(u)}(\psi_i(u)), \pi_{\mathcal{V}_n(v)}(\psi_i(u)) \rangle_{L^2(\Omega)} \\ &= \sum_{i=1}^{n} \langle \psi_i(u), \pi_{\mathcal{V}_n(v)}(\psi_i(u)) \rangle_{L^2(\Omega)} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \psi_i(u), \psi_j(v) \rangle^2_{L^2(\Omega)} \end{aligned} \tag{9.14}$$

where the last equality results from the expression of $\pi_{\mathcal{V}_n(v)}(\psi_i(u))$ given by Equation (9.8). Furthermore:

$$\begin{aligned} ||\pi_{\mathcal{V}_n(u)}||^2_{HS(L^2(\Omega))} &= \sum_{i=1}^{\infty} \langle \pi_{\mathcal{V}_n(u)}(\psi_i(u)), \pi_{\mathcal{V}_n(u)}(\psi_i(u)) \rangle_{L^2(\Omega)} \\ &= \sum_{i=1}^{n} \langle \psi_i(u), \psi_i(u) \rangle_{L^2(\Omega)} \\ &= n \end{aligned} \tag{9.15}$$

Similarly, one can prove that $||\pi_{\mathcal{V}_n(v)}||^2_{HS(L^2(\Omega))} = n$. Finally:

$$||\pi_{\mathcal{V}_n(u)} - \pi_{\mathcal{V}_n(v)}||^2_{HS(L^2(\Omega))} = 2n - 2\sum_{i=1}^{n}\sum_{j=1}^{n} \langle \psi_i(u), \psi_j(v) \rangle^2_{L^2(\Omega)} = 2f_n(u,v)^2 = 2\tilde{\delta}_n(u,v)^2 \tag{9.16}$$

where $f_n(u,v)$ was introduced in the proof of Property 9.2.6. □

Property 9.2.8. *For all* $n \in [\![1;\mathcal{N}]\!]$*, the sine dissimilarity is a pseudometric on* $L^2(\Omega \times [0;t_f])$*.*

Proof. The sine dissimilarity $\tilde{\delta}_n$ is nonnegative and symmetric. Equation (9.12) implies that for all $u \in L^2(\Omega \times [0;t_f])$, $\tilde{\delta}_n(u,u) = 0$. Equation (9.12) also yields:

$$\tilde{\delta}_n(u,v) = \frac{1}{\sqrt{2}}||\pi_{\mathcal{V}_n(u)} - \pi_{\mathcal{V}_n(v)}||_{HS(L^2(\Omega))} = \frac{1}{\sqrt{2}}||\pi_{\mathcal{V}_n(u)} - \pi_{\mathcal{V}_n(w)} + \pi_{\mathcal{V}_n(w)} - \pi_{\mathcal{V}_n(v)}||_{HS(L^2(\Omega))} \tag{9.17}$$

so the triangle inequality on the Hilbert-Schmidt norm gives the triangle inequality:

$$\tilde{\delta}_n(u,v) \leq \tilde{\delta}_n(u,w) + \tilde{\delta}_n(w,v).$$

□

Note that $\tilde{\delta}_n(u,v) = 0$ does not imply that $u = v$, which is the reason why the sine dissimilarity is not a metric on $L^2(\Omega \times [0;t_f])$. This is not a problem since we want this dissimilarity measure to be zero for all pairs of functions $(u,v) \in L^2(\Omega \times [0;t_f])^2$ whose trajectories over time in $L^2(\Omega)$ give the same POD approximation space, as explained in Section 9.1. The next property shows the link between the sine dissimilarity and the Grassmann dissimilarity used in [47] for dictionary-based ROM-nets:

Property 9.2.9 (Equivalence with the Grassmann dissimilarity for small angles)**.** *Given $n \in [\![1;\mathcal{N}]\!]$, let $\boldsymbol{\theta}_n$ denote the vector of principal angles between $\mathcal{V}_n(u)$ and $\mathcal{V}_n(v)$ for two square-integrable functions u and v. As $||\boldsymbol{\theta}_n||_2$ tends towards zero, the sine dissimilarity is asymptotically equivalent to the Grassmann dissimilarity $||\boldsymbol{\theta}_n||_2$, that is, using Bachmann-Landau notations:*

$$\tilde{\delta}_n(u,v) \underset{||\boldsymbol{\theta}_n||_2 \to 0}{\sim} ||\boldsymbol{\theta}_n||_2 \tag{9.18}$$

Proof. One must show that:

$$||\sin \boldsymbol{\theta}_n||_2 = ||\boldsymbol{\theta}_n||_2 + o(||\boldsymbol{\theta}_n||_2) \tag{9.19}$$

As $||\boldsymbol{\theta}_n||_2$ tends towards zero:

$$||\sin \boldsymbol{\theta}_n||_2 = \left(\sum_{i=1}^{n} \sin^2 \theta_{n,k}\right)^{1/2} = \left(\sum_{i=1}^{n} (\theta_{n,k} + o(\theta_{n,k}^2))^2\right)^{1/2} = \left(||\boldsymbol{\theta}_n||_2^2 + o(||\boldsymbol{\theta}_n||_2^2)\right)^{1/2} \tag{9.20}$$

which gives:

$$||\sin \boldsymbol{\theta}_n||_2 = ||\boldsymbol{\theta}_n||_2 \sqrt{1 + o(1)} = ||\boldsymbol{\theta}_n||_2 + o(||\boldsymbol{\theta}_n||_2) \tag{9.21}$$

□

Definition 9.2.10 (ROM-oriented dissimilarity between parameters)**.** *Given $n \in [\![1;\mathcal{N}]\!]$, the ROM-oriented dissimilarity between parameters x and x' in $\mathcal{X}$ is defined by:*

$$\delta_n\left(x,x'\right) := \tilde{\delta}_n\left(u\left(x\right), u\left(x'\right)\right) \tag{9.22}$$

where $u : \mathcal{X} \to L^2(\Omega \times [0;t_f])$ is either the primal variable (i.e. the solution of the physics problem) or a dual variable (i.e. an internal variable) defining a quantity of interest.

It is recalled that this dissimilarity is computed from simplified snapshots. Property 9.2.8 implies that the ROM-oriented dissimilarity is a pseudometric on $\mathcal{X}$. Several variants of this dissimilarity can be obtained according to the definition of the variable u. Using the primal variable should improve the quality of the POD Galerkin approximation, since the data would be clustered according to the angles between the subspaces spanned by the trajectories of the primal solution. This would give a *method-oriented* dissimilarity, that is, a dissimilarity favoring the accuracy of the numerical method (namely model order reduction) used for numerical simulations. Using a dual variable instead would improve the quality of the Gappy POD [46] reconstruction for the quantity of interest when hyper-reduction is used. This would define a *goal-oriented* method favoring the accuracy of numerical predictions of a quantity of interest. Of course, one could mix both strategies by taking a weighted average of these two variants of the ROM-oriented dissimilarity.

9.2.2 Case $n = 1$

In this section, the Hilbert space $\mathcal{H}$ is a subspace of $L^2(\Omega)$. We denote by $\|\cdot\|_{\mathcal{H}}$ the norm induced by the inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ of $\mathcal{H}$.

Definition 9.2.11 (Relative projection error)**.** *Let $u \in \mathcal{H} \setminus \{0\}$, and $\Psi_N = \{\psi_k\}_{1\leq 1\leq N} \in V(N,\mathcal{H})$ be an orthonormal reduced-order basis of dimension $N \in \mathbb{N}^*$ in $\mathcal{H}$. The relative projection error $\eta(u,\Psi_N)_{\mathcal{H}}$ of u on $\mathrm{span}(\Psi_N)$ is given by:*

$$\eta(u,\Psi_N)_{\mathcal{H}} := \frac{||u - \pi_{\mathrm{span}(\Psi_N)}(u)||_{\mathcal{H}}}{||u||_{\mathcal{H}}} = \frac{||u - \sum_{k=1}^{N}\langle u,\psi_k\rangle_{\mathcal{H}}\psi_k||_{\mathcal{H}}}{||u||_{\mathcal{H}}} \tag{9.23}$$

Remark 9.2.12. *The relative projection error does not depend on the choice of the orthonormal basis used to represent the subspace $\mathrm{span}(\Psi_N)$. Therefore, the notations $\eta(u,\Psi_N)_{\mathcal{H}}$ and $\eta(u,\mathrm{span}(\Psi_N))_{\mathcal{H}}$ can be used interchangeably. The notation $\eta(u,\Psi_N)$ can be used instead of $\eta(u,\Psi_N)_{\mathcal{H}}$ when there is no ambiguity on the Hilbert space $\mathcal{H}$ considered.*

Remark 9.2.13. *The relative projection error can be preferred over the absolute error when the norm of the solution changes significantly over the manifold $\mathcal{M}$. Contrary to the absolute error, it does not depend on the magnitude of the solution, and it is symmetric when evaluated between two solutions: $\eta(u,\mathrm{span}(\{v\}))_{\mathcal{H}} = \eta(v,\mathrm{span}(\{u\}))_{\mathcal{H}}$, which enables interpreting it as a dissimilarity measure. Moreover, the reducibility of different problems can be more easily compared via their normalized Kolmogorov widths. It is also common practice to plot the normalized singular values of the POD to manipulate percentages.*

In the previous section, we have defined the sine dissimilarity $\tilde{\delta}_n$ involving the sines of the principal angles between elementary approximation spaces, for time-dependent problems where the solutions can be seen as trajectories in $\mathcal{H}$, and we have seen that it has the properties of a pseudometric. In this section, we focus on the sine dissimilarity $\tilde{\delta}_1$ for $n = 1$ computed between nonzero elements of $\mathcal{H} \subset L^2(\Omega)$ with the following formula:

$$\tilde{\delta}_1(u,v)_{\mathcal{H}} := \sin \measuredangle_{\mathcal{H}}(u,v) = \sqrt{1 - \frac{\langle u,v\rangle_{\mathcal{H}}^2}{||u||_{\mathcal{H}}^2||v||_{\mathcal{H}}^2}}, \tag{9.24}$$

for $(u,v) \in (\mathcal{H}\setminus\{0\})^2$. Note that we have added the subscript $\mathcal{H}$ in case the inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ differs from the $L^2(\Omega)$ inner product. It is recalled that, contrary to the distance $\|\cdot\|_{\mathcal{H}}$ commonly used for the construction of dictionaries of local ROMs, the sine dissimilarity measure focuses on the shape of the solutions and is not affected by their intensities. Indeed, for all $(u,v) \in (\mathcal{H}\setminus\{0\})^2$ and for all $(\lambda_1,\lambda_2) \in \mathbb{R}^{*2}$, $\tilde{\delta}_1(\lambda_1 u,\lambda_2 v)_{\mathcal{H}} = \tilde{\delta}_1(u,v)_{\mathcal{H}}$. For $(u,v) \in (\mathcal{H}\setminus\{0\})^2$, let us introduce the following binary relation:

$$u \sim_{\tilde{\delta}_1} v \iff \tilde{\delta}_1(u,v)_{\mathcal{H}} = 0.$$

This binary relation is reflexive and symmetric. In addition, it is also transitive, because $\tilde{\delta}_1(u,v)_{\mathcal{H}}$ is zero if and only if u and v are linearly dependent, according to the equality case of the Cauchy-Schwarz inequality. This binary relation is thus an equivalence relation, and enables to define the following equivalence classes for the elements of $\mathcal{H}\setminus\{0\}$: $[u] := \{v \in \mathcal{H}\setminus\{0\} \mid v \sim_{\tilde{\delta}_1} u\}$. The quotient set $\mathcal{H}/\sim_{\tilde{\delta}_1}$ is defined as the set of all these equivalence classes. In particular in $\mathcal{M}$, collinear solutions are represented by the same element of this quotient set, and $\mathcal{M}/\sim_{\tilde{\delta}_1}$ can be seen as the set of directions covered by the solution manifold $\mathcal{M}$. The sine dissimilarity $\tilde{\delta}_1$ is a metric on $\mathcal{H}/\sim_{\tilde{\delta}_1}$.

Property 9.2.14. *The relative projection error can be expressed using the sine dissimilarity: for all* $u \in \mathcal{H} \setminus \{0\}$ *and* $\mathcal{H}_N \in \mathrm{Gr}(N, \mathcal{H})$,

$$\eta\left(u, \mathcal{H}_N\right)_{\mathcal{H}}^2 = 1 - N + \sum_{j=1}^{N} \tilde{\delta}_1(u, h_j)_{\mathcal{H}}^2,$$

where the N *functions* h_j *form an orthonormal basis of the subspace* $\mathcal{H}_N$.

Proof. The orthogonal projection of $u \in \mathcal{H} \setminus \{0\}$ onto the subspace $\mathcal{H}_N$ reads: $\pi_{\mathcal{H}_N}(u) = \sum_{j=1}^{N} \langle u, h_j \rangle_{\mathcal{H}} h_j$. Therefore:

$$\begin{aligned} \eta\left(u, \mathcal{H}_N\right)_{\mathcal{H}}^2 &= ||u||_{\mathcal{H}}^{-2} \left(||u||_{\mathcal{H}}^2 - 2 \sum_{j=1}^{N} \langle u, h_j \rangle_{\mathcal{H}}^2 + \sum_{j=1}^{N} \sum_{i=1}^{N} \langle u, h_j \rangle_{\mathcal{H}} \langle u, h_i \rangle_{\mathcal{H}} \langle h_i, h_j \rangle_{\mathcal{H}} \right) \\ &= 1 - \sum_{j=1}^{N} \frac{\langle u, h_j \rangle_{\mathcal{H}}^2}{||u||_{\mathcal{H}}^2}. \end{aligned}$$

The proof is ended using the orthonormality of the basis $\{h_j\}_{1 \leq j \leq N}$ of $\mathcal{H}_N$ and Equation (9.24). □

Corollary 9.2.15. *For all* $(u, v) \in (\mathcal{H} \setminus \{0\})^2$,

$$\tilde{\delta}_1(u, v)_{\mathcal{H}} = \eta\left(u, \mathrm{span}(\{v\})\right)_{\mathcal{H}}.$$

9.3 Optimal partitions of the solution manifold

As an unsupervised learning task, clustering has no indisputable evaluation criterion. This is the reason why there is no hierarchy in the large variety of clustering algorithms. The algorithm must be selected according to the purpose. For model order reduction purposes, we have seen that the Kolmogorov N-width relates the physics problem's reducibility to projection errors on the approximation space, which makes the (relative) projection error η a good candidate for an evaluation criterion.

As shown in Equation (7.9) in Property 7.1.3, Kolmogorov widths can be decreased by limiting the angular deviation within the clusters. Having defined a dissimilarity measure δ_n (or $\tilde{\delta}_n$) based on angles in Definitions 9.2.5 and 9.2.10, one must look for compact-shaped clusters in terms of the dissimilarity δ_n (or $\tilde{\delta}_n$). Therefore, we use PAM k-medoids clustering algorithm to reduce the intra-cluster maximum angular deviations as much as possible, see Section 4.3.2 for more details about k-medoids clustering algorithms. Our physics-informed clustering method consists in running simplified simulations and applying PAM to simulation data using the ROM-oriented dissimilarity. This cluster analysis defines an automatic data labeling procedure, giving labeled training examples for the classifier of a ROM-net.

In the next paragraphs, we introduce the concept of optimal partitions of the solution manifold in terms of normalized Kolmogorov widths, and give a theoretical result showing the link between optimal partitions and k-medoids clustering.

9.3.1 Normalized Kolmogorov width variant

Let us recall our notations. We are considering a generic parametrized partial differential equation (pPDE):

$$\mathcal{D}(u; x) = 0, \tag{9.25}$$

where $u \in \mathcal{H} \subset L^2(\Omega)$ and $x \in \mathcal{X}$ denote respectively the solution and the parameter, with $\mathcal{H}$ being a Hilbert space and $\mathcal{X}$ the parameter domain. The pPDE (9.25) is assumed to be well-posed in the sense of Hadamard: $\mathcal{X} \ni x \mapsto u \in \mathcal{H}$ solution of (9.25) is a continuous application, called the solution application. For clarity of presentation, this application is still denoted by u: for all $x \in \mathcal{X}$, $u(x)$ is the unique solution of (9.25). The solution manifold $\mathcal{M}$ is defined as the image of the solution application: $\mathcal{M} = u(\mathcal{X})$. We suppose that $\mathcal{M}$ does not contain the zero solution: $\exists\, \lambda > 0$ such that $\inf_{x \in \mathcal{X}} \|u(x)\|_{\mathcal{H}} = \inf_{u \in \mathcal{M}} \|u\|_{\mathcal{H}} > \lambda$.

The uncertain parameter x is modeled by a random variable X taking values in $\mathcal{X}$ and following a probability distribution denoted by $p_X : \mathcal{X} \to \mathbb{R}_+$. Another random variable can be defined using the solution application: $U := u(X)$, whose probability distribution p_U is obtained when X follows p_X. In particular, the solution manifold is the support of the probability density function p_U: $\mathcal{M} = \mathrm{supp}(p_U)$.

In section 7.1, we defined the Kolmogorov N-width d_N, its variant d'_N introduced in [195] involving the mean squared error, and a normalized Kolmogorov N-width $\tilde{d}_N$. We now introduce a variant of the normalized Kolmogorov width denoted by $\check{d}_N$, considering the mean squared error instead of the maximum error as in the definition of d'_N:

$$\begin{aligned} \check{d}_N(p_U)_{\mathcal{H}} &:= \left(\inf_{\mathcal{H}_N \in \mathrm{Gr}(N,\mathcal{H})} \mathbb{E}_{U \sim p_U} \left[\eta\left(U, \mathcal{H}_N\right)_{\mathcal{H}}^2 \right] \right)^{1/2} \\ &= \left(\inf_{\mathcal{H}_N \in \mathrm{Gr}(N,\mathcal{H})} \mathbb{E}_{X \sim p_X} \left[\eta\left(u(X), \mathcal{H}_N\right)_{\mathcal{H}}^2 \right] \right)^{1/2}. \end{aligned} \tag{9.26}$$

This variant is referred to as normalized Kolmogorov width too in the sequel.

9.3.2 Optimal K-N-ROM-dictionary partitions

Let $\mathcal{P} \subset \mathcal{M} \subset \mathcal{H} \setminus \{0\}$ be a subset of the solution manifold. The probability $\mathbb{P}(\mathcal{P})$ of the event $U \in \mathcal{P}$ reads:

$$\mathbb{P}(\mathcal{P}) = \int_{\mathcal{P}} p_U(u) du = \mathbb{E}_{U \sim p_U}[\mathbb{1}_{\mathcal{P}}(U)],$$

where $\mathbb{1}$ is the indicator function. A partition $\{\mathcal{M}_k\}_{1 \le k \le K}$ of $\mathcal{M}$ is a collection of non-empty subsets of $\mathcal{M}$ such that any point u of the solution manifold $\mathcal{M}$ belongs to exactly one of these subsets. Let $K \ge 2$, and consider a partition of $\mathcal{M}$ into K subsets. The following definition introduces *optimal K-N-ROM-dictionary partitions* as partitions that are optimal for reduced-order modeling.

Definition 9.3.1 (Optimal K-N-ROM-dictionary partitions)**.** *The family of sets $\{\mathcal{M}_k\}_{1 \le k \le K}$ is an optimal K-N-ROM-dictionary partition of $\mathcal{M}$ if it is a partition of $\mathcal{M}$ and*

$$\{\mathcal{M}_k\}_{1 \le k \le K} := \arg \inf_{\substack{\{\mathcal{P}_k\}_{1 \le k \le K} \\ \text{partition of } \mathcal{M}}} \sum_{k=1}^{K} \mathbb{P}(\mathcal{P}_k)\ \check{d}_N^2(p_{U|u \in \mathcal{P}_k})_{\mathcal{H}}. \tag{9.27}$$

An optimal K-N-ROM-dictionary partition of a solution manifold is a partition of size K, leading to a dictionary of local ROMs with N modes minimizing the expectation of the squared intra-cluster normalized Kolmogorov N-width. Using Equation (9.26), Equation (9.27) reads:

$$\{\mathcal{M}_k\}_{1\leq k\leq K} = \arg \inf_{\substack{\{\mathcal{P}_k\}_{1\leq k\leq K} \\ \text{partition of } \mathcal{M}}} \sum_{k=1}^{K} \mathbb{P}(\mathcal{P}_k) \inf_{\mathcal{H}_N^k \in \mathrm{Gr}(N,\mathcal{H})} \mathbb{E}_{U\sim p_{U|u\in\mathcal{P}_k}} \left[\eta\left(U, \mathcal{H}_N^k\right)_{\mathcal{H}}^2 \right]. \quad (9.28)$$

9.3.3 Optimal K-1-ROM-dictionary partitions

Taking $N = 1$, Equation (9.28) defines optimal K-1-ROM-dictionary partitions as the solutions of the optimization problem:

$$\{\mathcal{M}_k\}_{1\leq k\leq K} = \arg \inf_{\substack{\{\mathcal{P}_k\}_{1\leq k\leq K} \\ \text{partition of } \mathcal{M}}} \sum_{k=1}^{K} \mathbb{P}(\mathcal{P}_k) \inf_{\tilde{u}_k \in \mathcal{H}/\sim_{\tilde{\delta}_1}} \mathbb{E}_{U\sim p_{U|u\in\mathcal{P}_k}} \left[\tilde{\delta}_1(U, \tilde{u}_k)_{\mathcal{H}}^2 \right], \quad (9.29)$$

where the relative projection error is replaced by the sine dissimilarity thanks to Corollary 9.2.15. Note that the second infimum is taken on the quotient set $\mathcal{H}/\sim_{\tilde{\delta}_1}$, because looking for an optimal element of the quotient set is equivalent to searching for an optimal 1D approximation space in $\mathrm{Gr}(1, \mathcal{H})$. Equation (9.29) can be interpreted as the continuous version of a representative-based clustering problem, where for a given integer $K \geq 2$, the objective is to find K representative elements whose nearest neighbors for a given dissimilarity measure define the K clusters. Given a metric measure space $\mathcal{V}$ whose metric (*resp.* measure) is denoted by $\delta_{\mathcal{V}}$ (*resp.* $\mu_{\mathcal{V}}$), and given a subset $\mathcal{V}'$ of $\mathcal{V}$ with a nonzero measure, the continuous representative-based clustering problem can be stated as follows:

$$\inf_{\substack{\{\mathcal{P}_k\}_{1\leq k\leq K} \\ \text{partition of } \mathcal{V}'}} \sum_{k=1}^{K} \inf_{\tilde{u}_k \in \mathcal{V}} \int_{\mathcal{P}_k} \delta_{\mathcal{V}}^2(v, \tilde{u}_k) d\mu_{\mathcal{V}}(v),$$

which is equivalent to:

$$\inf_{\substack{\{\mathcal{P}_k\}_{1\leq k\leq K} \\ \text{partition of } \mathcal{V}'}} \sum_{k=1}^{K} \frac{\mu_{\mathcal{V}}(\mathcal{P}_k)}{\mu_{\mathcal{V}}(\mathcal{V}')} \inf_{\tilde{u}_k \in \mathcal{V}} \frac{1}{\mu_{\mathcal{V}}(\mathcal{P}_k)} \int_{\mathcal{P}_k} \delta_{\mathcal{V}}^2(v, \tilde{u}_k) d\mu_{\mathcal{V}}(v).$$

The ratio $\mu_{\mathcal{V}}(\mathcal{P}_k)/\mu_{\mathcal{V}}(\mathcal{V}')$ can be seen as the probability $\mathbb{P}(\mathcal{P}_k)$ of being in $\mathcal{P}_k$ when drawing a realization of the uniform distribution on $\mathcal{V}'$. The integral term normalized by the measure of the cluster corresponds to the expectation in Equation (9.29). In both cases, the integrand is a squared metric. When $\mathcal{V}$ is a Hilbert space and $\delta_{\mathcal{V}}$ is the norm induced by its inner product, the aforementioned clustering problem is a continuous k-means clustering problem, where the objective is to find clusters minimizing the sum of the intra-cluster inertia. In this case, the optimal representative elements are the centroids (or means) of the clusters. In a nutshell, the optimal K-1-ROM-dictionary partitions are the solutions of a representative-based clustering problem on the quotient set $\mathcal{H}/\sim_{\tilde{\delta}_1}$ endowed with the metric $\tilde{\delta}_1$.

9.3.4 Algorithm for the construction of a dictionary of local ROMs

The optimization problem defining optimal K-N-ROM-dictionary partitions in Definition 9.3.1 is numerically intractable, since there is an infinite number of possible partitions of the continuous solution manifold and since each candidate partition requires solving K optimization problems for the construction of the local ROMs. In the same fashion the (snapshot) POD has been proposed as a practical procedure for approximating the optimal N-ROM subspace, we propose an algorithm approximating the optimal K-N-ROM-dictionary partitions, given a sampling of the solution manifold as a set of m precomputed solutions $\widehat{\mathcal{M}} := \{u_i\}_{1\leq i\leq m}$. This *a priori* sample is usually done by applying a design of experiments over the parameter domain $\mathcal{X}$.

Approximate optimal K-N-ROM-dictionary partitions of discrete solution sets

We look for an approximation of the optimal K-N-ROM-dictionary partitions $\{\widehat{\mathcal{M}}_k\}_{1\leq k\leq K}$ of the discretized set $\widehat{\mathcal{M}}$. The probabilities $\mathbb{P}(\widehat{\mathcal{M}}_k)$ are obtained by taking the ratios $|\widehat{\mathcal{M}}_k|/m$, where $|\widehat{\mathcal{M}}_k|$ is the cardinality of $\widehat{\mathcal{M}}_k$. The true but unknown probability density functions $p_{U|u\in\mathcal{M}_k}$ are replaced by the empirical distributions:

$$\widehat{p}_{U|u\in\widehat{\mathcal{M}}_k}(u) = \frac{1}{|\widehat{\mathcal{M}}_k|}\sum_{i=1}^{m} \mathbb{1}_{\widehat{\mathcal{M}}_k}(u_i)\ \delta(u-u_i),$$

where δ is the Dirac delta function. Using Equation (9.26), the squared normalized Kolmogorov N-width of this probability mass function is then:

$$\check{d}_N(\widehat{p}_{U|u\in\widehat{\mathcal{M}}_k})^2_{\mathcal{H}} = \inf_{\mathcal{H}_N^k\in\mathrm{Gr}(N,\mathcal{H})} \frac{1}{|\widehat{\mathcal{M}}_k|}\sum_{i=1}^{m} \mathbb{1}_{\widehat{\mathcal{M}}_k}(u_i)\ \eta\left(u_i,\mathcal{H}_N\right)^2_{\mathcal{H}}.$$

Like the RBM and the snapshot POD methods, and to derive a computable algorithm, a basis for the approximation of the best subspace $\mathcal{H}_N^k$ is searched in the set $\mathcal{A}_N(\widehat{\mathcal{M}}_k)$, $N \leq \dim(\mathrm{span}(\widehat{\mathcal{M}}_k))$, defined as the set containing all the $\mathcal{H}$-orthonormal families of N elements of $\mathrm{span}(\widehat{\mathcal{M}}_k)$. From Definition 9.3.1 and Property 9.2.14, an approximation of the optimal K-N-ROM-dictionary partitions $\{\widehat{\mathcal{M}}_k\}_{1\leq k\leq K}$ of a discrete solution set $\widehat{\mathcal{M}}$ is sought as:

$$\arg \min_{\substack{\{\widehat{\mathcal{P}}_k\}_{1\leq k\leq K}\\ \text{partition of } \widehat{\mathcal{M}}}} \sum_{k=1}^{K} \min_{\{h_j^k\}_{1\leq j\leq N}\in\mathcal{A}_N(\widehat{\mathcal{M}})} \sum_{i=1}^{m} \mathbb{1}_{\widehat{\mathcal{P}}_k}(u_i)\ \sum_{j=1}^{N} \tilde{\delta}_1(u_i,h_j^k)^2_{\mathcal{H}}. \tag{9.30}$$

Approximate optimal K-1-ROM-dictionary partitions of discrete solution sets

Property 9.3.2. *When considering a discrete solution set $\widehat{\mathcal{M}}$ and if the 1-ROM subspaces are sought in* $\mathrm{span}(\widehat{\mathcal{M}})$*, the optimal K-1-ROM-dictionary partitions are exactly the minimizers of the cost function of k-medoids clustering with the sine dissimilarity measure $\tilde{\delta}_1$.*

Proof. From Equation (9.30) (or discretizing the solution manifold in Equation (9.29)), optimal K-1-ROM-dictionary partitions satisfy

$$\arg \min_{\substack{\{\widehat{\mathcal{P}}_k\}_{1\leq k\leq K} \\ \text{partition of } \widehat{\mathcal{M}}}} \sum_{k=1}^{K} \min_{\tilde{u}_k \in \widehat{\mathcal{M}}} \sum_{i=1}^{m} \mathbb{1}_{\widehat{\mathcal{P}}_k}(u_i)\ \tilde{\delta}_1(u_i, \tilde{u}_k)_{\mathcal{H}}^2,$$

from which we recognize the cost function of k-medoids clustering. □

Hence, the case $N = 1$ leads to an optimization problem for which various computable heuristic approaches have been proposed, including the Partitioning Around Medoids (PAM [84]).

Algorithm for approximate optimal partitions

Property 9.3.2 cannot be directly extended to $N > 1$, and optimizing the partition and the approximation spaces simultaneously in Equation (9.30) requires computing many candidate local subspaces, which is very expensive. As a practical algorithm, we propose to: (i) compute the optimal K-1-ROM-dictionary partitions over the sampled solution manifold, and (ii) compute the local N-dimensional approximation spaces using any classical reduced-order modeling method on each element of this partition, for instance the snapshot POD or the RBM.

Remark 9.3.3. *In addition to the arguments given in Remark 9.2.13 for the use of normalized Kolmogorov widths, using relative errors enables linking the concept of optimal partitions with a representative-based cluster analysis. Trying to do the same for absolute Kolmogorov widths would have required considering a dissimilarity obtained by symmetrizing the absolute errors: the link between the corresponding optimal K-1-ROM-dictionary partitions and such dissimilary would be lost, and Property 9.3.2 would not hold anymore.*

9.4 Snapshots selection

Once clusters have been identified within the dataset, one must select relevant points for which the entire high-fidelity simulation will be run to provide high-fidelity snapshots for the construction of the local ROMs. For each cluster, the high-fidelity snapshots must be well distributed and representative of the cluster's members. When one wants to use only one snapshot per cluster, then the clusters' medoids are good candidates. For more than one snapshot per cluster, a second k-medoids cluster analysis can be conducted within each cluster, with n_s subclusters where n_s is the desired number of high-fidelity snapshots per cluster, using the same dissimilarity measure as for the first clustering. High-fidelity snapshots can then be computed for the subclusters' medoids. This method corresponds to a two-stage hierarchical k-medoids clustering. Another way to select snapshots would consist in following a *maximin* greedy approach: we first select the cluster's medoid, then the simulation that is the farthest away from the medoid within the cluster, and then the simulation maximizing the minimum distance between the two aforementioned simulations. In other words, starting from the medoid, snapshots are iteratively selected by finding the simulation maximizing the minimum distance to the snapshots that have already been selected.

9.5 Application: 1D steady heat equation

9.5.1 Problem description

Let us consider the following ordinary differential equation:

$$\begin{cases} -\left(\lambda u'\right)'(\xi) &= s(\xi) \qquad \forall \xi \in [0;L] \\ u(0) &= u_0 \\ u(L) &= u_0 \end{cases} \tag{9.31}$$

where $\lambda \in L^2([0;L])$, $s \in L^2([0;L])$, $u_0 \in \mathbb{R}$ and $u - u_0 \in H_0^1([0;L])$. This equation describes the thermal behavior of an heterogeneous continuous medium of length L with thermal conductivity $\lambda(\xi)$ and temperature $u(\xi)$, in the presence of a heat source $s(\xi)$. We are interested in the behavior of the solution u under variying source terms and conductivity functions. The conductivity function λ is defined by:

$$\lambda(\xi) = \lambda_1 \mathbb{1}_{\{\epsilon(\zeta)L \leq \xi \leq (\epsilon(\zeta)+\zeta)L\}} + \lambda_2(\mathbb{1}_{\{\xi < \epsilon(\zeta)L\}} + \mathbb{1}_{\{\xi > (\epsilon(\zeta)+\zeta)L\}}) \tag{9.32}$$

with $\lambda_2 = 1000\lambda_1 \in \mathbb{R}_+^*$ and with ζ being a random variable following the uniform distribution $\mathcal{U}(0.1, 0.5)$. The random variable $\epsilon(\zeta)$ follows the uniform distribution $\mathcal{U}(0, 1-\zeta)$. The source term s is modeled by a zero-mean Gaussian process with an exponential covariance function. The problem described by Equation (9.31) is therefore parametrized by the heat source distribution s and the microstructural parameters ζ and $\epsilon(\zeta)$. The weak formulation of Equation (9.31) reads:

$$\int_0^L \lambda(\xi) v'(\xi) u'(\xi) d\xi = \int_0^L s(\xi) v(\xi) d\xi \quad \forall v \in H_0^1([0;L]) \tag{9.33}$$

The interval $[0;L]$ is discretized into $\mathcal{N} - 1 = 1999$ subdivisions of length $h = L/(\mathcal{N}-1)$. The vertices $\{\xi_i = ih\}_{0 \leq i \leq \mathcal{N}-1}$ define a finite-element mesh whose P1 shape functions are denoted by $\{\phi_i\}_{1 \leq i \leq \mathcal{N}-2}$. The shape functions ϕ_0 and $\phi_{\mathcal{N}-1}$ are not used because of the Dirichlet boundary conditions. The finite-element method computes a high-fidelity approximate solution $u - u_0$ in the space span $(\{\phi_i\}_{1 \leq i \leq \mathcal{N}-2})$, whose coordinates are stored in a vector $\mathbf{q} \in \mathbb{R}^{\mathcal{N}-2}$. This vector is the solution of the following linear system:

$$\mathbf{K}\mathbf{q} = \mathbf{f} \tag{9.34}$$

with $\mathbf{K} \in \mathbb{R}^{(\mathcal{N}-2)\times(\mathcal{N}-2)}$ given by:

$$K_{ij} = \int_0^L \lambda(\xi) \phi_i'(\xi) \phi_j'(\xi) d\xi \quad \forall (i,j) \in [\![1; \mathcal{N}-2]\!] \tag{9.35}$$

and $\mathbf{f} \in \mathbb{R}^{\mathcal{N}-2}$ given by:

$$f_i = \int_0^L s(\xi) \phi_i(\xi) d\xi \quad \forall (i,j) \in [\![1; \mathcal{N}-2]\!] \tag{9.36}$$

A dataset of 1000 realizations of the random source term and microstructural parameters is generated. For each example in the dataset, the finite-element solution $\mathbf{q}$ is computed with a Python routine. Figure 9.2 shows the solution's behavior for different configurations. One can observe that the solution is not affected by the source term in

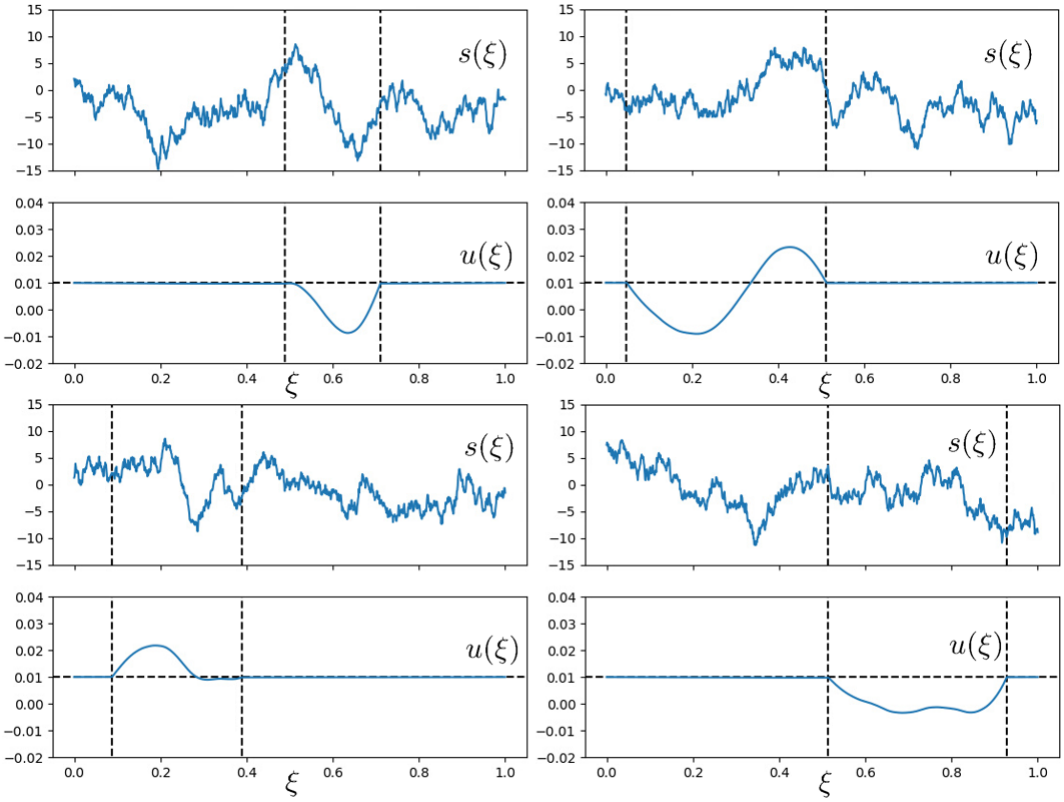

Figure 9.2: Examples of solutions $u(\xi)$ for different source terms $s(\xi)$. The vertical dashed lines indicate the locations of the interfaces between the constituents of the bimaterial. Between the two interfaces, the thermal conductivity is λ_1. Outside of this interval, the thermal conductivity is $\lambda_2 = 1000\lambda_1$.

high-conductivity regions. Figure 9.3 gives the singular values of the matrix containing the 1000 solutions. It can be observed that the decay of the singular values is rather slow for a 1D problem, meaning that this problem is non-reducible and that a dictionary of local ROBs may be required. The database is splitted into two subsets: a training set and a test set, both containing 500 examples. The training set is used to identify clusters and build the ROBs, while the test set is used for evaluation purposes.

Remark 9.5.1. *In the training phase of a dictionary-based ROM-net for time-dependent physics problems, the simplified problem that is simulated to provide data for the clustering procedure generally corresponds to a few time steps of the target problem. In this example, Equation* (9.31) *does not define a time-dependent problem. In this case, the simplified problem can be defined as the target problem solved on a coarse finite-element mesh.*

9.5.2 Comparison of different model order reduction strategies

Let $x = (s, \zeta, \epsilon(\zeta))$ denote the parameter of the problem. After projection of the source term in the finite-element basis, the parameter x is represented by a $\mathcal{N} + 2$-dimensional vector $\mathbf{x}$ whose coordinates are centered and scaled to unit variance. This way, distances in the parameter space can be computed with the Euclidean distance:

$$\delta_{\mathcal{X}}(x, x') = ||\mathbf{x} - \mathbf{x}'||_2 \tag{9.37}$$

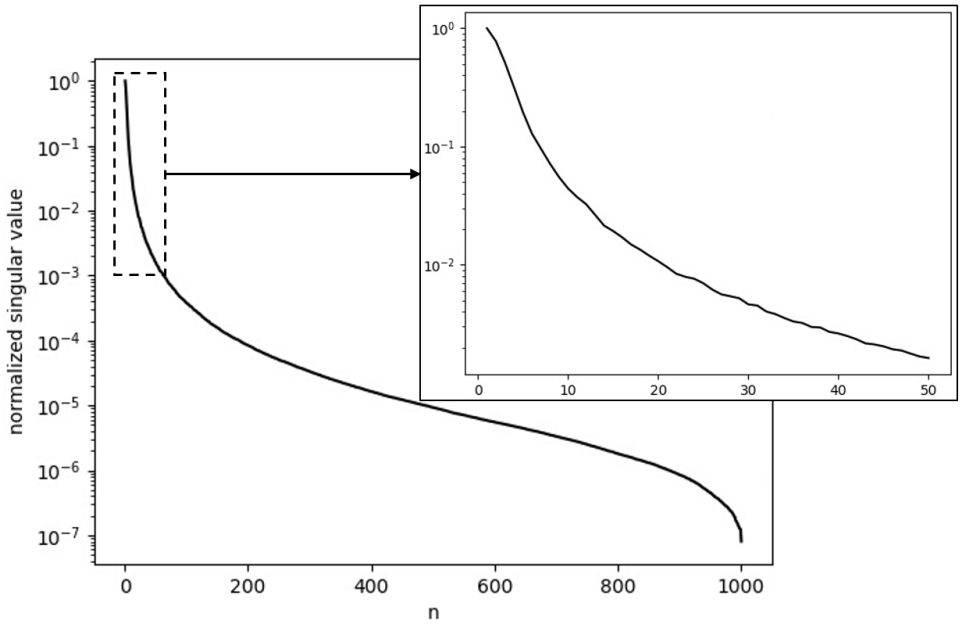

Figure 9.3: Decay of the singular values obtained by singular value decomposition on the matrix containing all the examples in the dataset.

Introducing the notation $\mathbf{q}(x) \in \mathbb{R}^{\mathcal{N}-2}$ for the solution of Equation (9.34) for a given parameter x, one can define a physics-informed dissimilarity measure $\delta_{\mathcal{H}}$ using the Euclidean distance in the solution space:

$$\delta_{\mathcal{H}}(x, x') = ||\mathbf{q}(x) - \mathbf{q}(x')||_2 \tag{9.38}$$

These dissimilarity measures are compared with the ROM-oriented dissimilarity measure $\tilde{\delta}_1$ introduced in Equation (9.24), obtained by computing the sine dissimilarity in the solution space. K-medoids clustering is used for both snapshots selection and manifold partitioning in conjunction with one of these three dissimilarity measures. Different model order reduction strategies are compared in terms of projection errors under the following setting:

- **Equivalent number of snapshots:** all the strategies use the same total number of snapshots, which ensures equal budgets for high-fidelity simulations in the training phase. It is recalled that the high-fidelity snapshots are given by high-fidelity simulations that are more expensive than the simplified simulations used to generate the database and find clusters.
- **Equivalent number of POD modes:** all the ROBs use the same number of modes, which ensures equivalent speed-ups when exploiting the ROMs.

If a dictionary of K local ROBs is compared with a global ROB made of N modes, then each local ROB must have N modes. For the construction of these local cluster-specific ROBs, $n_s = N$ snapshots are selected in each cluster using the two-stage hierarchical k-medoids clustering procedure. Hence, the total number of snapshots is $Kn_s = KN$. Snapshots for the construction of the global ROB are therefore selected by taking the medoids of a single k-medoids clustering with KN clusters.

Six model order reduction strategies are considered, namely:

- Three global ROBs containing N modes computed from KN snapshots. The snapshots are selected thanks to a k-medoids cluster analysis with KN clusters, using different dissimilarities:
  - **Global ROM 1** uses the dissimilarity $\delta_{\mathcal{X}}$ (Euclidean distance in the parameter space).
  - **Global ROM 2** uses the dissimilarity $\delta_{\mathcal{H}}$ (Euclidean distance in the solution space).
  - **Global ROM 3** uses the ROM-oriented dissimilarity $\tilde{\delta}_1$ (sine dissimilarity in the solution space).
- Three ROM dictionaries consisting of K local ROBs with N modes each. Each local ROB is inferred from N snapshots. Again, k-medoids is applied with different dissimilarity measures:
  - **ROM dictionary 1** uses the dissimilarity $\delta_{\mathcal{X}}$ (Euclidean distance in the parameter space). This strategy is the most natural and simple one among ROM dictionaries.
  - **ROM dictionary 2** uses the dissimilarity $\delta_{\mathcal{H}}$ (Euclidean distance in the solution space) like in [40, 41, 224, 43, 225]. This strategy belongs to physics-informed strategies.
  - **ROM dictionary 3** uses the ROM-oriented dissimilarity $\tilde{\delta}_1$ (sine dissimilarity in the solution space). This is the strategy we have introduced in this thesis for dictionary-based ROM-nets. Like ROM dictionary 2, it relies on a physics-informed cluster analysis, but with another dissimilarity.

In this section, the comparison is presented for $K = 6$ and $N = n_s = 3$, for reasons that will become clearer in the next chapter focusing on hyperparameter tuning. For clustering, we use our own implementation of PAM [84, 95] k-medoids algorithm, with multiple random initializations for the medoids. Projection errors as defined in Equation (9.23) are computed for the 500 test examples for each strategy, which enables estimating their probability density functions using Gaussian kernel density estimation (see section 6.6.1. of [55]). The violin plots of the projection errors are given in Figure 9.4, and the values of the quartiles and expectations are given in Table 9.1. The third ROM dictionary using the ROM-oriented dissimilarity clearly outperforms the other strategies. Although using a physics-informed clustering procedure, ROM dictionary 2 fails to improve the performances of global ROMs on this specific example. This result illustrates the fact that the Euclidean distance is not always appropriate for model order reduction purposes. ROM dictionary 1 gives the worst results, showing that integrating physics in cluster analyses is crucial when the final objective is to build local approximation spaces. Interestingly, these results also show that using local ROBs can deteriorate the performances of a global ROB when choosing an improper dissimilarity measure for clustering. In this example, the three global ROMs give approximately the same projection errors. These errors are lower than those obtained with ROM dictionary 1 and ROM dictionary 2 because the global ROMs have more relevant snapshots, since they use KN well-distributed snapshots instead of N badly-distributed snapshots. Hence, the dissimilarities $\delta_{\mathcal{X}}$ and $\delta_{\mathcal{H}}$ both define inefficient notions of locality in this example.

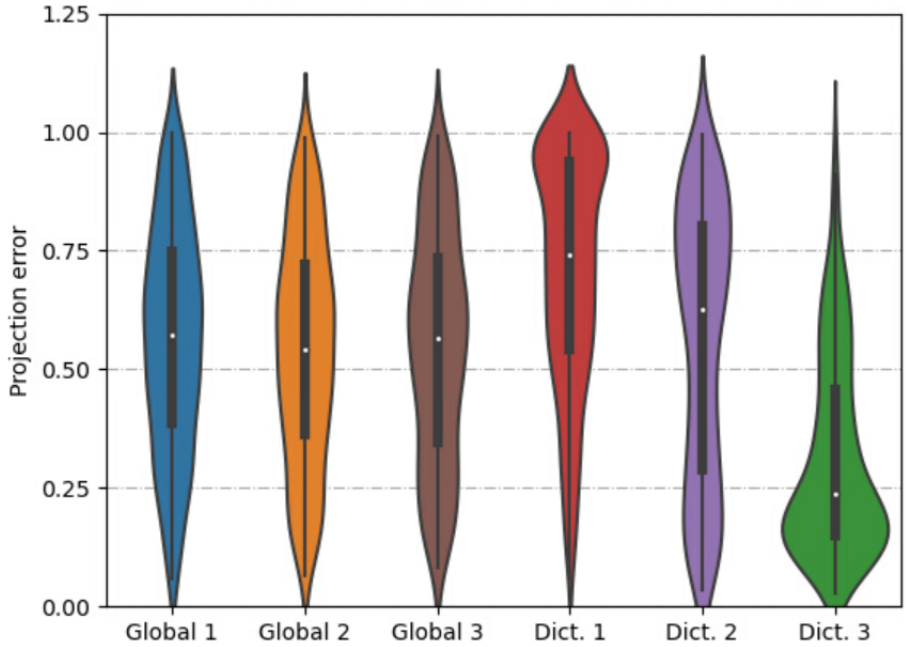

Figure 9.4: Violin plots of the projection errors for different model order reduction strategies, with $(K, N, n_s) = (6, 3, 3)$.

Table 9.1: Quartiles and means of the projection errors for different model order reduction strategies, with $(K, N, n_s) = (6, 3, 3)$.

Strategy	Dissimilarity	Q_1	Median	Q_3	Mean
Global ROM 1	$\delta_{\mathcal{X}}$	0.3863	0.5735	0.7477	0.5636
Global ROM 2	$\delta_{\mathcal{H}}$	0.3611	0.5427	0.7208	0.5397
Global ROM 3	$\tilde{\delta}_1$	0.3460	0.5666	0.7346	0.5480
ROM dictionary 1	$\delta_{\mathcal{X}}$	0.5434	0.7412	0.9379	0.7071
ROM dictionary 2	$\delta_{\mathcal{H}}$	0.2874	0.6280	0.8038	0.5586
ROM dictionary 3	$\tilde{\delta}_1$	0.1482	0.2369	0.4584	0.3132

Remark 9.5.2. *Figure 9.4 gives projection errors obtained when choosing the correct cluster and thus the most suitable local ROB. The ROM-net's classification errors would have the effect of moving the distribution of ROM dictionary 3 towards larger errors, reducing the gap between the errors made by the different model order reduction strategies. Therefore, particular attention must be paid to the training of the ROM-net's classifier.*

Figure 9.5 plots the projection error against the dissimilarity measure $\delta_{\mathcal{X}}$ (left), $\delta_{\mathcal{H}}$ (middle) and $\tilde{\delta}_1$ (right) separating a test example from its closest snapshot. One can clearly see the correlation between the projection error and our ROM-oriented dissimilarity $\tilde{\delta}_1$, contrasting with the absence of correlations between the projection error and the other dissimilarities. This example of a non-reducible problem validates the ROM-oriented dissimilarity introduced in this thesis.

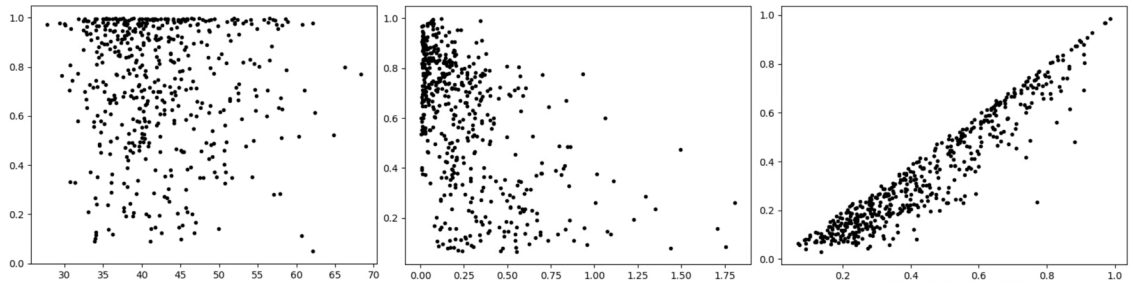

Figure 9.5: Scatter plots giving the projection error (y-axis) for test data against the dissimilarity with the closest snapshot (x-axis). From the left to the right: Euclidean distances in the parameter space, Euclidean distances in the solution space, and ROM-oriented dissimilarity.

9.6 Summary

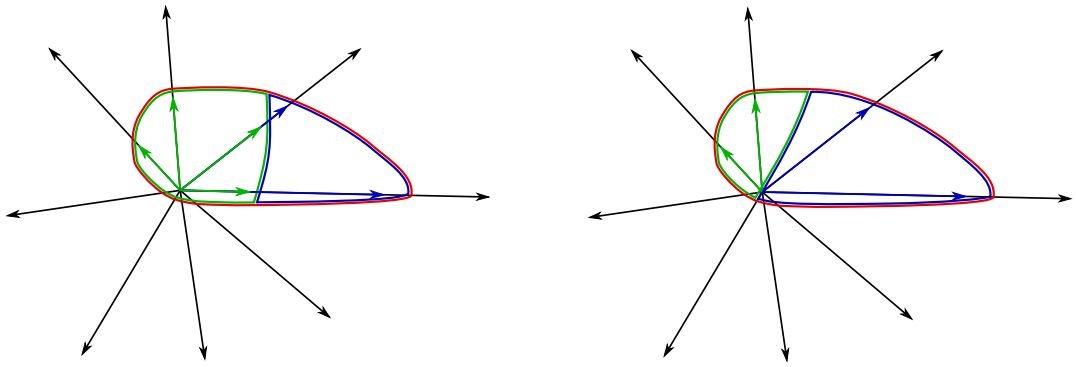

Figure 9.6: Schematic illustration showing the differences between clustering with the Euclidean distance (left) and with the ROM-oriented dissimilarity measure (right). The area inside the closed red curve corresponds to the solution manifold, and the arrows represent independent directions in the ambient space. The green and the blue curves delineate the two clusters.

In this chapter, we have introduced a physics-informed clustering procedure for the construction of a dictionary of local ROMs. It has been shown that the Euclidean distance may fail to identify relevant clusters for model order reduction purposes. To address this issue, a ROM-oriented dissimilarity measure based on angles is proposed. Contrary to the Euclidean distance, it does not depend on the magnitude of the physical fields. As illustrated on Figure 9.6, clustering with the Euclidean distance may give clusters sharing some directions of the ambient space, while an angle-based dissimilarity can efficiently identify clusters of data embedded in low-dimensional subspaces.

We stress that we only question the choice of the Euclidean distance, but not the methods using it for the construction of local ROMs. Indeed, our study does not include important ingredients used in the methods presented in [40, 41, 224, 225], such as the possibility to switch between clusters, the low-rank update of the local ROBs when switching between clusters, and the use of affine subspaces rather than vector subspaces which may result in clusters sharing fewer common directions.

However, our study on optimal partitions of the solution manifold in terms of a normalized Kolmogorov width gives theoretical motivations for the use of a representative-based clustering algorithm (such as k-medoids clustering) with our ROM-oriented dissimilarity measure.

* *
*

Chapter 10

Hyperparameters tuning

Abstract: *This chapter introduces an a priori efficiency criterion assessing the relevance of a ROM-net for a given problem. It relies on the concept of gain that evaluates the profitability of a ROM-net with respect to a single global ROM. This criterion can be evaluated before time-consuming steps in the training phase, and gives a very practical method for hyperparameters calibration under constrained computational costs.*

Note: The work presented in this chapter follows up the internship of Ali Ketata, that I had the pleasure of supervising at Safran in 2020.

Remark 10.0.1. *This chapter is taken from our paper [49], with some modifications.*

Contents

10.1 Gain with respect to a global reduced-order model

A dictionary-based ROM-net [47] is made of a dictionary of K local ROMs and a classifier $\mathcal{C}_K$ which automatically selects the best model from the dictionary for a given point in the parameter space without computing any physics-informed dissimilarity, see Figure 8.1. The real classifier $\mathcal{C}_K$ enables bypassing the simplified simulation that is required to evaluate the perfect classifier $\mathcal{K}_K$, see Figure 8.2. In this section, it is assumed that all the dictionary's ROMs have the same number of modes, denoted by N, and have been built from the same number of high-fidelity snapshots, denoted by n_s. A dictionary of K ROBs with N modes and n_s high-fidelity snapshots per basis is denoted by $\{\Psi_k^{(K,N,n_s)}\}_{k\in[\![1;K]\!]}$. The objective of this chapter is to define a practical method for the calibration of the hyperparameters K, N and n_s, based on an evaluation criterion quantifying the ROM-net's profitability with respect to a single global ROM. This criterion must be computable very early in the ROM-net's training phase, right after the physics-informed clustering procedure in Figure 8.2 and before the computation of high-fidelity snapshots, the construction of the ROMs, and the classifier's training phase. Therefore, the local ROBs $\{\Psi_k^{(K,N,n_s)}\}_{k\in[\![1;K]\!]}$ used in the evaluation criterion are simply built from Kn_s simplified snapshots selected for example by the two-stage hierarchical k-medoids clustering, instead of the corresponding high-fidelity snapshots that will be computed afterwards. Their performances are compared with the performance of a global ROB $\Psi_g^{(1,N,Kn_s)}$ containing N modes and inferred from the same Kn_s snapshots as $\{\Psi_k^{(K,N,n_s)}\}_{k\in[\![1;K]\!]}$. This global basis thus benefits from the physics-informed clustering procedure for the selection of its snapshots. The ROBs are related to the function $u(X) \in L^2(\Omega \times [0;t_f])$ parametrized by the random variable X representing the current point in the parameter space. The following definition introduces the *gain* used in our evaluation criterion:

Definition 10.1.1 (Gain)**.** *Given integers $K > 1$, $N > 0$ and $n_s > 0$ and a classifier $\mathcal{F}_K : \mathcal{X} \to [\![1;K]\!]$, the gain is defined by:*

$$G(X;K,N,n_s,\mathcal{F}_K) = \frac{\eta\left(u(X), \Psi_g^{(1,N,Kn_s)}\right)}{\eta\left(u(X), \Psi_{\mathcal{F}_K(X)}^{(K,N,n_s)}\right)} \tag{10.1}$$

where $u(X)$ results from a simplified simulation. For $K = 1$, the gain equals to 1.

Remark 10.1.2. *In Definition 10.1.1, the primal variable u can be replaced by a quantity of interest, depending on the choice made for the definition of the ROM-oriented dissimilarity.*

As a function of X, the gain can be seen as a random variable parametrized by the hyperparameters K, N and n_s and the classifier. The notations $G_{\mathcal{C}}(K,N,n_s)$ and $G_{\mathcal{K}}(K,N,n_s)$ denote $G(X;K,N,n_s,\mathcal{C}_K)$ and $G(X;K,N,n_s,\mathcal{K}_K)$ respectively. Right after the physics-informed clustering procedure, the user cannot evaluate the gain $G_{\mathcal{C}}(K,N,n_s)$ since the real classifier $\mathcal{C}_K$ has not been trained yet. However, the clusters implicitly define the perfect classifier $\mathcal{K}_K$ and thus the user has access to values of the gain $G_{\mathcal{K}}(K,N,n_s)$. In the next property, the following assumption is made:

[A1] The gain $G_{\mathcal{K}}(K,N,n_s)$ is assumed to be deterministic, which means that it is no longer a random variable but rather a deterministic function of the hyperparameters K, N and n_s. In other words, when the right cluster is chosen, the gain does not depend on X.

Property 10.1.3 (Gain decomposition)**.** *Under assumption [A1]:*

$$\mathbb{E}[G_{\mathcal{C}}(K,N,n_s)] = p\ G_{\mathcal{K}}(K,N,n_s) + (1-p)\ E(K,N,n_s) \tag{10.2}$$

where $p = \mathbb{P}(\mathcal{C}_K = \mathcal{K}_K)$ *is the classification accuracy and* $E(K,N,n_s)$ *given by:*

$$E(K,N,n_s) := \mathbb{E}[G_{\mathcal{C}}(K,N,n_s) \,|\, \mathcal{C}_K \neq \mathcal{K}_K] \tag{10.3}$$

is the conditional expectation of the gain $G_{\mathcal{C}}(K,N,n_s)$ *when selecting the wrong ROB.*

Proof. The expected gain $\mathbb{E}[G_{\mathcal{C}}(K,N,n_s)]$ satisfies:

$$\mathbb{E}[G_{\mathcal{C}}(K,N,n_s)] = p\ \mathbb{E}[G_{\mathcal{C}}(K,N,n_s) \,|\, \mathcal{C}_K = \mathcal{K}_K] + (1-p)\ E(K,N,n_s) \tag{10.4}$$

If $G_{\mathcal{K}}(K,N,n_s)$ is constant for fixed hyperparameters (K,N,n_s), then:

$$G_{\mathcal{K}}(K,N,n_s) = \mathbb{E}[G_{\mathcal{K}}(K,N,n_s) \,|\, \mathcal{C}_K = \mathcal{K}_K] = \mathbb{E}[G_{\mathcal{C}}(K,N,n_s) \,|\, \mathcal{C}_K = \mathcal{K}_K] \tag{10.5}$$

because the gains $G_{\mathcal{K}}(K,N,n_s)$ and $G_{\mathcal{C}}(K,N,n_s)$ return the same values when the real classifier $\mathcal{C}_K$ selects the right ROB. Replacing $\mathbb{E}[G_{\mathcal{C}}(K,N,n_s) \,|\, \mathcal{C}_K = \mathcal{K}_K]$ by $G_{\mathcal{K}}(K,N,n_s)$ in Equation (10.4) ends this proof. □

Two additional assumptions are made in what follows:

[A2] The classification accuracy p is modeled as a decreasing function of the number of clusters K defined on a finite interval $[\![1; K_{\max}]\!]$. Indeed, for a fixed number of training examples, increasing the number of classes K makes the classification task more complicated. When the number of classes is too large in comparison with the number of training data, the classifier hardly improves the performance of a random guess classifier.

[A3] The conditional expectation $E(K,N,n_s)$ is constant, meaning that the expected gain when choosing the wrong ROB does not depend on the hyperparameters K, N and n_s. For all K, N, n_s:

$$\begin{aligned} E := E(K,N,n_s) &= \mathbb{E}[G_{\mathcal{C}}(K,N,n_s) \,|\, \mathcal{C}_K \neq \mathcal{K}_K] \\ &\leq \mathbb{E}[G_{\mathcal{K}}(K,N,n_s) \,|\, \mathcal{C}_K \neq \mathcal{K}_K] = G_{\mathcal{K}}(K,N,n_s) \end{aligned} \tag{10.6}$$

so $E \leq G_{\mathcal{K}}(1,N,n_s) = 1$ in particular. In the application presented in the last section of this chapter, we take $E = 0.75$.

The next definition introduces the concept of *real profitability* for a dictionary-based ROM-net:

Definition 10.1.4 (Real ROM-net profitability)**.** *Given integers* $K > 1$, $N > 0$ *and* $n_s > 0$*, a dictionary-based ROM-net with classifier* $\mathcal{C}_K$ *and ROM dictionary* $\{\Psi_k^{(K,N,n_s)}\}_{k\in[\![1;K]\!]}$ *is profitable with a real profit* $G_r^* \in \mathbb{R}_+$ *if its expected gain satisfies* $\mathbb{E}[G_{\mathcal{C}}(K,N,n_s)] \geq G_r^*$.

This means that, on average, projection errors made by a global ROB are G_r^* times larger than those made by the ROM-net, even when classification errors are taken into account. However, the ROM-net profitability cannot be evaluated a priori on $\mathbb{E}[G_{\mathcal{C}}(K,N,n_s)]$, since the real classifier has not been trained yet and the dictionary of ROBs $\{\Psi_k^{(K,N,n_s)}\}_{k\in[\![1;K]\!]}$ inferred from high-fidelity snapshots have not been computed yet, see Figure 8.2. For these reasons, the following definition introduces the concept of *perfect profitability*:

Definition 10.1.5 (Perfect ROM-net profitability)**.** *Given integers $K > 1$, $N > 0$ and $n_s > 0$, a dictionary-based ROM-net with perfect classifier $\mathcal{K}_K$ and ROM dictionary $\{\Psi_k^{(K,N,n_s)}\}_{k\in[\![1;K]\!]}$ is perfectly profitable with a perfect profit $G_p^* \in \mathbb{R}_+$ if $\mathbb{E}[G_{\mathcal{K}}(K,N,n_s)] \geq G_p^*$.*

Property 10.1.6. *Let $G_r^* \in \mathbb{R}_+$. Let us consider a dictionary-based ROM-net with hyperparameters K, N, n_s. Under assumptions [A1], [A2] and [A3], the dictionary-based ROM-net is profitable with real profit G_r^* if and only if it is perfectly profitable with the following perfect profit:*

$$G_p^*(G_r^*) = \frac{G_r^* - (1 - p(K))E}{p(K)} \tag{10.7}$$

Proof. It is a direct consequence of the gain decomposition property (Property 10.1.3). □

When the gains are computed with the results of the simplified simulations and with ROBs inferred from simplified snapshots, the dictionary-based ROM-net is said to be *a priori profitable* with real profit $G_r^* > 1$ if:

$$\mathbb{E}[G_{\mathcal{K}}(K,N,n_s)] \geq \frac{G_r^* - (1 - p(K))E}{p(K)} \tag{10.8}$$

The a priori profitability can be assessed early in the ROM-net training phase, right after the physics-informed clustering procedure.

10.2 Practical method

The number of clusters K, the number of POD modes N and the number of snapshots per cluster n_s are three important hyperparameters when building a dictionary-based ROM-net. Choosing a good number of clusters K may be particularly difficult. The optimal value of K is related to the nonlinearity of the solution manifold: the more curved the solution manifold $\mathcal{M}$ is, the greater K must be to cover $\mathcal{M}$ with several subspaces. It also depends on the number of POD modes N: very fast simulations would require N to be small, which would increase the number of local bases required to cover the solution manifold. Last but not least, K also has an influence on the accuracy of the ROM-net's classifier. In a classification problem, increasing the number of classes while keeping the size of the training set constant makes the learning task tougher. Hence, the performance of a dictionary-based ROM-net does not monotonically increase with K since its classifier may choose the wrong model, leading to inaccurate numerical predictions.

The hyperparameters K, N and n_s must satisfy the following requirements:

[R1] **Limited computational resources:** the total number of high-fidelity snapshots Kn_s is limited by the maximum allowable budget in terms of high-fidelity simulations of the entire physics problem.

[R2] **Speed-up factor requirements:** to effectively reduce the computational cost of high-fidelity simulations, the number N of POD modes per local ROB must not exceed $\mathcal{N}^{1/3}$.

[R3] **Accuracy requirements:** the mean projection error must be lower than a user-defined threshold η^*:

$$\mathbb{E}[\eta\left(u(X), \Psi_{\mathcal{K}_K(X)}^{(K,N,n_s)}\right)] \leq \eta^* \tag{10.9}$$

[R4] **Gain requirements:** given a user-defined threshold $G_r^* > 1$, Equation (10.8) for the ROM-net a priori profitability must be satisfied to ensure that the local bases give better performances than a single global ROB.

Remark 10.2.1 (Concerning requirement R2)**.** *After the Galerkin projection of the governing equations onto a ROB made of N modes, the linear system to be solved at each iteration of the Newton-Raphson algorithm is full and thus has a complexity of $O(N^\alpha)$ with $2 \leq \alpha \leq 3$, which must be compared with the complexity $O(\mathcal{N}^\beta)$ of the sparse linear system obtained with the finite-element method, with $1 \leq \beta \leq 2$. The worst case is obtained for $\alpha = 3$ and $\beta = 1$, which gives an upper bound in the order of $\mathcal{N}^{1/3}$ for N.*

Given these constraints, we introduce the definition of *hyperparameters admissible set*:

Definition 10.2.2 (Hyperparameters admissible set)**.** *The hyperparameters admissible set is defined by:*

$$\mathcal{A} = \{(K, N, n_s) \in (\mathbb{N}^*)^3 \mid \text{ [R1], [R2], [R3], [R4] are satisfied.}\} \tag{10.10}$$

This definition gives a practical method for the ROM-net profitability analysis and hyperparameters tuning. The hyperparameters admissible set can be identified using simplified snapshots right after the clustering step in the training phase, see Figure 8.2. If the hyperparameters admissible set is empty, then it is not worth continuing the training phase of the dictionary-based ROM-net given the user-defined thresholds η^* and G_r^* and the maximum number of high-fidelity snapshots $n_{\text{snapshots}}^{\max}$. The user can either build a global ROB using the physics-informed clustering results to identify snapshots, or weaken some of the requirements [R1] to [R4]. The time spent for simplified simulations is not wasted: the user can justify the choice of using a global ROB, and can benefit from these simulations for high-fidelity snapshots selection. On the contrary, if the hyperparameters admissible set is not empty, then there is a benefit in using a dictionary-based ROM-net. The choice of the best hyperparameters configuration among the admissible ones depends on the user's priorities. However, given the cost of the entire training phase, a ROM-net is generally used for applications where the number of test simulations is very high, *e.g.* for parameter optimization or uncertainty quantification. In this case, once accuracy and gain requirements are met, one should take the smallest number of POD modes N to get the highest possible speed-up factor. Among the admissible configurations with the smallest number of modes, it is recommended to choose the value of K minimizing the mean projection error, to get the most accurate dictionary among the fastest admissible ones. The number of high-fidelity snapshots n_s per cluster must be fixed accordingly so that the total number Kn_s of high-fidelity snapshots remains lower than $n_{\text{snapshots}}^{\max}$.

Remark 10.2.3. *Choosing the smallest possible number of modes N generally implies choosing larger values for K, which usually decreases the performance of the ROM-net's classifier for automatic model recommendation. When interesting values for K are rather large (say greater than 8), one can artificially improve the classifier's accuracy by running several reduced simulations in parallel with the models having the highest membership probabilities. An error estimator could then be used to determine which reduced simulation*

is the most accurate, as proposed in [230]. Such a strategy increases the number of simulations to be run in the exploitation phase, but would enable working with large K's and thus small N's, lessening the computational complexity of online reduced simulations. In addition, when the number of training examples is not large enough compared to the number of clusters K for the classification task, the data augmentation algorithm presented in our paper [48] (see also Chapter 11 of this thesis report) for the classification of numerical simulations can be applied to reduce the risk of overfitting.

10.3 Back to the 1D steady heat equation

This section deals with the calibration of the hyperparameters (K, N, n_s) for the physics problem described in the previous chapter. K-medoids is applied on the training set with the ROM-oriented dissimilarity δ_n introduced in Equation (9.22). Since Equation (9.31) is time-independent, one must take $n = 1$ for the ROM-oriented dissimilarity. We simply use the notation δ instead of δ_1 (or $\tilde{\delta}_1$) for the ROM-oriented dissimilarity obtained by computing the sine dissimilarity in the solution space, whose formula is given in Equation (9.24).

The physics problem considered in this section gives only one field u per set of parameters. Therefore, the number of POD modes N is necessarily lower than or equal to the number of high-fidelity snapshots per cluster n_s. For simplification purposes, we take $N = n_s$. Given that $\mathcal{N} = 2000$, the number of POD modes N must be lower than $\lfloor \mathcal{N}^{1/3} \rfloor = 12$. To effectively reduce the computational cost of high-fidelity simulations, the maximum number of modes considered is $N = 5$.

Let us say that we are given a budget of 20 high-fidelity simulations. The hyperparameters must satisfy the inequality $Kn_s = KN \leq 20$. Our thresholds for the mean projection error and the mean gain are $\eta^* = 0.35$ and $G_r^* = 2$. A polynomial of degree 2 is considered for the model $p(K)$ for the classification accuracy, and its coefficients are determined by imposing $p(1) = 1$, $p(6) = 0.8$ (value taken from our paper [47]), $p(K_{\max}) = 1/K_{\max}$ (accuracy of a random guess for balanced classes) and $K_{\max} = 20$.

Figure 10.1 gives the mean projection error as a function of K and N. For $N = 4$ and $N = 5$ modes, the mean projection errors are below the tolerance η^* for all K. For $N = 3$, the accuracy criterion is satisfied for $K \geq 4$. The mean projection error for $N = 2$ modes falls below the tolerance for $K \geq 13$, which does not conform to the constraint $K \leq 10$ imposed by the allocated number of high-fidelity snapshots. With $N = 1$ mode, the mean projection error remains too large, which rejects configurations with $N = 1$. The configurations (K, N) satisfying the accuracy criterion and respecting the budget for high-fidelity snapshots are $K = 4, 5, 6$ for $N = 3$, $K = 2, 3, 4, 5$ for $N = 4$, and $K = 2, 3, 4$ for $N = 5$.

The gain curves are given in Figure 10.2. The dashed line in black delimits the ROM-net's profitability domain: configurations under this curve are irrelevant, either because the corresponding expected gain is too low, or because misclassification errors would be too frequent because too many classes are considered. The configuration $(K, N) = (5, 5)$ meets both gain and accuracy requirements, but violates the constraint $K \leq 4$ for $N = 5$ and thus requires too many high-fidelity snapshots. For $(K, N) = (3, 3)$, the gain is large enough but the mean projection error is larger than the tolerance, as seen in Figure 10.1. Finally, the admissible configurations are $K = 4, 5, 6$ for $N = 3$, and $K = 4, 5$

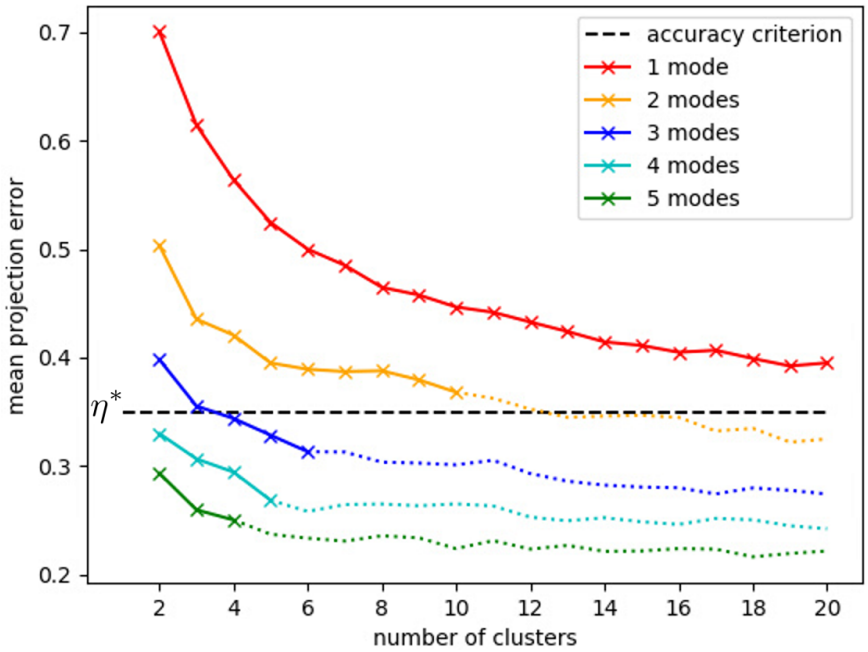

Figure 10.1: Mean projection error as a function of the number of clusters K for different number of modes N. Dotted lines indicate the configurations which do not comply with the allocated number of high-fidelity snapshots.

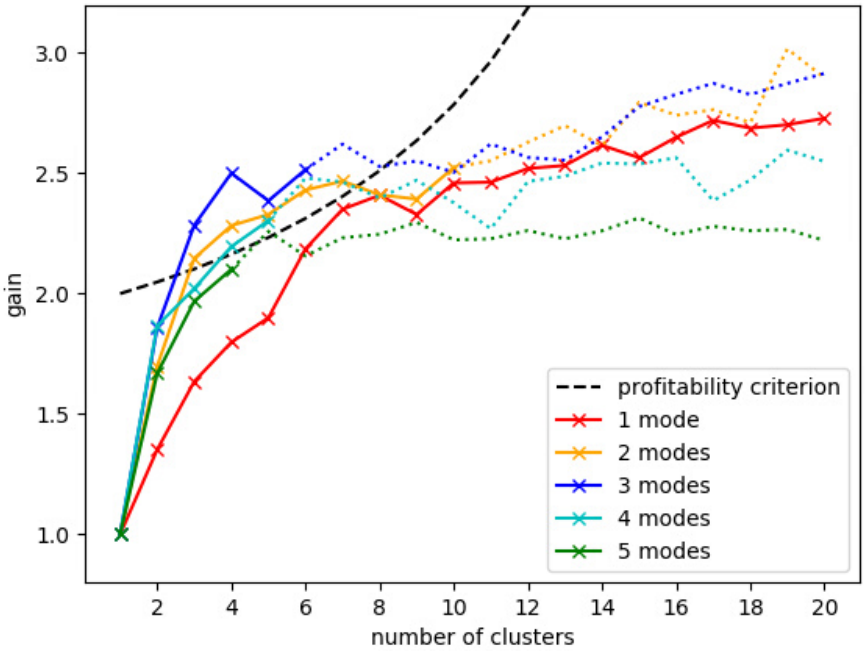

Figure 10.2: Gain as a function of the number of clusters K for different number of modes N. Dotted lines indicate the configurations which do not comply with the allocated number of high-fidelity snapshots.

for $N = 4$. The hyperparameters admissible set is represented in Figure 10.3. Among the admissible configurations, those with $N = 3$ are more interesting in terms of speed of

online reduced simulations. The lowest mean projection error is obtained for $K = 6$ when $N = 3$, see Figure 10.1. Therefore, we choose the hyperparameters $(K, N, n_s) = (6, 3, 3)$, corresponding to the lower right dot in Figure 10.3.

Remark 10.3.1. *It has been decided to take the configuration with the best accuracy among the admissible configurations with the smallest value for N, in order to have a simple and systematic approach for hyperparameters calibration. However, in the present example, one could also use the elbow method. The elbow method is commonly used for selecting the number of clusters for k-means clustering or the number of components for a PCA. It consists in choosing the elbow or knee point of the curve of an evaluation criterion. In spite of the difficulties of defining clearly the elbow point in some situations, this method raises interesting questions. In our example, if one uses the elbow method with the error curve, the best number of clusters is still $K = 6$: for $N = 3$, $K = 6$ is the elbow point. When using this method with the gain curve, the best number of clusters turns out to be $K = 4$, even when considering a smoothed version of the blue curve in Figure 10.2 to avoid undesirable fluctuations due to sampling and medoids initializations. Indeed, taking $K = 5$ or $K = 6$ does not significantly improve the gain when $N = 3$, whereas the number of high-fidelity snapshots and the complexity of the classification problem would be increased. The practical method presented in this chapter can be adapted according to the user's priorities between training cost, online speed, accuracy, and gain.*

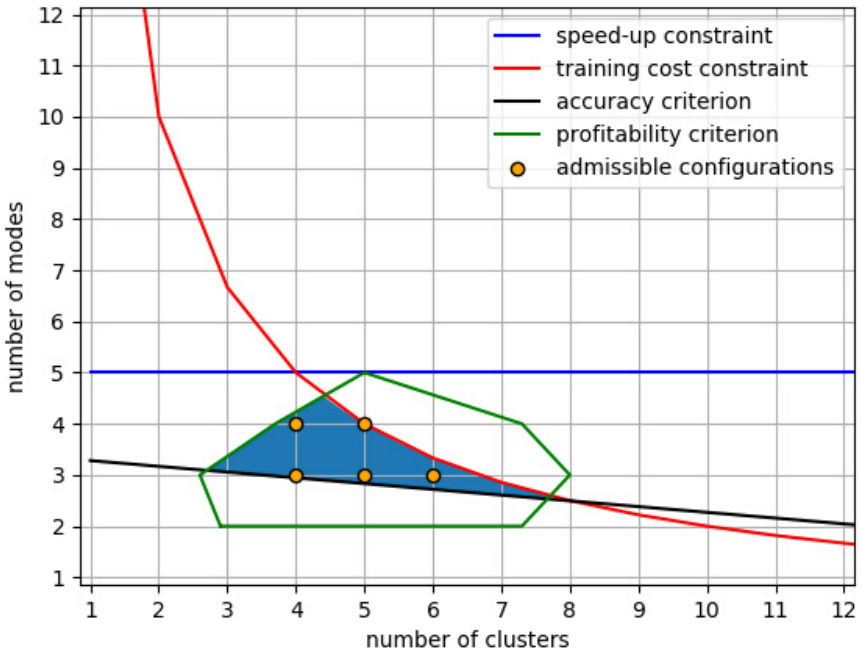

Figure 10.3: Hyperparameters admissible set.

* *
*

Chapter 11

Classification for automatic model recommendation

Abstract: *Our methodology resorts to classification algorithms for the selection of local ROMs adapted to the environment and the state of the physical system. For this classification task, labeled training data come from numerical simulations and correspond to physical fields discretized on a mesh. Three challenging difficulties arise: the lack of training data, their high dimensionality, and the non-applicability of common data augmentation techniques to physics data. This chapter introduces two algorithms to address these issues: one for dimensionality reduction via feature selection, and one for data augmentation. These algorithms are combined with a wide variety of classifiers for their evaluation.*

Remark 11.0.1. *This chapter is taken from our paper [48], with some modifications and novelties.*

Contents

11.1 Challenges to be addressed

Working with high-dimensional data increases the risk of overfitting and slows down the training process for the classifier, which restrains the exploration of the hyperparameters space. We have seen that feature selection can be used to identify a reduced number of features containing sufficient information for classification. The minimum redundancy maximum relevance (mRMR) algorithm [44, 45] tries to find a trade-off between the relevance of the selected features and their redundancy. However, for very large numbers of features like in computational physics, evaluating the redundancy is very computationally demanding. Fortunately, working on physics data provides other possibilities to define a redundancy measure. In this chapter, we propose a new feature selection algorithm suitable for features coming from the same physical quantity but corresponding to different points in a space-time discretization. It is assumed that this physical quantity, defined as a function of space and/or time, has some smoothness properties. This is often the case in physics, where the physical quantity satisfies partial differential equations and boundary conditions. In [233], it is shown that the solution of Poisson's equation on a Lipschitz domain in $\mathbb{R}^3$ with a L^2 source term and Dirichlet or Neumann boundary conditions is continuous. Poisson's equation is well known in physics, and can be found, for example, in electrostatics, in Gauss's law for gravity, in the stationary heat equation, and in the stationary particle diffusion equation. If the features of a random vector contain the discretized values of a smooth function of space and time, then their correlations are related to their proximities on the space-time grid. The approach presented in this chapter is depicted as a geostatistical variant of mRMR algorithm, in the sense that it consists in modeling the redundancy as a function of space and time.

Once the dimension of the input space is reduced, another challenge of the classification problem must be addressed: the lack of training data. *Data augmentation* refers to techniques aiming at enlarging the training set by generating new examples from the original ones. For image classification, many class-preserving operations can be used to create new images, such as translations, rotations, cropping, scaling, and changes in colors, brightness and contrast. Unfortunately, these common techniques cannot be used when considering physics data. For this type of data, new examples can be generated using generative adversarial networks (GAN [234]; see in [235] for the use of deep convolutional GANs in computational fluid dynamics). However, training GANs is quite complex in practice and may also be made more difficult by the lack of training examples. More simply, new data can be generated by convex combinations of the original examples. SMOTE [236] takes convex combinations of input data with their nearest neighbors in the input space. ADASYN [237] uses the same idea but focuses more on examples that are hard to learn, *i.e.*, those having examples of a foreign class in their neighborhoods. Both data augmentation algorithms use k-nearest neighbors algorithm and thus compute Euclidean distances in the input space. When working on high-dimensional physics data, this approach may suffer from the curse of dimensionality [80]. In addition, defining neighborhoods with the Euclidean distance in the input space is not always appropriate, as dictionary-based ROM-nets use physics-aware dissimilarities to label the data. The data augmentation algorithm developed in this chapter consists in growing sets around original examples by incrementally adding nearest neighbors in terms of the dissimilarity measure used for the automatic data labeling procedure presented in Section 8.3 and Chapter 9. These sets are used to generate new data by convex combinations. Contrary to SMOTE and ADASYN, the risk of generating new data with wrong labels is controlled by checking that the convex

hulls of the growing sets do not contain any example belonging to a foreign class.

In sum, the contributions of this chapter are motivated by difficulties that are inherent to classification tasks on simulation data and that can be summarized in three main issues:

- the lack of training data due to the expensive data labeling procedure involving simulations with a high-fidelity model (risk of overfitting),
- the high dimensionality of input data (risk of overfitting), and
- most common data augmentation techniques are not applicable to physics data.

The feature selection and data augmentation strategies introduced in this chapter are developed to tackle these difficulties.

11.2 Test case

Notations: The j-th feature of a random vector X is the real-valued random variable denoted by X^j. Its observations are denoted by x^j, or x_i^j when indexing is necessary, for example when considering training data. When X is obtained by discretizing a random field on a mesh, the feature X^j corresponds to the value taken by the random field at the j-th node. In the numerical application presented in this work, a random temperature field is considered. The spatial coordinates of the j-th node are stored in a vector $\boldsymbol{\xi}_j \in \mathbb{R}^3$. The categorical variable Y indicates which model should be used. □

In this work, input data $\{x_i\}_{1\leq i\leq m}$ correspond to several instances or *variabilities* of a physical field discretized on a mesh. Let $\mathcal{N}$ be the number of nodes in the mesh. If the physical field is scalar and defined at the nodes, then each observation x_i is a vector of $\mathbb{R}^{\mathcal{N}}$. For relatively small problems, $\mathcal{N}$ is in the order of 10^4 to 10^5. For some industrial problems, $\mathcal{N}$ can be in the order of 10^6 to 10^8. The dataset $\{x_i\}_{1\leq i\leq m}$ may come from experiments, numerical simulations, statistical models, or a combination of them, and contains from 10^2 to 10^4 observations. It is assumed that all features of all observations are known, contrary to some classification tasks in other disciplines encountering the problem of missing values. This assumption is clearly satisfied when data come from numerical simulations or statistical models. For experimental data, numerous techniques provide space-distributed measurements that can be projected onto the mesh, such as particle image velocimetry [238] in fluid dynamics, digital image correlation [239] and photoelastic experiments [240] in solid mechanics, and temperature-sensitive paints [241] measuring surface temperatures.

The framework considered in this chapter is the same as in the previous chapters, where the input variabilities are supposed to be used for an uncertainty propagation study in a physics problem $\mathcal{P}$, for which a high-fidelity model m_{HF} is available. The physics problem $\mathcal{P}$ is a time-dependent problem. As the high-fidelity model is too computationally expensive, dictionary-based ROM-nets have been introduced to reduce the computation time by means of a reduced-order model dictionary and a classifier playing the role of a model selector. The dictionary-based ROM-net is trained on the available dataset $\{x_i\}_{1\leq i\leq m}$. For

a given observation x_i, the class label y_i indicates the most appropriate model in the dictionary to be used for fast simulations with limited errors with respect to the high-fidelity model m_{HF}. Class labels are obtained by the following data labeling procedure[1]:

- **Step 1:** For each observation x_i in the dataset, use the high-fidelity model m_{HF} to solve a simplified version $\mathcal{P}'$ of the physics problem $\mathcal{P}$ (for example, the problem $\mathcal{P}'$ can consist in solving $\mathcal{P}$ for a few time increments only). The primal solution of $\mathcal{P}'$ computed for x_i is denoted by u_i. It consists of a collection $\{u_i^n\}_{1\leq n\leq n_t}$ of n_t fields defined on the mesh, with n_t being the number of time increments in problem $\mathcal{P}'$.
- **Step 2:** Given $\{u_i\}_{1\leq i\leq m}$, compute the dissimilarity matrix $\boldsymbol{\delta} \in \mathbb{R}^{m\times m}$ with the following formula:

$$\delta_{ij} = \delta(x_i, x_j) = d_{\mathrm{Gr}(\infty,\infty)}\left(\mathrm{span}(\{u_i^n\}_{1\leq n\leq n_t}), \mathrm{span}(\{u_j^n\}_{1\leq n\leq n_t})\right)$$

  with $d_{\mathrm{Gr}(\infty,\infty)}$ being the Grassmann metric defined in [242]. The coefficient δ_{ij} is a dissimilarity measure between x_i and x_j.
- **Step 3:** k-medoids clustering [84] is applied to the dissimilarity matrix $\boldsymbol{\delta}$. In this chapter, we consider $K = 4$ clusters. The label $y_i = \mathcal{K}(x_i) \in [\![1; K]\!]$ is given by the index of the cluster containing u_i.

This procedure gives $m = 1000$ examples of input–label pairs $\{(x_i, y_i)\}_{1\leq i\leq m}$. This dataset is split into a training set, a validation set, and a test set with cardinalities 600, 200, and 200, respectively, enabling the supervised training and evaluation of a classifier $\mathcal{C}$. For the sake of simplicity, the labeled data are renumbered so that the $m_{\mathrm{train}} = 600$ first input–output pairs $\{(x_i, y_i)\}_{1\leq i\leq m_{\mathrm{train}}}$ form the training set on which the feature selection and data augmentation algorithms presented in this chapter are trained.

In this work, the physics problem $\mathcal{P}$ is a temperature-dependent mechanical problem. The structure is made of an elasto-viscoplastic material whose behavior depends on the local value of the temperature field [243]. The random variable X is a random vector representing the evaluation of the random temperature field on a finite-element mesh containing $\mathcal{N} = 42445$ nodes (see Figure 11.1). The structure is subjected to centrifugal forces and pressure loads. The random temperature fields are generated by a stochastic model described in [47], where ten fluctuation modes are randomly combined and superposed to a reference temperature field. The realizations of the random temperature field are continuous and always satisfy the heat equation. Modeling random fields as random combinations of deterministic spatial functions is quite common when studying stochastic partial differential equations [244, 245, 246], because a random field can be approximated by truncating its Karhunen–Loève expansion [247]. More information about the stochastic model generating random temperature fields will be given in Chapter 12.

As already stated, the main contributions presented in this chapter are a feature selection strategy and a data augmentation algorithm adapted to the specificities and difficulties of classification problems encountered when training dictionary-based ROM-nets. Concerning feature selection, the main focus is on the fast quantification of features redundancy by taking advantage of the type of input data. Concerning data augmentation,

[1] This is the first version of the ROM-net's physics-informed clustering procedure that was presented in [47], using Grassmann distances. This procedure slightly differs from the one presented in this thesis, with the ROM-oriented dissimilarity measure based on the sine dissimilarity.

in addition to the constraints that have already been mentioned, it is likely that transforming an input example x_i substantially modifies the intermediate variable u_i, and thus the class label y_i might no longer be relevant for the transformed input. Avoiding this situation is crucial to ensure that the augmented data are correctly labeled. Our algorithms are applicable under the assumptions that the random vector X derives from a random field whose realizations are continuous with probability one (sample path continuity, see Definition 2.3.8) and belong to a convex domain $\mathcal{X}$ related to physics constraints. Last, a comparison of various classification algorithms is conducted to put into perspective the choice made in our first paper [47] to use an ensemble of deep neural networks for the ROM-net's classifier.

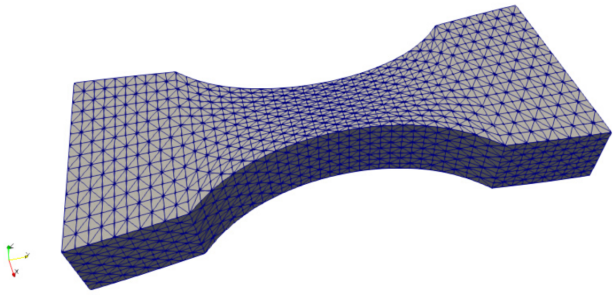

Figure 11.1: Finite-element mesh of the structure considered in this chapter.

Remark 11.2.1. *Another strategy would consist in using a regression algorithm for the classification task. Indeed, as our data labeling procedure is based on clustering, the classification problem could be replaced by a regression problem for the prediction of dissimilarities* $\{\delta(x, \tilde{x}_k)\}_{1 \leq k \leq K}$ *for* $x \in \mathcal{X}$*, with* $\tilde{x}_k$ *being the medoid of the* k*-th cluster. Given these distances for a new observation* x*, the class label is obtained by taking the integer* $k \in [\![1; K]\!]$ *associated to the smallest dissimilarity* $\delta(x, \tilde{x}_k)$*. However, the data augmentation algorithm presented in this chapter is not compatible with regression algorithms. For this reason, this chapter focuses on classifiers rather than regressors.*

11.3 Feature selection

11.3.1 A geostatistical variant of mRMR feature selection

When training dictionary-based ROM-nets, the number of features of the random vector X scales with the number of nodes $\mathcal{N}$ in the mesh. In particular, the number of features is exactly $\mathcal{N}$ if X is the nodal representation of a scalar field. Therefore, there are too many features to compute all redundancy terms $I(X^i, X^j)$. However, one can estimate the redundancy terms thanks to the proximities of the features on the mesh. Indeed, X is a regionalized variable: in our example, we recall that $\boldsymbol{\xi}_i \in \mathbb{R}^3$ denotes the position of the i-th node in the mesh, and that the feature X^i corresponds to the value taken by a random temperature field at $\boldsymbol{\xi}_i$. If two points $\boldsymbol{\xi}_i$ and $\boldsymbol{\xi}_j$ of the mesh are close to each other, the corresponding features X^i and X^j are likely to be correlated and thus redundant because of the smoothness of the temperature field. This idea is also valid when considering physical variables discretized in time.

The random temperature field is modeled by a Gaussian random field [62] as in [47], which is a common and simple approach when modeling uncertainties on a physical field.

As a consequence, X is a Gaussian random vector and the mutual information $I(X^i, X^j)$ has a simple formula involving the correlation coefficient:

Property 11.3.1 (Mutual information of two correlated Gaussian random variables [58], eq. 8.56, p. 252)**.** *Let (X^1, X^2) be a Gaussian random vector. The mutual information $I(X^1, X^2)$ reads*

$$I(X^1, X^2) = -\frac{1}{2}\ln\left(1-\rho^2\right) \tag{11.1}$$

where ρ denotes the correlation between X^1 and X^2.

This property implies that, for Gaussian random fields having isotropic correlation functions[2] ρ, the mutual information $I(X^i, X^j)$ only depends on the distance $||\boldsymbol{\xi}_i - \boldsymbol{\xi}_j||_2$. A wide variety of isotropic correlation functions are given in [62]. More generally, as Equation (11.1) is an increasing function of ρ^2, any isotropic upper (resp. lower) bound of the squared correlation function gives an isotropic upper (resp. lower) bound of the mutual information.

For the example studied in this chapter, Figure 11.2 shows that the mutual information $I(X^i, X^j)$ decreases as the corresponding distance $||\boldsymbol{\xi}_i - \boldsymbol{\xi}_j||_2$ increases. Therefore, our feature selection algorithm builds a metamodel $\tilde{I}$ replacing $I(X^i, X^j)$ by a function of the distance $||\boldsymbol{\xi}_i - \boldsymbol{\xi}_j||_2$, which drastically reduces the computational cost of mRMR algorithm for our particular problem. First of all, one must build a design of experiments (DoE) to select a few terms $I(X^i, X^j)$ to be computed exactly. The metamodel $\tilde{I}$ is calibrated to fit the corresponding precomputed redundancy terms. Then, mRMR feature selection is applied by replacing $I(X^i, X^j)$ with $\tilde{I}(||\boldsymbol{\xi}_i - \boldsymbol{\xi}_j||_2)$. The feature selection algorithm is described in Algorithm 1. We call this algorithm *geostatistical mRMR*, as geostatistics is the branch of statistics that deals with regionalized variables. A stopping criterion is added to the incremental procedure used in mRMR, enabling an automatic selection of the number of features to be kept for the classification task: the algorithm stops when the value of $\underset{i\in[\![1;\mathcal{N}]\!]\setminus S_k}{\arg\max}\left(I(X^i,Y) - \frac{1}{k}\sum_{j\in S_k}\tilde{I}\left(\|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|_2\right)\right)$ has not changed much during a number of iterations. A condition on the mutual information $I(X^i, Y)$ can also be added to avoid selecting quasi-irrelevant features. It should be noted that the number of selected features does not depend on the number of nodes $\mathcal{N}$ in the mesh. In addition, for stage 1 of Algorithm 1, computing all the terms $\|\boldsymbol{\xi}_{j_1} - \boldsymbol{\xi}_{j_2}\|_2$ of the matrix of pairwise mesh nodes distances is not necessary: only a few lines of this matrix corresponding to randomly selected nodes are evaluated, which is sufficient to build the DoE. In other words, one computes the distances between a few nodes and all the mesh nodes.

[2] The correlation function $\rho(\boldsymbol{\xi}, \boldsymbol{\xi}')$ of a random field is isotropic if it only depends on the distance $||\boldsymbol{\xi} - \boldsymbol{\xi}'||_2$.

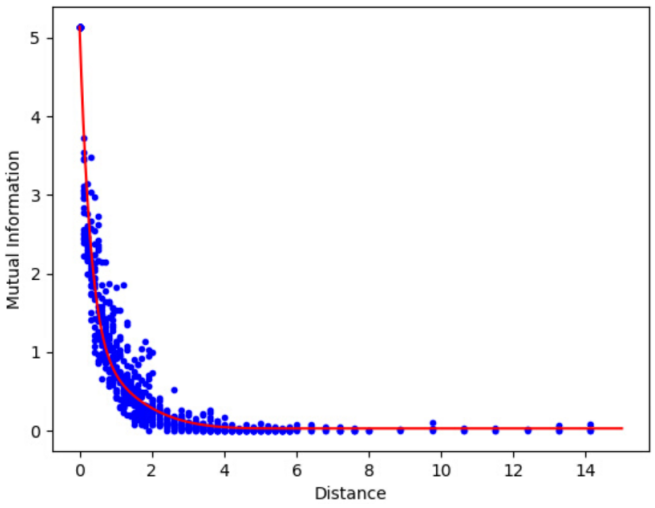

Figure 11.2: Mutual information $I(X^i, X^j)$ as a function of the distance $||\boldsymbol{\xi}_i - \boldsymbol{\xi}_j||_2$.

Algorithm 1 Geostatistical mRMR

Input: training set $\{(x_i, y_i)\}_{1 \leq i \leq m_{\text{train}}}$, set of mesh nodes $\{\boldsymbol{\xi}_i\}_{1 \leq i \leq \mathcal{N}}$, stopping criterion.
Output: set of selected features.

Stage 1 (design of experiments):
  Select distance values r_j.
  For each r_j, draw n_j pairs of mesh nodes $(\boldsymbol{\xi}_{j_1}, \boldsymbol{\xi}_{j_2})$ such that $\|\boldsymbol{\xi}_{j_1} - \boldsymbol{\xi}_{j_2}\|_2 \approx r_j$.
Stage 2 (metamodel for redundancy terms):
  Compute the mutual information $I(X^i, X^j)$ for each pair selected in Stage 1.
  Train a metamodel $\tilde{I}$ such that $I(X^i, X^j) \approx \tilde{I}\left(\|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|_2\right)$.
Stage 3 (compute relevance terms):
  Compute $I(X^i, Y)$ for all $i \in [\![1; \mathcal{N}]\!]$.
Stage 4 (greedy feature selection):
  $S_1 := \underset{i \in [\![1;\mathcal{N}]\!]}{\arg\max}\ I(X^i, Y)$
  $k := 1$
  while stopping criterion not satisfied **do**
    $S_{k+1} := S_k \cup \{ \underset{i \in [\![1;\mathcal{N}]\!] \setminus S_k}{\arg\max} \left(I(X^i, Y) - \frac{1}{k} \sum_{j \in S_k} \tilde{I}\left(\|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\|_2\right) \right) \}$
    $k := k + 1$
  end while
return S_k

Remark 11.3.2. *A parallel can be drawn between our feature selection strategy and hyper-reduction methods [33, 185, 189, 34] used to accelerate complex nonlinear problems in physics (see in [248] for design optimization and [35] for large-scale simulations). Hyper-reduction methods aim at finding a reduced set of integration points in the finite-element mesh that is sufficient to predict the behavior of the physical system. The constitutive equations are solved on this reduced-integration domain only, while the values of quantities of interest at the remaining integration points can be recovered with the Gappy POD [46]. In short, hyper-reduced solvers make predictions from a reduced number of points in a mesh,*

like the classifiers used in this chapter do when combined with the geostatistical mRMR. Although the objectives are different, both hyper-reduction and geostatistical mRMR feature selection benefit from the properties of physics data to reduce the complexity of numerical tasks.

11.3.2 Numerical results

The red curve on Figure 11.2 corresponds to the metamodel estimating redundancy terms. In this example, we choose

$$\tilde{I}(r) = I_\infty + \gamma_1(r_1 - r)^{\alpha_1} H(r_1 - r) + \gamma_2(r_2 - r)^{\alpha_2} H(r_2 - r) \tag{11.2}$$

where H is the Heaviside step function and $I_\infty, \gamma_1, \gamma_2, r_1, r_2, \alpha_1, \alpha_2$ are calibration parameters that are adjusted manually. In the DoE, the step between distances r_j is smaller for small distances, in order to better capture the evolution of the mutual information in its high gradient regime. The number n_j of pairs of nodes separated by a distance of r_j selected in the DoE also depends on r_j: as higher variances were expected for small distances, n_j decreases when r_j increases. In total, 749 terms $I(X^i, X^j)$ are computed, which takes 5.12 seconds using Scikit-learn [249]. Building the DoE takes only 0.33 seconds. Then, the greedy procedure takes 303 seconds and selects 87 features among the 42445 original ones. The first iteration is the longest one with 276 seconds, because it includes the computation of all the relevance terms $I(X^i, Y)$. As a comparison, the original mRMR algorithm takes 6469 seconds to compute 7 iterations only. We did not let mRMR algorithm go further, as the per-iteration computation time grows with the iteration number. For a fair comparison, our implementations of mRMR and stages 3 and 4 of the geostatistical mRMR are the same except that redundancy terms are evaluated with Scikit-learn for mRMR and with the function $\tilde{I}$ for the geostatistical mRMR.

Table 11.1 compares the relevance $D(S, Y)$, the true redundancy $R(S)$, the approximate redundancy $\tilde{R}(S)$ estimated with $\tilde{I}$, the true cost function $D(S, Y) - R(S)$, and the approximate cost function $D(S, Y) - \tilde{R}(S)$ for three different feature selection strategies:

- the geostatistical mRMR feature selection (Algorithm 1), selecting a set S^* of features;
- a univariate filter algorithm selecting the features with the highest mutual information (MI) scores $I(X^i, Y)$. This algorithms finds a set S^{MI} maximizing the relevance for a given cardinality; and
- a purely geometric feature selection algorithm, randomly selecting the first feature and adding features in a greedy manner so that the distance to the closest point $\boldsymbol{\xi}_i,\ i \in S_k$ is maximized. This algorithm tends to select a set S^G of well-distributed features in order to get a low redundancy for a given cardinality.

As the geostatistical mRMR automatically selected 87 features, the two other approaches are applied with $|S^G| = |S^{MI}| = 87$ as a target. Table 11.1 shows that the relevance of the set S^* selected by our algorithm is in the same order of magnitude as the relevance of the set S^{MI}. Its redundancy is in the same order of magnitude as the redundancy of the set S^G. These results show that the geostatistical mRMR algorithm does have the desired behavior: it selects a subset of features S^* with high relevance and low redundancy.

Figure 11.3 shows the features selected by the three different algorithms. The classification accuracies of several classifiers using the reduced features S^* are given at the end of this chapter.

Table 11.1: Evaluation of the geostatistical mRMR feature selection algorithm.

Algorithm	$D(S,Y)$	$\tilde{R}(S)$	$R(S)$	$D(S,Y) - \tilde{R}(S)$	$D(S,Y) - R(S)$
Geostat. mRMR (S^*)	0.0460	0.0816	0.1111	-0.0356	-0.0651
MI-based filter (S^{MI})	0.0671	0.9794	0.8129	-0.9124	-0.7458
Geometric filter (S^G)	0.0090	0.0788	0.1072	-0.0699	-0.0982

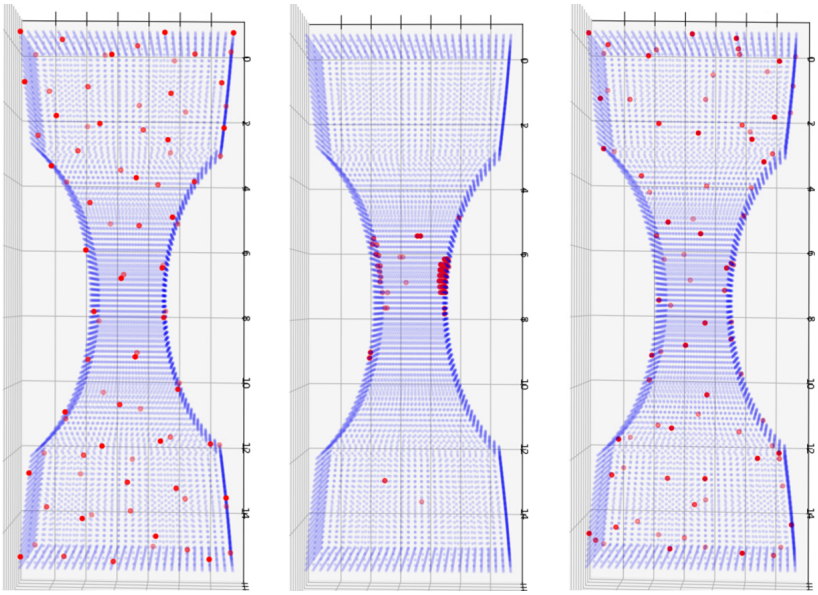

Figure 11.3: Red dots indicate the selected features. From the left to the right: geometric feature selection, MI-based feature selection, and geostatistical mRMR.

Remark 11.3.3. *The geometric feature selection algorithm gives rather good results in terms of the cost function, but it does not mean that it is an appropriate approach. Indeed, one can see that the relevance of S^G is very low, as this algorithm does not use any information concerning the classification problem.*

11.4 Data augmentation

This section introduces a new data augmentation algorithm based on the concept of *pure sets*. As already explained, the aim of data augmentation is to generate new labeled examples from training data, in order to enlarge the training set for machine learning problems on small datasets. The main difficulty is to find class-preserving transformations ensuring that the generated examples have the correct label.

11.4.1 Pure sets

Definition 11.4.1 (Convex set [250], p. 10)**.** *Let V be a real vector space. A non-empty set $\mathcal{S} \subset V$ is convex if*

$$\forall (x_1, x_2) \in \mathcal{S}^2, \ \forall \lambda \in [0;1], \quad \lambda x_1 + (1-\lambda) x_2 \in \mathcal{S} \tag{11.3}$$

Definition 11.4.2 (Convex combination [250], p. 11)**.** *Let $\{x_i\}_{1 \leq i \leq n}$ be a finite set of elements of a real vector space V. A convex combination of $\{x_i\}_{1 \leq i \leq n}$ is a vector $x \in V$ such that*

$$\exists \ (\lambda_i)_{1 \leq i \leq n} \in \mathbb{R}_+^n \quad | \quad \sum_{i=1}^{n} \lambda_i = 1 \quad \textit{and} \quad x = \sum_{i=1}^{n} \lambda_i x_i \tag{11.4}$$

Definition 11.4.3 (Convex hull of a set [250], p. 12)**.** *Let V be a real vector space and $\mathcal{S}$ a non-empty set included in V. The convex hull or convex envelope $\mathcal{E}(\mathcal{S})$ of $\mathcal{S}$ is the smallest convex set containing $\mathcal{S}$. Equivalently, the convex hull $\mathcal{E}(\mathcal{S})$ can be defined as the set of all convex combinations of all finite subsets of $\mathcal{S}$.*

Property 11.4.4 (Image of a convex hull by a linear map)**.** *Let V and W be two real vector spaces, and let $\mathcal{L} : V \to W$ be a linear map. Let $\mathcal{S}$ be a non-empty set included in V. Then,*

$$\mathcal{L}\left(\mathcal{E}(\mathcal{S})\right) = \mathcal{E}\left(\mathcal{L}(\mathcal{S})\right) \tag{11.5}$$

Proof. Let $z \in \mathcal{E}\left(\mathcal{L}(\mathcal{S})\right)$. Following the definition of a convex hull, there exists $n \in \mathbb{N}^*$ such that

$$\exists \ (w_i)_{1 \leq i \leq n} \in \mathcal{L}(\mathcal{S})^n, \ \exists \ (\lambda_i)_{1 \leq i \leq n} \in \mathbb{R}_+^n \quad | \quad \sum_{i=1}^{n} \lambda_i = 1 \text{ and } z = \sum_{i=1}^{n} \lambda_i w_i \tag{11.6}$$

For all $i \in [\![1; n]\!]$, as $w_i \in \mathcal{L}(\mathcal{S})$, there exists $v_i \in S$ such that $w_i = \mathcal{L}(v_i)$. By linearity of $\mathcal{L}$:

$$z = \sum_{i=1}^{n} \lambda_i \mathcal{L}(v_i) = \mathcal{L}\left(\sum_{i=1}^{n} \lambda_i v_i\right) \in \mathcal{L}\left(\mathcal{E}(\mathcal{S})\right) \tag{11.7}$$

so $\mathcal{E}\left(\mathcal{L}(\mathcal{S})\right) \subset \mathcal{L}\left(\mathcal{E}(\mathcal{S})\right)$. The other inclusion can be shown using exactly the same arguments. Thus, $\mathcal{L}\left(\mathcal{E}(\mathcal{S})\right) = \mathcal{E}\left(\mathcal{L}(\mathcal{S})\right)$. □

This property has a very simple yet important consequence for the data augmentation algorithm presented in this chapter.

Property 11.4.5. *Let V and W be two real vector spaces, and let $\mathcal{L} : V \to W$ be a linear map. Let $\mathcal{S}$ be a non-empty set included in V. Then, for all $x \in V$:*

$$\mathcal{L}(x) \notin \mathcal{E}\left(\mathcal{L}(\mathcal{S})\right) \Rightarrow x \notin \mathcal{E}(\mathcal{S}) \tag{11.8}$$

Proof. By contraposition, $x \in \mathcal{E}(\mathcal{S}) \Rightarrow \mathcal{L}(x) \in \mathcal{L}\left(\mathcal{E}(\mathcal{S})\right) = \mathcal{E}\left(\mathcal{L}(\mathcal{S})\right)$. □

Our data augmentation strategy uses this property in the particular case where the linear map is a projection. As a reminder, the notation $\mathcal{K}$ stands for the true classifier assigning any input x to a single label $y \in [\![1; K]\!]$. Before giving the description of the algorithm, let us introduce the definition of *pure sets* in a labeled dataset and a proposition for the characterization of pure sets.

Definition 11.4.6 (Pure set)**.** *Let n be a positive integer, and let $\mathcal{S} = \{x_i\}_{1 \leq i \leq n}$ be a finite set of elements of a real vector space V labeled by $\mathcal{K}$. Let $\mathcal{S}_\mathcal{I} = \{x_i\}_{i \in \mathcal{I} \subset [\![1;n]\!]}$ be a non-empty subset of $\mathcal{S}$. The set $\mathcal{S}_\mathcal{I}$ is pure in $\mathcal{S}$ if $\mathcal{K}(\mathcal{S} \cap \mathcal{E}(\mathcal{S}_\mathcal{I}))$ is a singleton, which means that the set $\mathcal{S}_\mathcal{I}$ is pure in $\mathcal{S}$ if all of the points of $\mathcal{S}$ that belong to the convex hull of $\mathcal{S}_\mathcal{I}$ have the same label.*

Let $\mathcal{S} = \{x_i\}_{1 \leq i \leq n}$ be a finite set of elements of a finite-dimensional real vector space V labeled by integers $\{y_i\}_{1 \leq i \leq n}$ in $[\![1; K]\!]$, with $K \leq n$. For all $k \in [\![1; K]\!]$, C_k denotes the set of elements of $\mathcal{S}$ labeled by k:

$$C_k = \{x_i \in \mathcal{S} \mid y_i = k\} \tag{11.9}$$

For any subset S_k of C_k with cardinality $|S_k|$, $\hat{A}_{S_k} \in \mathbb{R}^{\dim(V) \times |S_k|}$ denotes the matrix whose columns contain the coordinates of the elements of S_k. The matrix denoted by A_{S_k} is obtained by adding a row of ones below the matrix $\hat{A}_{S_k}$, giving a matrix of size $(1 + \dim(V)) \times |S_k|$.

Proposition 11.4.7 (Pure set characterization)**.** *Let S_k be a subset of C_k with cardinality $|S_k|$. The set S_k is pure in $\mathcal{S}$ if and only if for all x in $\mathcal{S} \setminus C_k$ the linear system:*

$$A_{S_k} w = \begin{pmatrix} x \\ 1 \end{pmatrix} \tag{11.10}$$

has no nonnegative solution $w \in \mathbb{R}_+^{|S_k|}$.

Proof. Let $x \in \mathcal{S} \setminus C_k$. Equation (11.10) has no nonnegative solution if and only if

$$\nexists\, w \in \mathbb{R}_+^{|S_k|} \quad | \quad \sum_{i=1}^{|S_k|} w_i = 1 \text{ and } \hat{A}_{S_k} w = x \tag{11.11}$$

$$\iff x \notin \mathcal{E}(S_k) \tag{11.12}$$

which ends the proof. □

Corollary 11.4.8 (Pure set testing)**.** *Let S_k be a subset of C_k with cardinality $|S_k|$, and let $\mathcal{L} : V \to W$ be a linear map, where W is a finite-dimensional real vector space. If for all x in $\mathcal{S} \setminus C_k$ the linear system*

$$A_{\mathcal{L}(S_k)} w = \begin{pmatrix} \mathcal{L}(x) \\ 1 \end{pmatrix} \tag{11.13}$$

has no nonnegative solution in $\mathbb{R}_+^{|S_k|}$, then S_k is pure in $\mathcal{S}$.

Proof. Equation (11.13) characterizes the purity of $\mathcal{L}(S_k)$ in $\mathcal{L}(\mathcal{S})$ (Proposition 11.4.7), which implies that S_k is pure in $\mathcal{S}$ (Property 11.4.5). □

Figure 11.4 illustrates the concept of pure sets. On this figure, the set C_1 is made of all the elements represented by dots, while the crosses form the set $C_2 = \mathcal{S} \setminus C_1$. On the left, the subset formed by the six black dots is pure since its convex hull delimited by dashed lines contains only dots. The subset made of the six black dots on the right-hand side of the figure is not pure because of the presence of a cross in its convex hull. Equation (11.10) has a nonnegative solution when using the coordinates of this cross in its right-hand side.

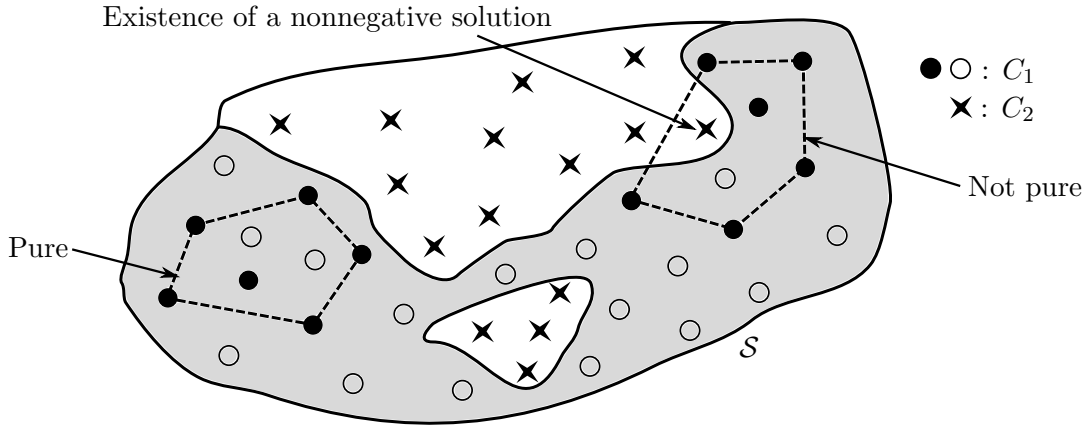

Figure 11.4: Illustration of the concept of pure sets on a binary classification problem.

11.4.2 The data augmentation algorithm

The objective is to generate new data points $x \in \mathcal{X}$ in a given class $y \in [\![1; K]\!]$ from the preexisting observations in that class. To this end, one must apply class-preserving transformations on the training examples. New examples can be created by taking convex combinations of some subsets of the training set, for example. One way of controlling the risk that newly generated examples have wrong labels is to take convex combinations of subsets only if they are pure. Indeed, if the k-th class C_k contains a set S_k that is pure in the training set, one can expect that the probability $\mathbb{P}(Y = k \mid X \in \mathcal{E}(S_k))$ is high enough to get new examples of class C_k by drawing samples in $\mathcal{E}(S_k)$. In addition, the third Hadamard well-posedness condition states that the solution of a physics equation changes continuously with respect to the parameters of the problem. In the neighborhood of a point x_0 belonging to a pure set S_k, the primal solution u stays in the neighborhood of the solution u_0 obtained with x_0 and is thus likely to have the same label. Therefore, the objective of our algorithm is to find pure sets in the training set in order to generate new examples by convex combinations with a limited risk of getting incorrectly labeled examples. The pure sets detected by the algorithm are listed in a matrix of lists $\mathcal{S}$ such that $\mathcal{S}[k, i]$ contains the indices of the training points forming the i-th pure set of the k-th class. The pure sets are grown from different starting points or seeds in the training set by iteratively adding the seeds' nearest neighbors in terms of the precomputed dissimilarity measure δ used for clustering in the data labeling procedure. The growth stops before losing the purity of the subsets. However, checking the purity in the high-dimensional input space can cause difficulties, even when training the data augmentation algorithm after a first dimensionality reduction like in this chapter. For this reason, the algorithm checks the purity after having applied a feature selector π_S with a small random subset of features S containing d features. Let us apply Property 11.4.5 with $V = W$ being the input vector space containing $\mathcal{X}$ and with the linear map $\mathcal{L}$ being the feature selector π_S.

As Property 11.4.5 states, if no point of $\pi_S\left(\{x_j\}_{1\leq j\leq m_{\text{train}}} \setminus C_k\right)$ belongs to the convex hull of $\pi_S\left(\{x_j\}_{j\in\mathcal{S}[k,i]}\right)$, then the set $\mathcal{E}\left(\{x_j\}_{j\in\mathcal{S}[k,i]}\right)$ does not contain any point labeled with $k' \neq k$. As a set can lose its purity after projection, the algorithms tries $p_{\max}$ random feature selectors π_S before considering that the set is not pure. In practice, the purity of $\pi_S\left(\{x_j\}_{j\in\mathcal{S}[k,i]}\right)$ in $\pi_S\left(\{x_j\}_{1\leq j\leq m_{\text{train}}}\right)$ is numerically tested by solving a nonnegative least squares (NNLS [251]) problem. If for all points $x \in \{x_j\}_{1\leq j\leq m_{\text{train}}} \setminus C_k$ the inequality

$$\min_{w\in\mathbb{R}_+^{|\mathcal{S}[k,i]|}} ||A_{\pi_S\left(\{x_j\}_{j\in\mathcal{S}[k,i]}\right)} w - \tilde{\pi}_S(x)||_2 \geq \varepsilon_{\text{DA}} ||\tilde{\pi}_S(x)||_2 \tag{11.14}$$

is satisfied with $\tilde{\pi}_S(x) = (\pi_S(x)^T\ 1)^T$ and with ε_{DA} being the tolerance of the data augmentation algorithm, then Corollary 11.4.8 implies that $\{x_j\}_{j\in\mathcal{S}[k,i]}$ is pure in $\{x_j\}_{1\leq j\leq m_{\text{train}}}$. Algorithm 3 describes the data augmentation algorithm. It calls Algorithm 2 to find n well-distributed seeds per class before growing pure sets. It is noteworthy that using few pure sets to generate many examples would increase the distribution gap [252] between augmented data and original data. To avoid this issue, one had better use many well-distributed seeds to distribute data augmentation efforts between the pure sets and thus limit the divergence between the augmented distribution and the true data-generating distribution.

Remark 11.4.9. *Realizations of the random variable X belong to a convex domain $\mathcal{X}$ related to physics constraints. When considering surface random temperature fields defined on the boundaries of a solid, $\mathcal{X}$ is a hypercube consisting of all the fields taking values between zero Kelvin degree and the material's melting point. These random fields can be used as Dirichlet boundary conditions for the nonlinear heat equation. The assumption of a linear thermal behavior is added when considering three-dimensional random temperature fields defined inside the solid, so that the set $\mathcal{X}$ remains convex when adding the constraint that the random field must satisfy the heat equation. More generally, convex combinations respect physics constraints defined by linear operators, such as linear partial differential equations and Dirichlet, Neumann, and Robin boundary conditions.*

Algorithm 2 Seeds selection for data augmentation. Note: all the dissimilarities have already been computed in the data labeling procedure.

Input: training set $\{(x_i, y_i)\}_{1\leq i\leq m_{\text{train}}}$, class label k, class center $\tilde{x}_k$, dissimilarity matrix $\boldsymbol{\delta}$, target number of seeds n, preselection parameters $(\varepsilon_1, \varepsilon_2) \in [0;1]^2$.

Output: List l_k of n indices of seeds candidates for the k-th class.

Stage 1 (filter the data):
Find the minimum dissimilarity δ^k_{ref} separating the class center $\tilde{x}_k$ from a point belonging to another class.
Remove points having neighbors belonging to foreign classes within a distance of $\varepsilon_1\delta^k_{\text{ref}}$.
Remove isolated points having no neighbor within a distance of $\varepsilon_2\delta^k_{\text{ref}}$.
$\mathcal{I}_k :=$ set of the indices of the remaining points in class k.
Stage 2 (maximin greedy selection):
Initialize l_k with the index of the class center $\tilde{x}_k$.
for $i \in [\![2; \min(n, |\mathcal{I}_k| - 1)]\!]$ **do**
    $j := \underset{l\in\mathcal{I}_k\setminus l_k}{\arg\max}\ \underset{q\in l_k}{\min}\ \delta_{lq}$
    Append j to l_k.
end for
return l_k

Algorithm 3 Data augmentation algorithm

Input: training set $\{(x_i, y_i)\}_{1 \leq i \leq m_{\text{train}}}$, dissimilarity matrix $\boldsymbol{\delta}$, per-class number of seeds n, maximum number of pure set testings $p_{\max}$, dimension d of subspaces for pure set testings, number of augmented data m_{DA}.

Output: augmented data $\{(\tilde{x}_i, \tilde{y}_i)\}_{1 \leq i \leq m_{DA}}$ and matrix $\mathcal{S}$ listing pure sets.

Stage 1 (find pure sets in the training set):
for $k \in [\![1; K]\!]$ **do**
  Apply Algorithm 2 to get the list l_k of n indices of seeds candidates.
  for $i \in [\![1; n]\!]$ **do**
    $\mathcal{S}_1 := \{l_k[i]\}$
    neighbors := argsort($\boldsymbol{\delta}[l_k[i], :]$)
    $j := 1$
    setPurity := True
    while setPurity **do**
      $\mathcal{S}_{j+1} := \mathcal{S}_j \cup \{\text{neighbors}[j]\}$
      $j := j + 1$
      $p := 1$
      Select a random subset S of d features.
      while $\{\pi_S(x_q)\}_{q \in \mathcal{S}_j}$ is not pure in $\{\pi_S(x_q)\}_{1 \leq q \leq m_{\text{train}}}$ and $p \leq p_{\max}$ **do**
        Select a new random subset S of d features.
        p := p+1
      end while
      if $p = p_{\max} + 1$ **then**
        setPurity := False
      end if
    end while
    $\mathcal{S}[k, i] := \mathcal{S}_{j-1}$
  end for
end for
Stage 2 (generate new data):
Generate m_{DA} random convex combinations $\{\tilde{x}_i\}_{1 \leq i \leq m_{DA}}$ of the pure sets listed in $\mathcal{S}$. Convex combinations $\tilde{x}_i$ of the pure set described in $\mathcal{S}[k, j]$ are labeled by $\tilde{y}_i = k$.
return $\{(\tilde{x}_i, \tilde{y}_i)\}_{1 \leq i \leq m_{DA}}$ and $\mathcal{S}$

11.4.3 Numerical results

Linear discriminant analysis (LDA), commonly used for classification tasks, can also be used for supervised dimensionality reduction by projecting the data onto the subspace maximizing the between-class variance, as explained in [55]. For the classification problem presented in this chapter, the training data are visualized in the two-dimensional subspace obtained by LDA in Figure 11.5. Although this subspace is the one that best separates the classes, one can see that the training examples do not form well-separated groups. For this reason, testing the purity of subsets of training data before generating new examples by convex combinations is necessary to reduce the risk of getting incorrectly labeled augmented data.

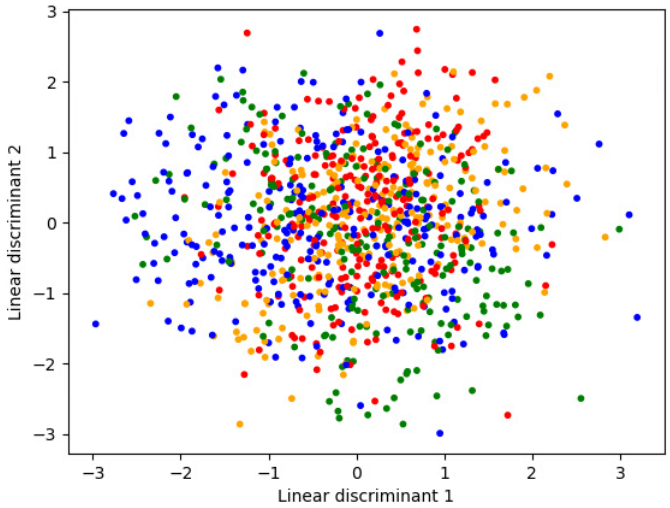

Figure 11.5: Data visualization in the 2D subspace maximizing the separation between classes (supervised linear dimensionality reduction using linear discriminant analysis (LDA)).

The data augmentation algorithm finds about 60 pure sets per class with an average population of five training examples, using random subspaces of dimension 5 to test the purity. Note that two pure sets are merged only when one is included in the other, as the union of two pure sets is not always pure. The computation time for the data augmentation training phase (stage 1 of Algorithm 3) is 40 minutes. Once pure sets have been found, one can generate as many augmented examples as necessary (stage 2 of Algorithm 3). Generating 5400 examples to multiply the size of the training set by 10 takes less than a second. Among the augmented data, 400 examples are taken for the evaluation of the data augmentation algorithm. The data labeling procedure involving numerical simulations is applied for these 400 examples in order to estimate the percentage of incorrectly labeled data. It turns out that none of these examples is incorrectly labeled, which validates the algorithm for our problem. The benefits of data augmentation for the classification task are evaluated in the next section.

11.5 Validation of our feature selection and data augmentation algorithms

11.5.1 Classification performances of various classifiers

In this section, 14 different classifiers are trained and evaluated on our classification problem. To evaluate whether the features selected by geostatistical mRMR are relevant for classification purposes, each classifier is tested twice: once in combination with the geostatistical mRMR and once with principal component analysis (PCA) with 10 modes. As the random temperature fields derive from a Gaussian random field involving only 10 modes, the database obtained after applying PCA contains all the information of the original data. Each combination of one of the 14 classifiers with PCA or feature selection is trained twice: once on the true training set containing $m_{\text{train}} = 600$ examples, and once on the augmented training set made of 6000 examples.

All the classifiers are trained with Scikit-learn [249], except multilayer perceptrons (MLPs; i.e., fully-connected feedforward deep neural networks) and radial basis function networks (RBFNs) which are trained with PyTorch [253]. We train the RBFNs in a fully supervised manner with Gaussian radial basis functions, which means that the parameters of the radial basis functions are learned by gradient descent like the weights of the fully-connected layers. In addition, we use only one RBF hidden layer followed by one fully-connected layer for RBFNs, as these artificial neural networks generally have shallow architectures, as explained in [4]. Deeper architectures have been tested for MLPs, with dropout [132], batch normalization [133], ReLU activation functions, and with the number of hidden layers ranging from 2 to 8 with a maximum of 500 neurons per layer. The architectures and the values of some hyperparameters such as the learning rate for Adam optimizer [135], batch size, number of epochs, and dropout rate are calibrated by evaluating the classifier on the validation set after each training attempt. Scikit-learn's MLP classifier has also been tested; it is called *simple MLP* in this thesis, because its architecture is only made of fully-connected layers and does not include dropout nor batch normalization. All the classifiers based on artificial neural networks are trained with Tikhonov regularization (or L^2 regularization of the network's parameters) and early stopping [3]. Logistic regression is trained with elastic net regularization [105] consisting in a weighted average of L^1 and L^2 penalties of the model's parameters. Kernels used for support vector machines (SVMs) are obtained by combining several polynomial kernels with different hyperparameters. Kernel design could be optimized using multiple kernel learning algorithms [254], but we simply build our kernels by evaluating different combinations on the validation set, just as when we look for a good architecture for artificial neural networks. For all of the classifiers using regularization terms in their loss functions, namely, neural networks, SVMs, and logistic regression, hyperparameters such as the regularization strength (weight of the regularization term in the loss function) or the elastic net mixing coefficient are also calibrated using the validation set. For tree-based classifiers, model capacity is controlled by adjusting the maximum depth of the tree and the minimum number of samples at a leaf node. Given the instability of decision trees and their known tendency to overfit, our analysis includes random forests, as well as AdaBoost and gradient boosting with decision trees as base estimators, whose hyperparameters are calibrated on the validation set. Finally, this comparative study also includes the k-nearest neighbors classifier whose number of nearest neighbors must be calibrated, and three generative classifiers that have (almost) no hyperparameter to calibrate, namely, Gaussian naive Bayes, LDA and QDA classifiers.

The classification accuracies on test data are given in Table 11.2 for classifiers trained with PCA and in Table 11.3 for those trained with feature selection. Of course, this ranking is specific to the classification problem presented in this chapter, no general conclusion can be drawn from this particular numerical application. On this classification problem, when using augmented data in the training phase, the highest test accuracy reached with linear classifiers is 43.5%, obtained with the linear SVM combined with PCA. The fact that k-nearest neighbors classifiers barely exceed 50.0% of accuracy on this problem is related to an observation that was made in [47]: there is no simple correlation between the Euclidean distance and the physics-informed dissimilarity measure used in dictionary-based ROM-nets. MLPs get the best results, reaching 87.0% of accuracy when combined with our data augmentation and feature selection algorithms. Interestingly, quadratic discriminant analysis (QDA) gives excellent results while having no hyperparameter to tune, contrary to the two other families of classifiers obtaining the best results: MLPs and multiple kernel SVMs. This makes QDA the best compromise between accuracy and training complexity

for this specific classification task.

Although PCA perfectly describes the input data in this example, the geostatistical mRMR feature selection algorithm enables reaching higher accuracies with some classifiers. Not only does it behave as the original mRMR when selecting features, but it also gives satisfying results when combined with a classifier. Concerning data augmentation, Tables 11.2 and 11.3 show that our algorithm significantly improves classification results. The accuracy gain due to data augmentation is 4.98% on average and ranges from −2.5% to 10.5%, increasing the accuracy in 25 cases out of 28.

Table 11.2: Test accuracies of different classifiers with dimensionality reduction by principal component analysis (PCA), with and without data augmentation (DA).

Classifier	Acc. with DA	Acc. without DA
Multilayer perceptron	86.5%	81.5%
Simple multilayer perceptron	85.0%	79.5%
Quadratic discriminant analysis	76.0%	70.0%
Multiple kernel support vector machine	73.0%	68.0%
Radial basis function network	63.5%	62.0%
k-nearest neighbors	51.0%	46.0%
AdaBoost	50.5%	52.5%
Gradient-boosted trees	49.5%	48.0%
Random forest	45.0%	47.5%
Linear support vector machine	43.5%	33.0%
Gaussian naive Bayes	38.5%	31.5%
Penalized logistic regression	38.5%	28.0%
Decision tree	34.0%	36.5%
Linear discriminant analysis	33.5%	29.0%

Table 11.3: Test accuracies of different classifiers with dimensionality reduction by feature selection (FS), with and without data augmentation (DA).

Classifier	Acc. with DA	Acc. without DA
Multilayer perceptron	87.0%	81.0%
Simple multilayer perceptron	84.0%	80.0%
Quadratic discriminant analysis	77.5%	70.5%
Multiple kernel support vector machine	72.5%	66.0%
Random forest	69.0%	63.0%
AdaBoost	68.5%	63.0%
Gradient-boosted trees	68.0%	58.5%
Radial basis function network	62.5%	60.0%
Decision tree	55.5%	43.5%
k-nearest neighbors	50.0%	47.0%
Linear support vector machine	40.5%	34.5%
Gaussian naive Bayes	39.5%	34.5%
Penalized logistic regression	37.0%	29.0%
Linear discriminant analysis	32.5%	29.0%

11.5.2 Comparison with a CNN

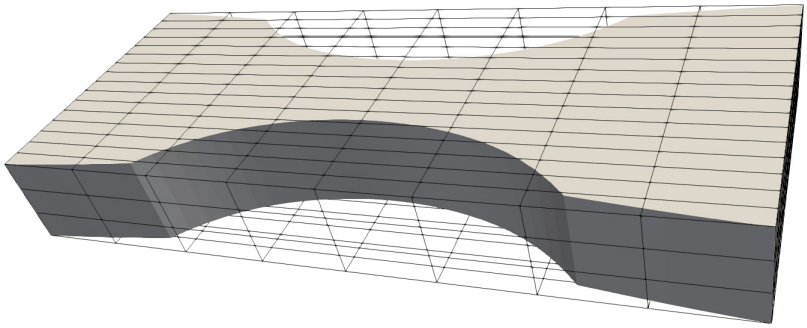

Figure 11.6: Voxel grid used to get a 3D image of a temperature field defined on a finite-element mesh.

Convolutional neural networks (CNNs) are commonly used for image classification. Our classification problem shares some similarities with image classification, since each input feature is attached to a node in the mesh, just as each feature of an image is attached to a pixel. Instead of using a feature selection method or any dimensionality reduction method, one can project 3D temperature fields onto a voxel grid to get 3D images of these fields that can be fed into a CNN with 3D convolution filters. The voxel grid used in this work is illustrated in Figure 11.6. It contains $16 \times 8 \times 3$ voxels. The voxels being outside the solid body Ω are assigned a zero value, while for those being inside Ω, the local value of the temperature field is evaluated with the finite-element shape functions. This preprocessing step has some drawbacks compared to feature selection, namely:

- Projection onto the voxel grid takes about 5 seconds per field on this rather small mesh, whereas extracting the nodal values identified by feature selection is instantaneous;
- Some voxels may carry useless information, since some of them are located outside of the solid body Ω;
- There might be too many voxels in irrelevant areas of Ω, and conversely, there might be too few voxels in relevant areas.

Nonetheless, feature selection does not enable using a CNN. To go further, one could replace the traditional CNN with *geometric deep learning* methods [255] applying convolution filters on point clouds, meshes and graphs, which avoids projecting the data onto a voxel grid. However, this solution does not tackle the difficulties related to the high-dimensionality of input data.

The CNNs used in this study are trained with PyTorch [253] with Adam optimizer [135]. The architecture giving the best performances on this classification problem is made of 3 blocks containing a convolution layer, batch normalization [133] and ReLU activation function, followed by a Max-Pooling layer, one fully-connected layer with ReLU activation and dropout [132], and a final fully-connected layer with softmax activation function. The architecture and the values of some hyperparameters such as the number of epochs, the

batch size, the learning rate, the L^2 regularization strength, the dropout rate, the number of channels per convolution layer, the filters' sizes, and padding have been optimized manually going back and forths between the training set and the validation set. On test data, the CNN reaches an accuracy of 74.5% with data augmentation, and 74.0% without data augmentation. The CNN has not been trained on voxel grids with other resolutions.

11.5.3 How to further improve classification performances?

Table 11.4: Test accuracies obtained by ensemble methods with dimensionality reduction by principal component analysis (PCA).

Classifier	Acc. with DA
Stacking (6 MLPs and logistic regression)	89.5%
Ensemble averaging (6 MLPs)	89.0%

Table 11.5: Test accuracies obtained by ensemble methods with dimensionality reduction by feature selection (FS).

Classifier	Acc. with DA
Stacking (6 MLPs and logistic regression)	90.0%
Ensemble averaging (6 MLPs)	89.0%

Ensemble methods can be used to reduce overfitting and increase the accuracy on test data. In addition, it enables recycling different variants of a classifier that the user has trained for different hyperparameters. Using ensemble averaging with classifiers trained on the augmented dataset with feature selection, we manage to combine six MLPs with different architectures to reach an accuracy of 89.0%. When stacking these MLPs with a ridge logistic regression analyzing the predicted membership probabilities, we get an accuracy of 90.0%. Following the same procedures with six MLPs trained on the PCA representation of the data, ensemble averaging (resp. stacking with ridge logistic regression) gives an accuracy of 89.0% (resp. 89.5%). These results are summarized in Table 11.4 and Table 11.5. In addition to ensemble learning methods, one can also use random noise injection to increase noise robustness, as explained in [3].

11.6 Applicability to other problems

The feature selection and data augmentation algorithms introduced in this thesis can either be used together like in the previous section, or they can be used separately and combined with other algorithms. For instance, the data augmentation algorithm can be used in conjunction with any dimensionality reduction technique. However, feature selection is recommended when the interpretability of input data is important. In addition, unsupervised dimensionality reduction techniques such as PCA, sparse PCA, kernel PCA, and deep autoencoders extract features that are relevant to reconstruct data after compression, while supervised dimensionality reduction techniques such as mRMR and geostatistical mRMR select features that are suitable for the specific supervised learning task that is considered.

Although this work is motivated by difficulties encountered when training dictionary-based ROM-nets, our algorithms could be used for other computational methods involving classifiers for model recommendation, such as the LDEIM [42]. The nature of the models in the dictionary, the underlying physics describing the problem, and the way input data are labeled do not matter. It is important to emphasize that our data augmentation algorithm is dedicated to classification problems, whereas our feature selection algorithm could be applied to regression problems too as the definition of the relevance $D(S, Y)$ can be extended to continuous output variables Y. As a consequence, the geostatistical mRMR feature selection algorithm can be applied before training a metamodel that directly predicts a quantity of interest, as long as the inputs are regionalized variables. One could also think of a classifier making qualitative predictions, such as a binary classifier saying whether the system will fail for a given configuration. Our algorithms are applicable to all these types of problems on physics data, and more generally on continuous fields[3] for the feature selection algorithm and on data with linear constraints for the data augmentation algorithm (with a possible extension to nonlinear constraints, as explained in Remark 11.4.9).

* *
*

[3]More precisely: random fields whose realizations are continuous with probability one, see Definition 2.3.8 for sample path continuity.

Part V

Application to an industrial problem

Résumé

Cette partie clôt ce mémoire de thèse avec l'application d'un ROM-net à un problème industriel réel. Il s'agit de quantifier les incertitudes sur le comportement mécanique d'une aube de turbine haute pression dans un turboréacteur, étant données des incertitudes sur le chargement thermique. Située juste après la chambre de combustion, la turbine haute pression est un des organes critiques d'un turboréacteur. En effet, les aubes de turbine haute pression sont soumises à un chargement thermo-mécanique extrême résultant de la combinaison des efforts centrifuges engendrés par le rotor et de l'écoulement turbulent des gaz chauds sortants de la chambre de combustion.

Connaître avec précision le chargement thermique appliqué à une aube de turbine haute pression est à ce jour impossible. Bien que nous disposions de modèles sophistiqués pour simuler l'écoulement turbulent issu de la chambre de combustion et l'évolution du champ de température au sein d'une aube, les prédictions numériques sont difficilement validables à l'aide d'essais en raison du manque d'instruments de mesure fiables permettant d'enregistrer l'évolution temporelle exacte de la température en chaque point de l'aube. Les peintures thermo-sensibles sont utilisées pour mesurer la température maximale atteinte sur la surface externe de l'aube. Cependant, ces peintures ne sont précises qu'à 50 degrés près et ne reproduisent pas un champ de température surfacique réellement observé à un instant donné, puisque la température mesurée en chaque point correspond à la température maximale atteinte au cours d'un cycle de chargement représentatif d'un vol. À haute température, les propriétés du superalliage base nickel constituant l'aube dépendent fortement de la température. Par conséquent, une incertitude de 50 degrés sur la température locale se traduit par de grandes incertitudes sur les prédictions numériques du comportement mécanique. Simuler avec précision le comportement mécanique d'une aube et son endommagement progressif est essentiel dans l'optique de la réduction de l'empreinte écologique de l'industrie aéronautique, car l'amélioration du rendement d'un turboréacteur passe par l'augmentation de la température de sortie de chambre de combustion, ce qui exacerbe le chargement thermo-mécanique sévère appliqué aux aubes de turbine haute pression.

Pour s'assurer de la fiabilité d'une aube, il est donc nécessaire de prendre en compte ces incertitudes en phase de conception afin de contrôler les risques et d'optimiser le design de l'aube et les marges prévues. Lancer un grand nombre de simulations mécaniques pour différents chargements thermiques possibles n'est pas envisageable, car une seule simulation peut prendre plusieurs semaines en raison de la taille du maillage éléments finis, de la complexité de la loi de comportement, et du nombre de cycles de chargement à simuler avant convergence de la réponse de l'aube. La réduction d'ordre de modèle permet d'accélérer ces simulations en calculant une solution approchée du problème mécanique, sous réserve que les incertitudes sur la thermique restent raisonnables. Pour aller plus loin

dans la prise en compte de fortes incertitudes tout en maintenant des temps de calculs raisonnables, un ROM-net peut être utilisé afin d'adapter le choix du modèle réduit en fonction du scénario de chargement thermique considéré.

Dans cette partie, nous détaillons chacune des étapes de construction d'un ROM-net pour l'application à l'aube de turbine. Le ROM-net est ensuite utilisé pour propager les incertitudes sur les quantités d'intérêt du problème, permettant d'avoir une vision statistique de l'état de santé de l'aube. Les prédictions du ROM-net sont comparées à celles du modèle haute-fidélité plus coûteux pour validation. Dans cet exemple, le ROM-net accélère les calculs d'un facteur 636, tout en maintenant un niveau d'erreur de l'ordre de 1% à 3% sur les quantités d'intérêt, selon l'indicateur d'erreur choisi.

* *
*

Chapter 12

Industrial context

Abstract: *This chapter presents an industrial test case for the evaluation of the methodologies developed in this thesis. It consists in predicting the mechanical behavior of a high-pressure (HP) turbine blade in an aircraft engine with uncertainties on the thermal loading. The industrial context and the models for the mechanical behavior and the thermal loading are presented, with a particular emphasis on the assumptions that have been made.*

Remark 12.0.1. *In the figures given in the coming chapters, values on the axes have been voluntarily removed for confidentiality reasons.*

Remark 12.0.2. *This chapter is taken from our paper [50], with some modifications.*

Contents

12.1 HP turbine blades in an aircraft engine

12.1.1 Thermomechanical fatigue of HP turbine blades

High-pressure (HP) turbine blades are critical parts in an aircraft engine. Located downstream of the combustion chamber (see Figure 12.1), they are subjected to extreme thermomechanical loadings resulting from the combination of centrifugal forces, pressure loads, and hot turbulent fluid flows whose temperatures are higher than the material's melting point. The thermomechanical loading repeated over time progressively damages the blades and leads to crack initiation under thermomechanical fatigue. Predicting the fatigue lifetime is crucial not only for safety reasons, but also for ecological issues, since reducing fuel consumption and improving the engine's efficiency requires increasing the temperature of the gases leaving the combustion chamber.

Figure 12.1: The LEAP, turbofan developed by CFM International, a joint venture between Safran Aircraft Engines and GE Aviation. This engine powers Airbus A320neo, Boeing 737 MAX and COMAC C919 planes. Picture taken from https://medialibrary.safran-group.com/Photos/media/178745. ©2017 Antonio Gomez, Safran.

High-pressure turbine blades are made of monocrystalline nickel-based superalloys that have good mechanical properties at high temperatures. To reduce the temperature inside this material, the blades contain cooling channels in which fresh air circulates. In addition, the blade's outer surface is protected by a thin thermal barrier coating. In spite of these advanced cooling technologies, the rotor blades undergo centrifugal forces at high temperatures, causing inelastic strains. Under this cyclic thermomechanical loading repeated over the flights, the structure has a viscoplastic behavior and reaches a viscoplastic stabilized response, where the dissipated energy per cycle still has a nonzero value. This is called *plastic shakedown*, and leads to *low-cycle fatigue*. At cruise flight, the persistent

centrifugal force applied at high temperature induces progressive (or time-dependent) inelastic deformations: this phenomenon is called *creep*. In addition, the difference between gas pressures on the extrados and the intrados of the blade generates bending effects. Environmental factors may also locally modify the chemical composition of the material, leading to its *oxidation*. As oxidized parts are more brittle, they facilitate crack initiation and growth. *Thermal fatigue* resulting from temperature gradients is another life-limiting factor. Temperature gradients make cold parts of the structure prevent the thermal expansion of hot parts, creating compressive thermal stresses in these hot parts. Due to their higher temperatures, the hot parts are more viscous and have a lower yield stress, which make them prone to develop inelastic strains in compression. When the temperature cools down after landing, tensile *residual stresses* appear in parts which were compressed at high temperatures and favour crack nucleation. Given the complex temperature field resulting from the internal cooling channels and the turbulent gas flow, thermal fatigue has a strong influence on the turbine blade's lifetime. In particular, during transient regimes such as take-off, an important temperature gradient appears between the leading edge and the trailing edge of the blade, since the latter has a low thermal inertia due to its small thickness and thus warms up faster.

In short, the behavior of a high-pressure turbine blade results from a complex interaction between low-cycle fatigue, thermal fatigue, creep, and oxidation. Due to the cost and the complexity of experiments on parts of an aircraft engine, numerical simulations play a major role in the design of high-pressure turbine blades and their fatigue lifetime assessment.

12.1.2 Industrial test case and objectives

Industrial problem

Figure 12.2 gives the simplified geometry and the finite-element mesh of a real high-pressure turbine blade. The mesh is made of quadratic tetrahedral elements, and contains approximately 10^6 nodes. The elasto-viscoplastic mechanical behavior is described by a crystal plasticity model presented in Section 12.2.3. Computing the fatigue lifetime of the HP turbine blade requires simulating its behavior until the stabilization of the mechanical response, which may last several weeks because of the size of the mesh, the complexity of the constitutive equations, and the number of loading cycles in the transient regime. With such a computation time, uncertainty quantification with Monte Carlo simulations is unaffordable. In addition, such simulations are too time-consuming to be integrated in design iterations, which limits them to the final validation and certification steps, while the design process still relies on simplified models. Accelerating these complex simulations is a key challenge, as it would provide useful numerical tools to improve design processes and quantify the effect of the uncertainties on the environment of the system. With the help of domain decomposition methods, the computation time can be reduced by solving equilibrium equations in parallel on different subdomains of the geometry. Using the implementation of the Adaptive MultiPreconditioned FETI solver [256] in Zset finite-element software [79], the simulation of one single loading cycle of the HP turbine blade with 48 subdomains takes approximately 53 minutes. Domain decomposition methods have the advantage of computing the exact high-fidelity solution in a reasonable computation time thanks to advances in high-performance computing. Model order reduction methods, and in particular ROM-nets, are complementary to domain decomposition methods, as they

compute approximate solutions much faster. Model order reduction is more suitable for uncertainty propagation purposes where thousands of simulations must be run, but domain decomposition methods remain important to reduce the cost of the training phase as much as possible.

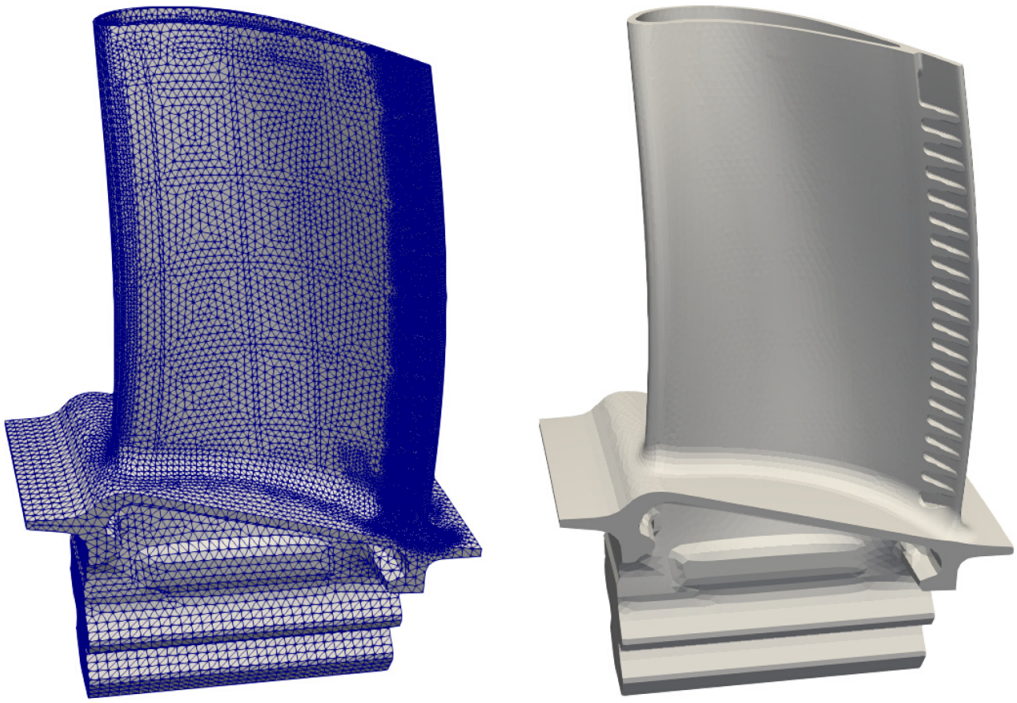

Figure 12.2: High-pressure turbine blade geometry and mesh (micro-perforations are not modeled).

Objectives

The objective is to use a ROM-net to quantify uncertainties on the mechanical behavior of the high-pressure turbine blade, given uncertainties on the thermal loading. The reduction of the computation time should enable Monte Carlo simulations for uncertainty quantification. This work is a proof of concept: it aims at evaluating the performances of a ROM-net on a real industrial test case, but not at giving real conclusions on the uncertainties in the design of high-pressure turbine blades. In other words, the objective of this study is to evaluate and validate a methodology on an industrial test case, rather than using realistic modeling assumptions and data to derive relevant information on the design of high-pressure turbine blades.

In particular, we are not interested in predicting the state of the structure after a large number of flight-representative loading cycles. Only one cycle is simulated. Cyclic extrapolation of the behavior of a high-pressure turbine blade has been studied in [65, 35] and is out of the scope of the present work.

12.2 Model and assumptions

12.2.1 Modeling assumptions

Weak thermomechanical coupling

It is assumed that the heat produced or dissipated by mechanical phenomena has negligible effects in comparison with thermal conduction, which enables avoiding strongly coupled thermomechanical simulations and running thermal and mechanical simulations separately instead. Under a weak thermomechanical coupling, the first step consists in solving the heat equation to determine the temperature field and its evolution over time. The temperature field history defines the thermal loading and is used to compute thermal strains and temperature-dependent material parameters for the mechanical constitutive laws. Once the thermal loading is known, the temperature-dependent mechanical problem must be solved in order to predict the mechanical response of the structure.

Cyclic thermomechanical loading

The thermomechanical loading applied to the high-pressure turbine blade during its whole life is modeled as a cyclic loading, with one cycle being equivalent to one flight. The rotation speed of the turbine's rotor is proportional to a periodic function of time $\omega(t)$ whose evolution over one period (or cycle, see Figure 12.3) is representative of one flight with its three main regimes, namely take-off, cruise, and landing. The period (or duration of one cycle) is denoted by t_c. The rotation speed between flights k and $k+1$ is zero, which means that $\omega(kt_c) = 0$ for any integer k. The rotation speed $\omega(t)$ is scaled so that its maximum is 1.

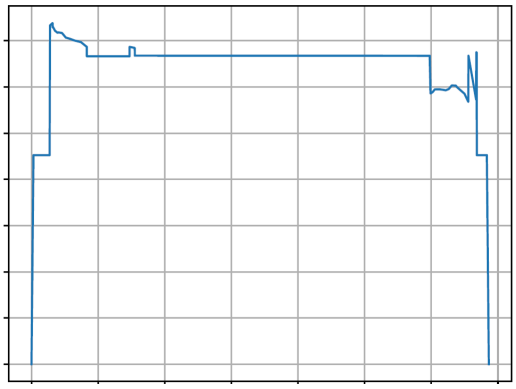

Figure 12.3: Function $\omega(t)$ defining one cycle for the rotation speed.

Let $\Omega \subset \mathbb{R}^3$ denote the solid body representing the high-pressure turbine blade, with $\partial\Omega$ denoting its outer surface. Let $\partial\Omega^p \subset \partial\Omega$ be the surface corresponding to the intrados and extrados. The thermal loading is defined as:

$$\forall \boldsymbol{\xi} \in \Omega, \quad \forall t \in \mathbb{R}_+, \qquad T(\boldsymbol{\xi}, t) = (1 - \omega(t))T_0 + \omega(t)T_{\max}(\boldsymbol{\xi}) \tag{12.1}$$

where $T_0 = 293\ K$ and $T_{\max}$ is the temperature field obtained when the rotation speed reaches its maximum. This field $T_{\max}$ is obtained either by an aerothermal simulation

or by a stochastic model, as explained later. Similarly, the pressure load applied on $\partial\Omega^p$ reads:

$$\forall \boldsymbol{\xi} \in \partial\Omega^p, \quad \forall t \in \mathbb{R}_+, \qquad p^{\partial\Omega}(\boldsymbol{\xi}, t) = (1 - \omega(t))p_0^{\partial\Omega} + \omega(t)p_{\max}^{\partial\Omega}(\boldsymbol{\xi}) \tag{12.2}$$

where $p_0^{\partial\Omega} = 1$ atm is the atmospheric pressure at sea level, and where $p_{\max}^{\partial\Omega}$ is the pressure field obtained when the rotation speed reaches its maximum. The clamping of the blade's fir-tree foot on the rotor disk is modeled by displacements boundary conditions that are not detailed here.

Geometric details and TBC

Small geometric details of the structure have been removed to simplify the geometry. Nonetheless, the main cooling channels are considered. The effects of the thermal barrier coating (TBC) have been integrated in aerothermal simulations, but the TBC is not considered in the mechanical simulation although its damage locally increases the temperature in the nickel-based superalloy and thus affects the fatigue resistance of the structure. Additional centrifugal effects due to the TBC are not taken into account.

Influential factors

The predicted mechanical response of the structure depends on many different factors. Below is a non-exhaustive list of influential factors that are possible sources of uncertainties in the numerical simulation:

- **Thermal loading:** The viscoplastic behavior of the nickel-based superalloy is very sensitive to the temperature field and its gradients. However, the temperature field is not accurately known because of the impossibility of validating numerical predictions experimentally. Indeed, temperature-sensitive paints are accurate to within 50 K only, and they do not capture a real surface temperature field since they measure the maximum temperature reached locally during the experiment.
- **Crystal orientation:** Because of the complexity of the manufacturing process of monocrystalline blades, the orientation of the crystal is not perfectly controlled. As the superalloy has anisotropic mechanical properties, defaults in crystal orientation highly affect the location of damaged zones in the structure.
- **Mechanical loading:** The centrifugal forces are well known because they are related to the rotation speed that is easy to measure. On the contrary, pressure loads are uncertain because of the turbulent nature of the incoming fluid flow. However, the effects of pressure loads uncertainties on the mechanical response are less significant than those of the thermal loading and crystal orientation uncertainties.
- **Constitutive laws:** Uncertainties on the choice of the constitutive model, the relevance of the modeling assumptions, and the values of the calibrated parameters involved in the constitutive equations also influence the results of the numerical simulations.

For simplification purposes, the only source of uncertainty that is considered in this work is the thermal loading. The equations of the mechanical problem are then seen as

parametrized equations, where the parameter is the temperature field $T_{\max}$ (see Equation (12.1)) obtained when the rotation speed reaches its maximum value. The dimension of the parameter space is then the number of nodes in the finite-element mesh. The mechanical loading is assumed to be deterministic. With the crystal orientation, the constitutive laws and their parameters (or coefficients), they are considered as known data describing the context of the study and given by experts from Safran.

12.2.2 Stochastic model for the thermal loading

A stochastic model is required to take into account the uncertainties on the thermal loading. Given the definition of the thermal loading in Equation (12.1), we only need to model uncertainties in space through the field $T_{\max}$ obtained when the rotation speed reaches its maximum value. The random temperature fields must satisfy some constraints: they must satisfy the heat equation, and they must not take values out of the interval $[0\ K; T_{\text{melt}}]$, where T_{melt} is the melting point of the superalloy. These random fields are obtained by adding random fluctuations to a reference temperature field, see Figure 12.4. The reference field was computed by engineers from Safran, using the software *Ansys Fluent*[1] for aerothermal simulations. The data-generating distribution is defined as a Gaussian mixture model made of two Gaussian distributions with the same covariance function but with distinct means, and with a prior probability of 0.5 for each Gaussian distribution. The Gaussian distributions are obtained by taking the four first eigenfunctions of the covariance function (see Karhunen-Loève expansion [247]), with a standard deviation of 15 K. Therefore, realizations of the random temperature field read:

$$\forall \boldsymbol{\xi} \in \Omega, \quad T(\boldsymbol{\xi}) = T_{\text{ref}}(\boldsymbol{\xi}) + \Upsilon_0\ \delta T_0(\boldsymbol{\xi}) + \sum_{i=1}^{4} \Upsilon_i\ \delta T_i(\boldsymbol{\xi}) \tag{12.3}$$

where T_{ref} is the reference field, δT_0 is a temperature perturbation at the trailing edge whose maximum value is 50 K, $\{\delta T_i\}_{1\leq i\leq 4}$ are fluctuation modes, Υ_0 is a random variable following the Bernoulli distribution with parameter 0.5, and $\{\Upsilon_i\}_{1\leq i\leq 4}$ are independent and identically distributed random variables following the standard normal distribution $\mathcal{N}(0,1)$. The variable Υ_0 is also independent of the other variables Υ_i. The different fields involved in Equation (12.3) can be visualized in Figure 12.4. Equation (12.3) defines a mixture distribution with two Gaussian distributions whose means are T_{ref} and $T_{\text{ref}} + \delta T_0$. We voluntarily define this mixture distribution with δT_0 adding 50 K in a critical zone of the turbine blade in order to check that our physics-informed cluster analysis can successfully detect two relevant clusters, *i.e.* one for fields obtained with $\Upsilon_0(\theta) = 0$ and one for fields obtained with $\Upsilon_0(\theta) = 1$. Indeed, the temperature perturbation δT_0 is expected to significantly modify the mechanical response of the high-pressure turbine blade. All the fields $\{\delta T_i\}_{0\leq i\leq 4}$ satisfy the steady heat equation like T_{ref}, which ensures that the random fields always satisfy the heat equation under the assumption of a linear thermal behavior. For nonlinear thermal behaviors, Equation (12.3) would define surface temperature fields that would be used as Dirichlet boundary conditions for the computation of bulk temperature fields. The assumption of a linear thermal behavior is adopted here to avoid solving the heat equation for every realization of the random temperature field.

Let us now give more details about the construction of the fluctuation modes $\{\delta T_i\}_{1\leq i\leq 4}$. First, surface fluctuation modes are computed on the boundary $\partial\Omega$ using the method given

[1] https://www.ansys.com/products/fluids/ansys-fluent

in [257] for the construction of random fields on a curved surface. The correlation function is defined as a function of the geodesic distance d_G along the surface $\partial\Omega$:

$$\rho(\boldsymbol{\xi}, \boldsymbol{\xi}') = \exp\left(-\frac{d_G(\boldsymbol{\xi}, \boldsymbol{\xi}')}{d_G^0}\right) \tag{12.4}$$

where d_G^0 is a correlation length. Geodesic distances are computed thanks to the algorithm described in [258, 259] and implemented in the Python library *gdist*[2]. A covariance matrix is built by evaluating the correlation function on pairs of nodes of the outer surface of the finite-element mesh, and multiplying the correlation by the constant variance. The four surface modes are then obtained by finding the four eigenvectors corresponding to the largest eigenvalues of the covariance matrix. The steady heat equation with Dirichlet boundary conditions is solved for each of these surface modes to derive the 3D fluctuation modes, using *Zset* [79] finite-element solver. The Python library *BasicTools*[3] developed by Safran is used to read the finite-element mesh and write the temperature fields in a format that can be used for simulations on *Zset*.

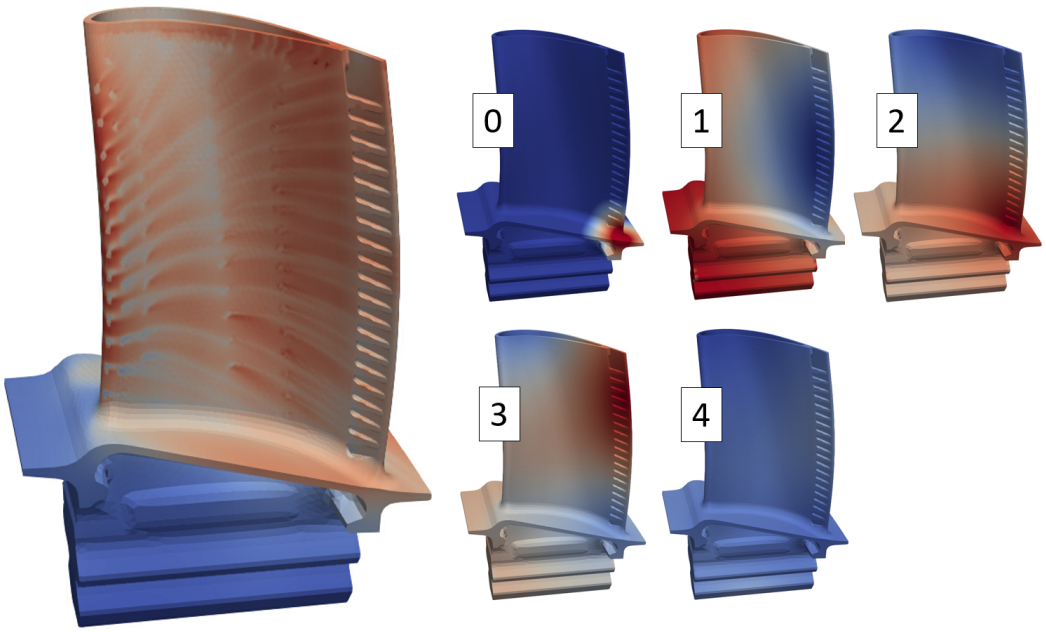

Figure 12.4: Reference temperature field (on the left), temperature perturbation at the trailing edge (field 0), and fluctuation modes (fields 1 to 4). The fluctuations in the fourth mode are located inside the blade, in the cooling channels.

12.2.3 Mechanical constitutive model

It is assumed that the mechanical behavior of the high-pressure turbine blade can be described in the framework of the infinitesimal strain theory. The mechanical response of the structure during the first loading cycle is described by the following equilibrium

[2] https://pypi.org/project/gdist/
[3] https://gitlab.com/drti/basic-tools

equations and boundary conditions:

$$\left\{\begin{array}{rcll} \mathbf{div}(\boldsymbol{\sigma}(\boldsymbol{\xi},t)) + f_C(\boldsymbol{\xi},t) &=& \mathbf{0} & \forall t \in [0;t_c] \quad \forall \boldsymbol{\xi} \in \Omega \\ \boldsymbol{\sigma}(\boldsymbol{\xi},t).\mathbf{n}(\boldsymbol{\xi},t) &=& -p^{\partial\Omega}(\boldsymbol{\xi},t)\mathbf{n}(\boldsymbol{\xi},t) & \forall t \in [0;t_c] \quad \forall \boldsymbol{\xi} \in \partial\Omega^p \\ \mathbf{u}(\boldsymbol{\xi},t) &=& \mathbf{u}^{\partial\Omega}(\boldsymbol{\xi},t) & \forall t \in [0;t_c] \quad \forall \boldsymbol{\xi} \in \partial\Omega \setminus \partial\Omega^p \end{array}\right. \tag{12.5}$$

where $\mathbf{u}(\boldsymbol{\xi},t)$ is the displacement field (primal variable), $\boldsymbol{\sigma}(\boldsymbol{\xi},t)$ is the symmetric second-order Cauchy stress tensor, $f_C(\boldsymbol{\xi},t)$ is the local volumic centrifugal force, $\mathbf{u}^{\partial\Omega}(\boldsymbol{\xi},t)$ is the imposed displacement, and $\mathbf{n}(\boldsymbol{\xi},t)$ is the outward-pointing normal vector to the outer surface $\partial\Omega$. The relation between the stress tensor and the displacement field is described by constitutive laws modeling the mechanical behavior of the monocrystalline nickel-based superalloy. At high temperatures, this material has an elasto-viscoplastic behavior that can be described in the crystal plasticity framework [260, 261] to model inelastic strains generated by the motion of dislocations[4] in different slip systems of the crystal. The strain tensor ε is defined as the symmetric part of the displacement gradient (with respect to $\boldsymbol{\xi}$):

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\boldsymbol{\nabla}\mathbf{u} + (\boldsymbol{\nabla}\mathbf{u})^T\right) \tag{12.6}$$

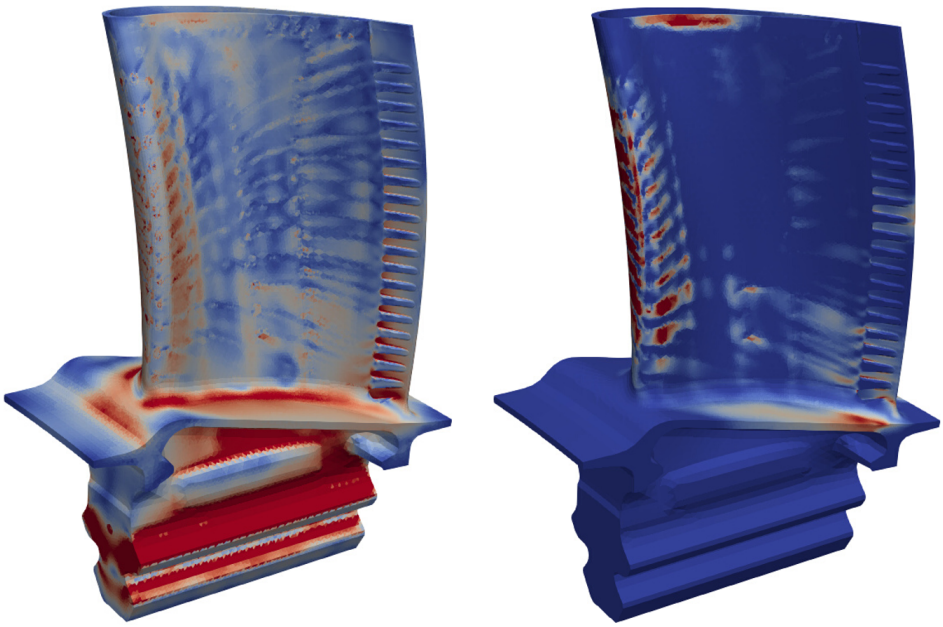

Figure 12.5: On the left: von Mises stress field σ_{eq} obtained when the rotation speed reaches its maximum value. On the right: accumulated plastic strain p^o_{cum} in octahedral slip systems at the end of the first cycle. Note: the foot of the high-pressure turbine blade has an elastic behavior, while the rest of the blade has a viscoplastic behavior described by a crystal plasticity model.

The stress tensor is obtained from the elastic strain tensor thanks to Hooke's law:

$$\boldsymbol{\sigma} = \mathbf{C} : (\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^p - \alpha(T - T_0)\mathbf{1}) \tag{12.7}$$

where $\boldsymbol{\varepsilon}^p$ is the tensor of inelastic strains and $\mathbf{1}$ is the identity second-order tensor. The fourth-order tensor $\mathbf{C}$ is the stiffness tensor. Given the face-centered cubic crystal structure of the superalloy, the stiffness tensor is anisotropic but has only three independent

[4] Linear defects in the crystal structure.

coefficients. The thermal expansion of crystals with cubic symmetry is isotropic, which explains why the thermal expansion coefficient α is the same in all directions. The time evolution of hidden variables such as inelastic strains are described by ordinary differential equations that must be solved at every integration point of the finite-element mesh. The inelastic strain rate can be decomposed into contributions of dislocations motions in 12 octahedral slip systems and 6 cubic slip systems:

$$\dot{\boldsymbol{\varepsilon}}^p = \sum_{s=1}^{12} \dot{\gamma}_s^o \ \mathrm{sym}\left(\mathbf{l}_s^o \otimes \mathbf{n}_s^o\right) + \sum_{s=1}^{6} \dot{\gamma}_s^c \ \mathrm{sym}\left(\mathbf{l}_s^c \otimes \mathbf{n}_s^c\right) = \sum_{s=1}^{12} \dot{\gamma}_s^o \mathbf{m}_s^o + \sum_{s=1}^{6} \dot{\gamma}_s^c \mathbf{m}_s^c \tag{12.8}$$

where $\dot{\gamma}_s^o$ (*resp.* $\dot{\gamma}_s^c$) is the shear strain rate in the s-th octahedral (*resp.* cubic) slip system. The tensor $\mathbf{m}_s^o$ (*resp.* $\mathbf{m}_s^c$) is the orientation tensor of the s-th octahedral (*resp.* cubic) slip system, defined by the normal $\mathbf{n}_s^o$ (*resp.* $\mathbf{n}_s^c$) to the slip plane and the slip direction $\mathbf{l}_s^o$ (*resp.* $\mathbf{l}_s^c$). The shear strain rates $\dot{\gamma}_s^o$ are given by a hyperbolic viscoplastic flow rule:

$$\dot{\gamma}_s^o = \varepsilon_h^o \ \sinh\left(\left\langle \frac{|\tau_s^o - x_s^o| - r_s^o}{K_h^o} \right\rangle^{n_h^o}\right) \ \mathrm{sign}(\tau_s^o - x_s^o) \tag{12.9}$$

where ε_h^o, K_h^o and n_h^o are material parameters. Similar equations are satisfied in cubic slip systems. The resolved shear stresses τ_s^o are given by Schmid's law:

$$\tau_s^o = \boldsymbol{\sigma} : \mathbf{m}_s^o \tag{12.10}$$

Again, similar equations are valid for cubic slip systems. The stress variables x_s^o, x_s^c, r_s^o and r_s^c describe hardening phenoma, *i.e.* the evolution of the shape of the elastic domain within which no dissipative phenoma occur. The back-stresses x_s^o (and x_s^c) are the solutions of an ordinary differential equation modeling kinematic hardening with static recovery:

$$\dot{x}_s^o = c^o \dot{\gamma}_s^o - d^o x_s^o |\dot{\gamma}_s^o| - c^o \left(\frac{|x_s^o|}{M^o}\right)^{m^o} \tag{12.11}$$

Isotropic hardening is modeled by the following equations:

$$r_s^o = r_0^o + Q^o \left(1 - \exp\left(-b^o \nu_s^o\right)\right) \tag{12.12}$$

with $\dot{\nu}_s^o = |\dot{\gamma}_s^o|$. All the constitutive equations given in this section are true for all $\boldsymbol{\xi} \in \Omega$ and for all $t \in [0; t_c]$, and are solved at every integration point of the finite-element mesh. All the coefficients involved in these equations depend on the local value of the temperature field. The problem is thus seen as a system of partial differential equations and ordinary differential equations parametrized by the thermal loading. The standard procedure for the computation of a fatigue lifetime with an uncoupled damage model consists in solving the mechanical problem for a large number of cycles until the stabilization of the mechanical response (plastic shakedown). Then, a damage field can be computed in a post-processing step and can be linked to a fatigue lifetime. For high-pressure turbine blades, fatigue models generally consider interaction effects with oxidation and creep, like in [262, 263]. In this work, no fatigue lifetime is computed since we only solve the problem for the very first cycle. Instead, our quantity of interest is a strain indicator that partially describes the damage state of the material. This quantity of interest corresponds to the accumulated plastic strain in octahedral slip systems at the end of the first cycle, which reads:

$$p_{\mathrm{cum}}^o(\boldsymbol{\xi}) = \int_0^{t_c} \sqrt{\frac{2}{3} \ \dot{\boldsymbol{\varepsilon}}^{p,o}(\boldsymbol{\xi}, t) : \dot{\boldsymbol{\varepsilon}}^{p,o}(\boldsymbol{\xi}, t)} \ dt \tag{12.13}$$

with:

$$\dot{\boldsymbol{\varepsilon}}^{p,o} = \sum_{s=1}^{12} \dot{\gamma}_s^o \mathbf{m}_s^o \tag{12.14}$$

It is also common to look at the values of the von Mises equivalent stress field defined as:

$$\sigma_{\text{eq}} = \sqrt{\frac{3}{2}\,\mathbf{s} : \mathbf{s}}, \qquad \mathbf{s} = \boldsymbol{\sigma} - \frac{1}{3}\text{tr}(\boldsymbol{\sigma})\mathbf{1} \tag{12.15}$$

Therefore, the variables considered for the evaluation of the ROM-net and for uncertainty quantification are the accumulated plastic strain p_{cum}^o in octahedral slip systems at the end of the first cycle, and the von Mises stress σ_{eq} obtained when the rotation speed reaches its maximum value. These variables can be visualized in Figure 12.5 for a reference thermal loading.

* *
*

Chapter 13

ROM-net's training phase

Abstract: *This chapter develops the different stages of the ROM-net's training phase for the industrial test case presented in the previous chapter. Given a budget of 200 high-fidelity simulations, a dictionary containing two local ROMs is constructed thanks to the physics-informed clustering procedure. A logistic regression classifier is trained for automatic model recommendation using information identified by feature selection. Finally, an alternative to the Gappy POD for full-field reconstruction is presented.*

Note: We gratefully acknowledge Clément Bénard, Sébastien Da Veiga and Christian Rey for their valuable tips and advice for the work presented in this chapter.

Remark 13.0.1. *This chapter is taken from our paper [50], with some modifications.*

Contents

13.1 Design of numerical experiments

Given the computational cost of high-fidelity mechanical simulations of the high-pressure turbine blade, the training data are sampled from the stochastic model for the thermal loading using a design of experiments (DoE). We are allowed to run 200 high-fidelity simulations, so a database of 200 temperature fields must be built. This database includes two separate datasets coming from two independent DoEs:

- The first dataset is built from a Maximum Projection LHS design (*MaxProj LHS DoE*) and contains 80 points. This dataset will be used for the construction of the dictionary of local ROMs via clustering. The MaxProj LHS DoE has good space-filling properties on projections onto subspaces of any dimension.
- The second dataset is built from a Sobol' sequence (*Sobol' DoE*) of 120 points. Using a suboptimal DoE method ensures that this second dataset is different and independent from the first one. The lower quality of this dataset with respect to the first one is compensated by its larger population. This dataset will be used for learning tasks requiring more training examples than the construction of the local ROMs, namely the classification task for automatic model recommendation, and the training of cluster-specific surrogate models for the reconstruction of full fields from hyper-reduced predictions on a reduced-integration domain. These surrogate models (*Gappy surrogates*) replace the Gappy POD [46] method that is commonly used in hyper-reduced simulations to retrieve dual variables on the whole mesh.

The workflow for each DoE is illustrated on Figure 13.8. We advise the reader to refer to Figure 13.8 while reading this chapter.

The DoEs are built with the platform *Lagun*[1] developed by Safran in collaboration with IFPEN. The fact that these two datasets come from two separate DoEs is beneficial: as each of them is supposed to have good space-filling properties, they are both representative of the possible thermal loading and can therefore be used to define a training set and a test set for a given learning task. For instance, the classifier trained on the Sobol' DoE can be tested on the MaxProj LHS DoE. The local ROMs built from snapshots belonging to the MaxProj LHS DoE can make predictions on the Sobol' DoE that will be used for the training of the Gappy surrogates, which is relevant since the Gappy surrogates are supposed to analyze ROM predictions on new unseen data in the exploitation phase.

Drawing random temperature fields as defined in Equation (12.3) requires sampling data from the random variables $\{\Upsilon_i\}_{0\leq i\leq 4}$, where Υ_0 follows the Bernoulli distribution with parameter 0.5 and the variables Υ_i for $i \in [\![1;4]\!]$ are independent standard normal variables and independent of Υ_0. Both DoE methods (Maximum Projection LHS and Sobol' sequence) generate point clouds with a uniform distribution in the unit hypercube. Figures 13.1 and 13.2 show the projections onto 2-dimensional subspaces of the 5D point clouds used to build our datasets. The marginal distributions are plotted to check that they well approximate the uniform distribution. These point clouds, considered as samples of a random vector $(\chi_0, \chi_1, \chi_2, \chi_3, \chi_4)$ following the uniform distribution on the unit hypercube, are transformed into realizations of the random vector $(\Upsilon_0, \Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4)$ using the following transformations:

$$\Upsilon_0 = \mathbb{1}_{\chi_0>1/2} \quad \text{and} \quad \forall i \in [\![1;4]\!], \quad \Upsilon_i = F^{-1}(\chi_i) \tag{13.1}$$

[1] https://gitlab.com/drti/lagun

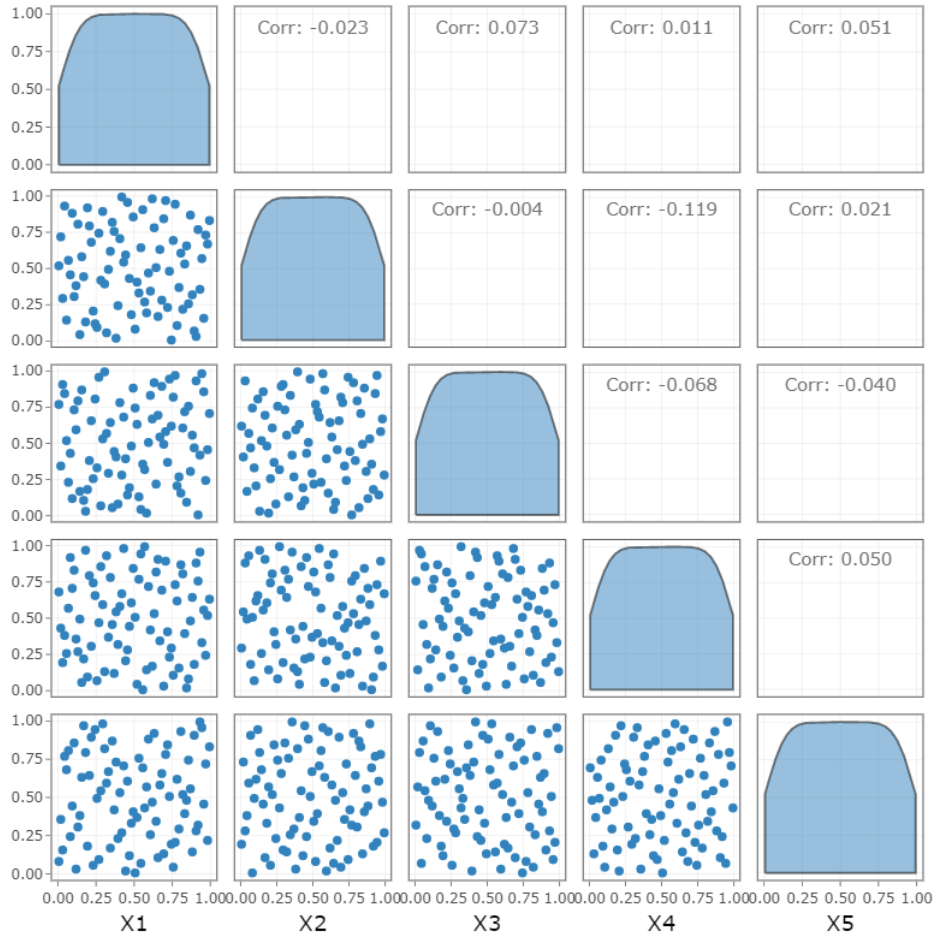

Figure 13.1: Visualization of the MaxProj LHS DoE. The marginal distributions are represented on the diagonal. The 5D DoE is projected on 2D subspaces for visualization purposes, in order to check space-filling properties in 2D.

where F^{-1} is the inverse of the cumulative distribution function of the standard normal distribution. The resulting samples define the MaxProj dataset and the Sobol' dataset of random temperature fields, using Equation (12.3). Each temperature field defines a thermal loading, using Equation (12.1). The 200 corresponding mechanical problems are solved for one loading cycle with the finite-element software *Zset* [79] with the domain decomposition method described in [256], with 48 subdomains. The average computation time for one simulation is 53 minutes.

13.2 ROM dictionary construction

13.2.1 Clustering

The 80 simulations associated to the MaxProj dataset are used as clustering data. Loading all the simulation data takes about 5 minutes, and computing the pairwise ROM-oriented dissimilarities takes only a few seconds. The ROM-oriented dissimilarity defined in Definition 9.2.10 is computed with $n = 1$ using Equation (9.24), *i.e.* each simulation is

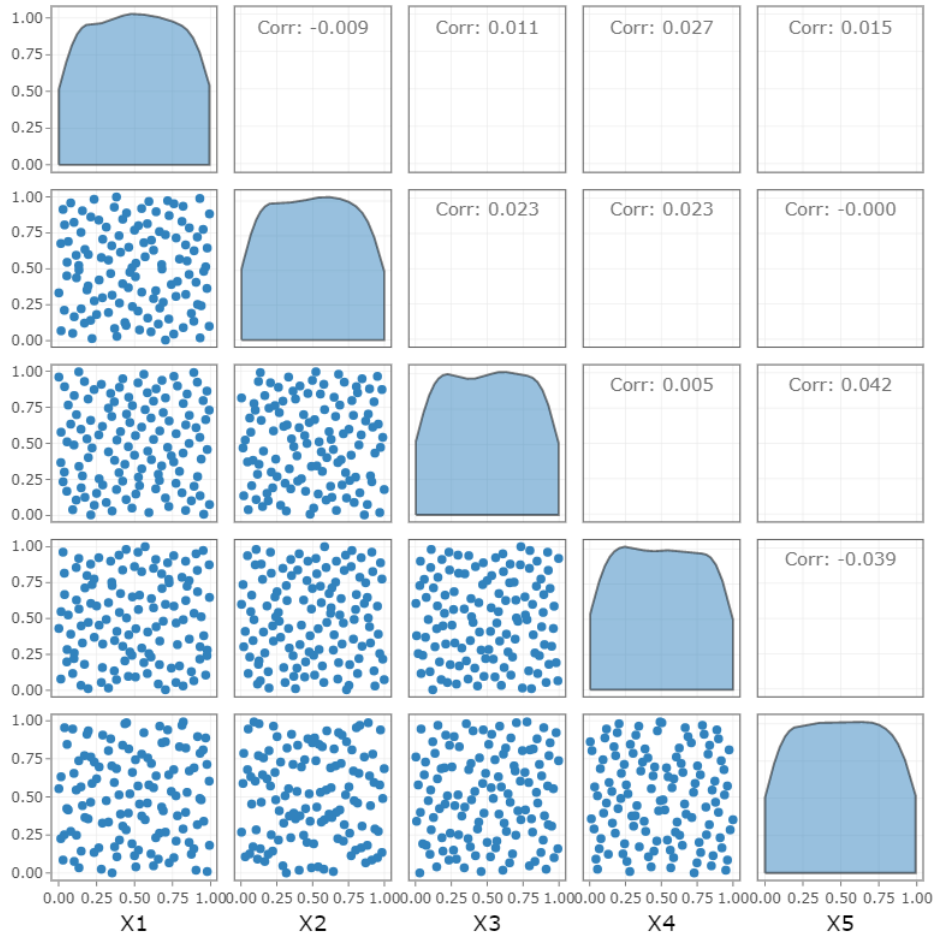

Figure 13.2: Visualization of the Sobol' DoE. The marginal distributions are represented on the diagonal. The 5D DoE is projected on 2D subspaces for visualization purposes, in order to check space-filling properties in 2D.

represented by one field. Two variants are tested: a method-oriented variant, where the dissimilarities are computed from the displacements fields at the maximum rotation speed, and a goal-oriented variant, where the dissimilarities involve the quantity of interest p^o_{cum} (accumulated plastic strain in octahedral slip systems at the end of the simulation). The dataset is partitioned into two clusters using our implementation of PAM [84, 95] k-medoids algorithm, with 10 different random initializations for the medoids. The clustering results can be visualized thanks to Multidimensional Scaling (MDS) [264]. MDS is an information visualization method which consists in finding a low-dimensional dataset $\mathbf{Z}_0$ whose matrix of Euclidean distances $\mathbf{d}(\mathbf{Z}_0)$ is an approximation of the true dissimilarity matrix $\boldsymbol{\delta}$. To that end, a cost function called stress function is minimized with respect to $\mathbf{Z}$:

$$\mathbf{Z}_0 = \arg\min_{\mathbf{Z}} \left(\varsigma(\mathbf{Z};\boldsymbol{\delta})\right) = \arg\min_{\mathbf{Z}} \left(\sum_{i<j} (\delta_{ij} - d_{ij}(\mathbf{Z}))^2\right) \tag{13.2}$$

This minimization problem is solved with the algorithm Scaling by MAjorizing a COmplicated Function (SMACOF, [265]) implemented in Scikit-learn [249]. Figures 13.3 and 13.4 show the clusters on the MDS representations with the two variants of the ROM-oriented

dissimilarity measure. Each figure compares the clustering results with the expected clusters corresponding to $\Upsilon_0 = 0$ (*i.e.* without the perturbation δT_0) and $\Upsilon_0 = 1$ (*i.e.* with the perturbation δT_0). On this example, the method-oriented variant using the displacement field does not manage to distinguish the expected clusters. On the contrary, the goal-oriented variant leads to clusters that almost correspond to the expected ones, with only 4 points with wrong labels out of 80. In the sequel, the results obtained with the goal-oriented variant are considered. The medoids of the two clusters are given in Figure 13.5. Cluster 0 contains temperature fields for which $\Upsilon_0 = 1$, while cluster 1 contains fields for which $\Upsilon_0 = 0$. It can be observed that the quantity of interest clearly differs from one cluster to the other, while the differences are hardly visible on the displacement field. The displacement field combines deformations associated to different phenomena (thermal expansion, elastic strains, viscoplastic strains) that are not necessarily related to damage in the structure, which could explain why the quantity of interest p^o_{cum} seems to be more appropriate for clustering in this example.

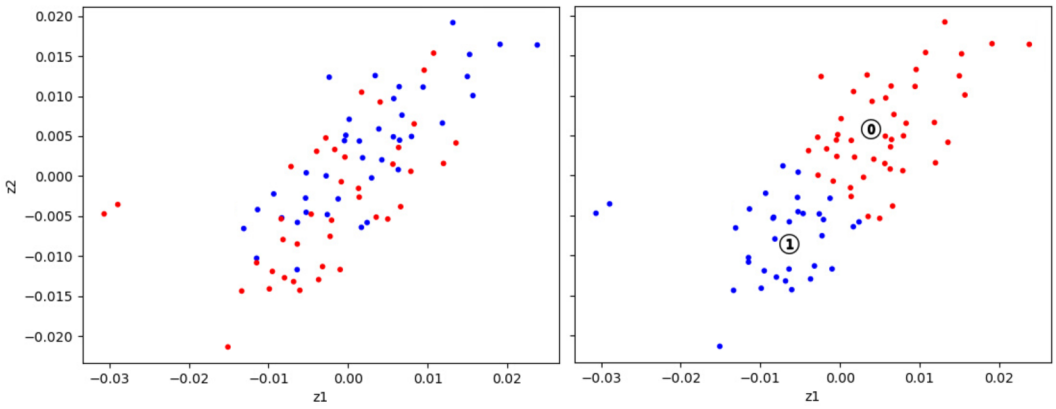

Figure 13.3: MDS representation of the clustering results using the ROM-oriented dissimilarity measure on the displacement field (method-oriented variant). On the left, the colors correspond to the expected clusters. On the right, the colors correspond to the clusters identified by the clustering algorithm. The positions of the labels 0 and 1 coincide with the positions of the clusters' medoids. The MDS relative error $\varsigma(\mathbf{Z}_0; \boldsymbol{\delta})/\varsigma(\mathbf{0}; \boldsymbol{\delta})$ is 7.9%.

13.2.2 Construction of local ROMs

For simplification purposes, and because of the computation time and the memory required to store high-fidelity simulation data, we do not make the distinction between simplified snapshots and high-fidelity snapshots. This means that the simulations used for the physics-informed clustering procedure are not a simplified version of the target problem, so that these simulation data can directly provide snapshots for the construction of the local ROMs. In the case where the target problem would require computing, say, 50 cycles, then the simplified simulations would correspond to the first loading cycle and the high-fidelity ones would contain the responses to 50 loading cycles (or fewer cycles if the ROMs were used for cyclic extrapolation as in [65, 35]). In the present study, the target problem consists in predicting the state of the structure after the first cycle only, which explains why we chose not to distinguish two fidelity levels for this proof of concept.

The question of snapshots selection, though, remains relevant. Indeed, using all the simulations as snapshots would slow down the training phase of the local ROMs. For each

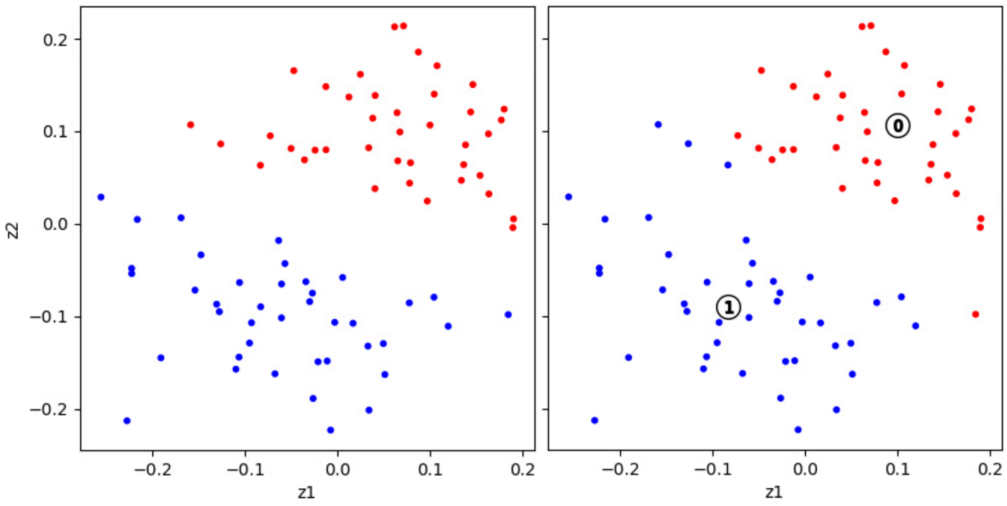

Figure 13.4: MDS representation of the clustering results using the ROM-oriented dissimilarity measure on the quantity of interest p^o_{cum} (goal-oriented variant). On the left, the colors correspond to the expected clusters. On the right, the colors correspond to the clusters identified by the clustering algorithm. The positions of the labels 0 and 1 coincide with the positions of the clusters' medoids. The MDS relative error $\varsigma(\mathbf{Z}_0;\boldsymbol{\delta})/\varsigma(\mathbf{0};\boldsymbol{\delta})$ is 12%.

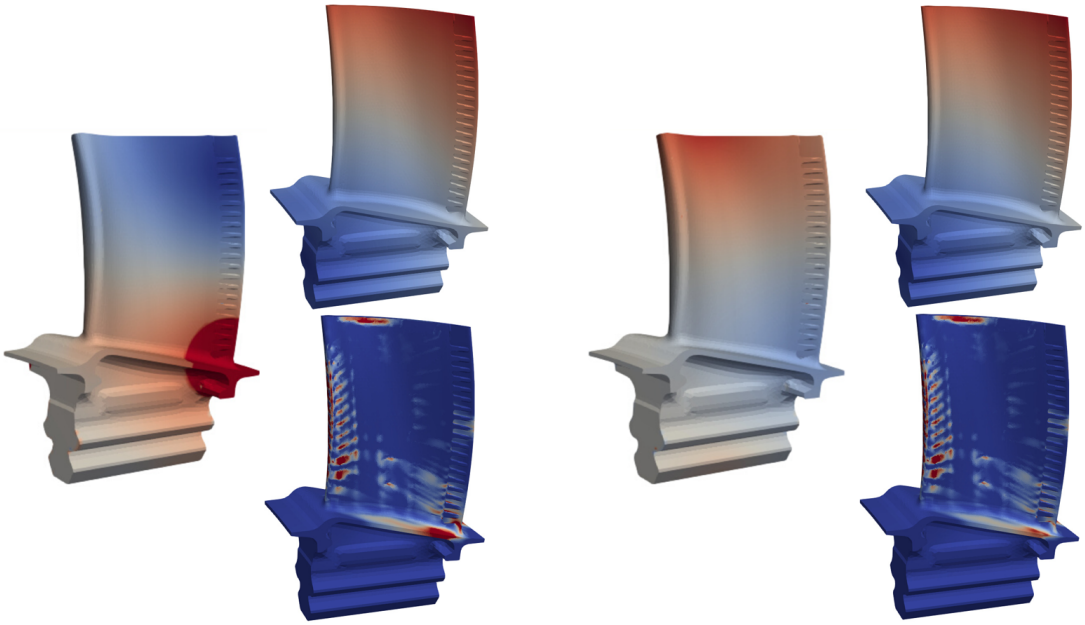

Figure 13.5: The 3 fields on the left correspond to the medoid of cluster 0, and those on the right correspond to the medoid of cluster 1. The fields in the first and the third columns show the differences between the medoids' temperature fields and the reference temperature field T_{ref} (the scale is truncated for the first field). The second and the fourth columns show the displacement magnitude field $\sqrt{\mathbf{u}.\mathbf{u}}$ (top) and the quantity of interest p^o_{cum} (bottom).

cluster, 20 simulations are selected to provide snapshots for the local ROMs. Given that it

represents half of the clusters' populations, the two-stage hierarchical clustering procedure for snapshots selection is not adapted. Instead, the simulations are selected in a maximin greedy approach similar to the one described in Algorithm 2 for seeds selection for data augmentation, starting from the medoid. Figure 13.6 shows which simulations have been selected for the construction of the local ROMs.

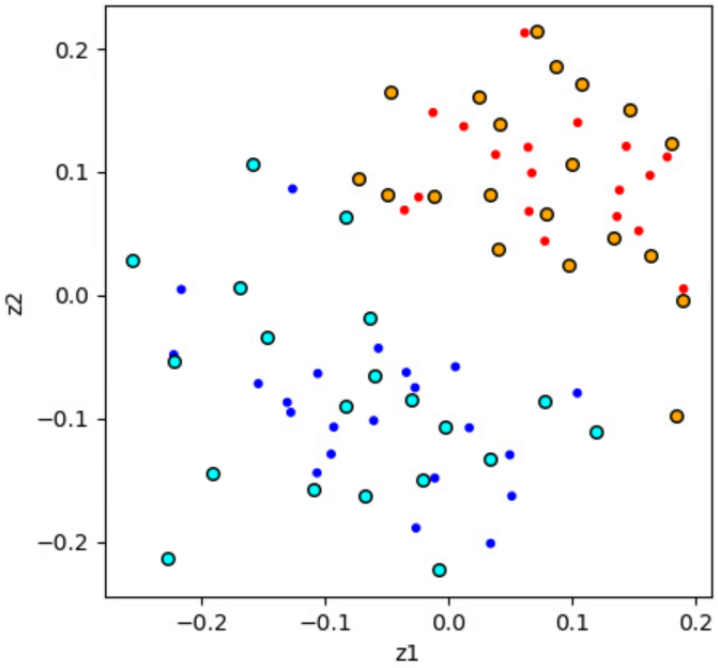

Figure 13.6: MDS representation of the clustering results. Orange points represent the snapshots selected for cluster 0, while the light blue points represent the snapshots selected for cluster 1. For each cluster, the snapshots are selected by a maximin procedure starting from the medoid.

The local ROMs are built with the snapshot POD [181, 182] and the Empirical Cubature Method (ECM [34]) for hyper-reduction, using Safran's module of *Mordicus* code, developed by Fabien Casenave in the *FUI Mordicus* project. This Python code enables for non-intrusive model order reduction, as explained in [35] for an earlier version. The snapshot POD and the ECM are done in parallel with shared memory on 24 cores. The tolerance for the snapshot POD is set to 10^{-8} for the displacement field, and to 10^{-4} for dual variables (the quantity of interest p^o_{cum} and the six components of the stress tensor). The POD bases for the dual variables will be used for their reconstruction with the Gappy surrogates. The tolerance for the ECM is set to 5×10^{-4}. The primal POD bases of both local ROMs contain 18 displacement modes. The local ROM 0 (*resp.* 1) has 10 (*resp.* 12) modes for the quantity of interest p^o_{cum}, and both ROMs have between 8 and 13 modes for stress components. The ECM selects 506 (*resp.* 510) integration points for the reduced-integration domain of ROM 0 (*resp.* 1). Building one local ROM takes approximately 2 hours and 30 minutes.

Remark 13.2.1. *In addition to Mordicus code, the Python library BasicTools*[2] *developed at Safran is used for all operations on finite-element meshes and simulation data.*

[2] https://gitlab.com/drti/basic-tools

13.3 Automatic model recommendation

In this section, a classifier is trained for the automatic model recommendation task. The 120 temperature fields coming from the Sobol' dataset are used as training data for the classifier. Their labels are determined by finding their closest medoid in terms of the ROM-oriented dissimilarity measure. Hence, for each temperature field of the Sobol' dataset, two dissimilarities are computed: one with the medoid of the first cluster, and one with the medoid of the second cluster. Once trained, the classifier can be evaluated on the 80 labeled temperature fields of the MaxProj dataset.

13.3.1 Feature selection

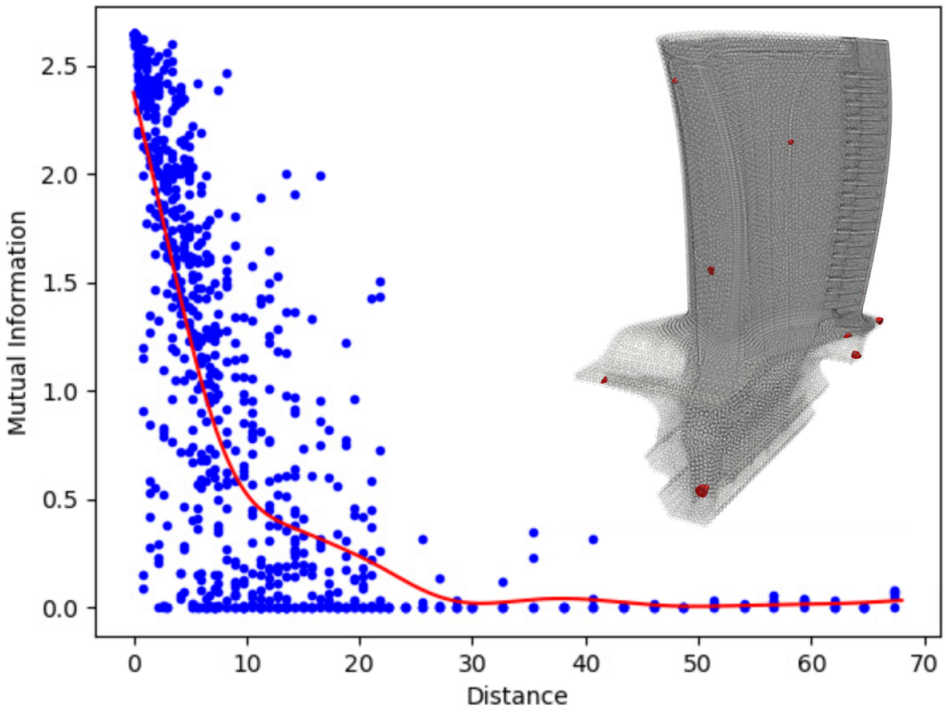

Figure 13.7: Feature selection results. The kriging metamodel for redundancy terms is represented by the red curve and built from 800 true redundancy terms (blue points). The elements containing the selected nodes are represented in the turbine blade geometry.

Each temperature field is discretized on the finite-element mesh, which contains approximately 10^6 nodes. To reduce the dimension of the input space and facilitate the training phase of the classifier, we apply the geostatistical mRMR feature selection algorithm described in Algorithm 1 on data from the Sobol' dataset. First, 800 pairs of nodes are selected in the mesh, which takes 18 seconds. The 800 corresponding redundancy terms are computed with Scikit-learn [249] in less than 3 seconds. Figure 13.7 plots the values of these redundancy terms versus the Euclidean distance between the nodes. Contrary to the simple test case considered for the validation of our feature selection strategy, the correlation between the redundancy mutual information terms and the distance between the nodes is poor, with a lot of noise. This can be due to the fact that the random temperature fields have been built using Gaussian random fields on the outer surface with an

isotropic correlation function depending on the geodesic distance along the surface rather than the Euclidean distance. Since the turbine blade is a relatively thin structure, two nodes, one on the intrados and another one on the extrados, can be close to each other in the Euclidean distance, but with totally uncorrelated temperature fluctuations because of the large geodesic distance separating them. On the contrary, two points on the same side of the turbine blade can have correlated temperature variations while being separated by a Euclidean distance in the order of the blade's thickness. The length of the mutual information's high-variance regime seems to correspond to the blade's chord, which supports this explanation. The thinness of the turbine blade induces anisotropy in the correlation function of the bulk Gaussian random field defining the thermal loading, which implies an anisotropic behavior of the mutual information according to Property 11.1. The use of a local temperature perturbation δT_0 in conjunction with fluctuation modes having larger length scales may also partially explain the large variance of redundancy terms. Nonetheless, it remains clear that redundancy terms are smaller for large distances. This trend is captured by a kriging metamodel (Gaussian process regression [106, 107]) trained with Scikit-learn in a few seconds, with a sum-kernel involving the Matérn kernel with parameter 5/2 (to get a continuous and twice differentiable metamodel) and length scale 1, and a white kernel to estimate the noise level of the signal. The curve of the metamodel is given in Figure 13.7. Then, for each node of the finite-element mesh, the mutual information with the label variable is computed. The computation of these relevance terms (about a million terms) are distributed between 280 cores, which gives a total computation time of 15 minutes. Among the original features, 5986 features are preselected by discarding those with a relevance mutual information lower than 0.05. The geostatistical mRMR selects 11 features in 42 seconds. The corresponding nodes in the finite-element mesh can be visualized in Figure 13.7.

Remark 13.3.1. *The metamodel for redundancy terms could be improved by defining it as a function of the precomputed geodesic distances along the outer surface rather than the Euclidean distances. Each finite-element node would be associated to its nearest neighbor on the outer surface before computing the approximate mutual information from geodesic distances. Apart from this potential improvement, one could also tighten the feature selection stopping criterion in order to select more features and therefore reduce the influence of the errors made by the metamodel, if the selected features do not enable reaching satisfying levels of accuracy for the classification task.*

13.3.2 Classification

The classifier is trained on the Sobol' dataset, using the values of the temperature fields at the 11 nodes identified by the feature selection algorithm. The classifier is a logistic regression [109, 110, 111] with elastic net regularization [105] implemented in Scikit-learn. The two hyperparameters involved in the elastic net regularization are calibrated using 5-fold cross-validation, giving a value of 0.001 for the inverse of the regularization strength, and 0.4 for the weight of the L^1 penalty term (and thus 0.6 for the L^2 penalty term). Thanks to the L^1 penalty term, the classifier only uses 5 features among the 11 input features. The classifier's accuracy, evaluated on the MaxProj dataset to use new unseen data, reaches 98.75%. The confusion matrix indicates that 100% of the test examples belonging to class 0 have been correctly labeled, and that 2.38% of the test examples belonging to class 1 have been misclassified. Table 13.1 summarizes the values of precision, recall and F1-score on test data.

Table 13.1: Classification results.

Class	Precision	Recall	F1-score	Support
0	0.9744	1.0000	0.9870	38
1	1.0000	0.9762	0.9880	42
Accuracy	-	-	0.9875	80
Macro avg	0.9872	0.9881	0.9875	80
Weighted avg	0.9878	0.9875	0.9875	80

Remark 13.3.2. *Our data augmentation algorithm described in Algorithm 3 is not needed here, since the classifier's accuracy is already satisfying.*

13.4 Surrogate model for Gappy reconstruction

When using hyper-reduction, the ROM calls the constitutive equations solver only at the integration points belonging to the reduced-integration domain. It is recalled that the ECM selected 506 (*resp.* 510) integration points for the reduced-integration domain of ROM 0 (*resp.* 1), and that the finite-element mesh initially contains millions of integration points. Therefore, after a reduced simulation, dual variables defined at integration points are known only at integration points of the reduced-integration domain. To retrieve the full field, the Gappy POD [46] finds the coefficients in the POD basis that minimize the squared error between the reconstructed field and the ROM predictions on the reduced-integration domain, see Section 6.4.3. This minimization problem defines the POD coefficients as a linear function of the predicted values on the reduced-integration domain. Although these coefficients are optimal in the least squares sense, they can be biased by the errors made by the ROM. To alleviate this problem, we propose to replace the common Gappy POD procedure by a metamodel or *Gappy surrogate*. The inputs and the outputs of the Gappy surrogate are the same as for the Gappy POD: the input is a vector containing the values of a dual variable on the reduced-integration domain, and the output is a vector containing the optimal coefficients in the POD basis. One Gappy surrogate must be built for each dual variable of interest: in our case, 7 surrogate models per cluster are required, namely one for the quantity of interest p^o_{cum} and one for every component of the Cauchy stress tensor.

The training data for these Gappy surrogates are obtained by running reduced simulations with the local ROMs, using the thermal loadings of the Sobol' dataset. Indeed, the two local ROMs have been built on the MaxProj dataset, therefore thermal loadings of the Sobol' dataset can play the role of test data for the ROMs. For each thermal loading in the Sobol' dataset, the true high-fidelity solution is already known since it has been computed to provide training data for the classifier. In addition, the exact labels for these thermal loadings are known, which means that we know which local ROM to choose for each thermal loading of the Sobol' dataset. Given ROM predictions on the reduced-integration domain, the optimal coefficients in the POD basis are given by the projections of the true prediction made by the high-fidelity model (the finite-element model) onto the POD modes. This provides the true outputs for the Sobol' dataset, which can then be used as a training set for the Gappy surrogates.

Given the high-dimensionality of the input data (there are more than 500 integration points in the reduced-integration domains) with respect to the number of training examples (120 examples), a multi-task Lasso metamodel is used. The hyperparameter controlling the regularization strength is optimized by 5-fold cross-validation. Training the 14 Gappy surrogates (7 for each cluster) takes 1 hour. The Gappy surrogates select between 8% and 18% of the integration points in the reduced-integration domains, thanks to the regularization term used in the loss function of multi-task Lasso (see Equation (5.12)). The mean cross-validated coefficients of determination are 0.9637 (*resp.* 0.8935) for the quantity of interest for cluster 0 (*resp.* cluster 1), and range from 0.9404 to 0.9938 for stress components. These satisfying results mean that it is not required to train a kriging metamodel with the variables selected by Lasso to get nonlinear Gappy surrogates. The Gappy surrogates are then linear, just as the Gappy POD.

Remark 13.4.1. *In this strategy, the local ROMs solve the equations of the mechanical problem, which enables using linear surrogate models to reconstruct dual variables. Using surrogate models instead of local ROMs for the prediction of dual variables directly from the input temperature field would have been more difficult, given the nonlinearities of this mechanical problem and the lack of training data for regression. In addition, such surrogate models would require a parametrization of the input temperature fields, whereas the local ROMs use the exact values of the temperature fields on the RID without assuming any model for the thermal loading. However, it is noteworthy that the use of surrogate models is incompatible with cyclic extrapolation, be it for complete surrogates or Gappy surrogates, since these metamodels are trained with data obtained at a given cycle.*

13.5 Summary

In summary, the dictionary-based ROM-net used for mechanical simulations of the high-pressure turbine blade is made of a dictionary of two local hyper-reduced-order models and a logistic regression classifier. The classifier analyzes the values of the input temperature field at 11 nodes only, identified by our feature selection strategy. For a given thermal loading in the exploitation phase, after the reduced simulation with the local ROM recommended by the classifier, linear cluster-specific Gappy surrogates reconstruct the full dual fields (quantity of interest and stress components) from their predicted values on the reduced-integration domain. Figure 13.8 summarizes the workflow.

* *
*

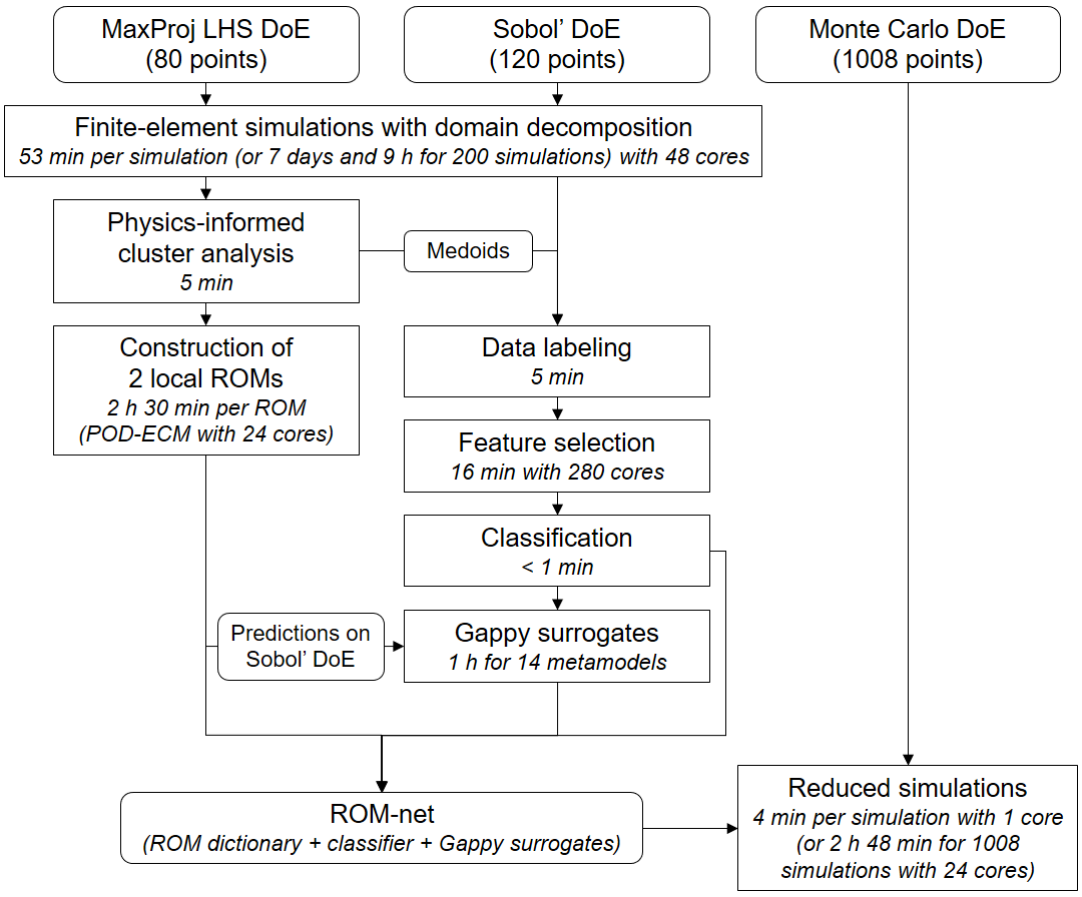

Figure 13.8: Workflow for the training and the exploitation of a dictionary-based ROM-net for uncertainty quantification on the high-pressure turbine blade.

Chapter 14

ROM-net's exploitation phase

Abstract: *This chapter deals with the evaluation of the ROM-net trained in the previous chapter for the industrial test case and its use for uncertainty propagation. The accuracy of the ROM-net's predictions are quantified through different error indicators, on the basis of 20 simulations with new thermal loadings for which the high-fidelity solutions are known. The distributions of quantities of interest are estimated by Monte Carlo simulations.*

Remark 14.0.1. *This chapter is taken from our paper [50], with some modifications.*

Contents

14.1 Uncertainty quantification results

Once trained, the ROM-net can be applied for the quantification of uncertainties on the mechanical behavior of the HP turbine blade resulting from the uncertainties on the thermal loading. Since the ROM-net online operations can be performed sequentially on one single core, 24 cores are used in order to compute the solution for 24 thermal loadings at once. This way, 42 batches of 24 Monte Carlo simulations are run in 2 hours and 48 minutes, using Safran's module of *Mordicus* code developed by Fabien Casenave in the *FUI Mordicus* project. The 1008 thermal loadings used for this study are generated by randomly sampling points from the uniform distribution on the 5D unit hypercube and applying the transformation given in Equation (13.1).

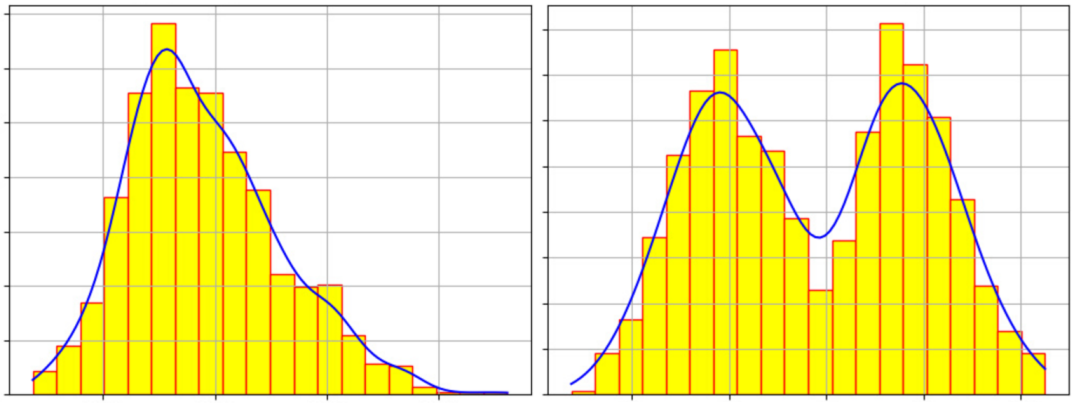

Figure 14.1: Histograms and probability density functions of the quantities of interest $\overline{p}^o_{\text{cum}}$ (left) and $\overline{\sigma}_{\text{eq}}$ (right).

Table 14.1: Widths of the confidence intervals (CI) for the expectations, expressed as percentages of the estimated expectations.

Estimated variable	Confidence level	Relative CI width
$\mathbb{E}[\overline{p}^o_{\text{cum}}]$	0.95	2.16%
$\mathbb{E}[\overline{p}^o_{\text{cum}}]$	0.99	2.84%
$\mathbb{E}[\overline{\sigma}_{\text{eq}}]$	0.95	1.26%
$\mathbb{E}[\overline{\sigma}_{\text{eq}}]$	0.99	1.66%

Let us introduce a zone of interest Ω' defined by all of the integration points at which p^o_{cum} is higher than $0.4 \times \max p^o_{\text{cum}}(\boldsymbol{\xi})$ for the thermal loading defined by $T_{\text{ref}} + \delta T_0$. This zone of interest contains 209 integration points. The values of the variables p^o_{cum} and σ_{eq} averaged over Ω' are denoted by $\overline{p}^o_{\text{cum}}$ and $\overline{\sigma}_{\text{eq}}$.

The expected values of $\overline{p}^o_{\text{cum}}$ and $\overline{\sigma}_{\text{eq}}$ are estimated with the empirical means (see Strong Law of Large Numbers, Theorem 3.3.1). The variances of $\overline{p}^o_{\text{cum}}$ and $\overline{\sigma}_{\text{eq}}$ are computed with the unbiased sample variances defined in Equation (3.4). The Central Limit Theorem 3.3.2 gives asymptotic confidence intervals for the expected values, see Equation (3.6). The widths of the confidence intervals are expressed as a percentage of the estimated value for the expectations in Table 14.1.

The probability density functions of the quantities of interest can be estimated using

Gaussian kernel density estimation (see section 6.6.1. of [55]). Figure 14.1 gives the histograms and estimated distributions for $\overline{p}^o_{\text{cum}}$ and $\overline{\sigma}_{\text{eq}}$. The shapes of these distributions highly depend on the assumptions made for the stochastic thermal loading, and would therefore be different with more realistic statistical assumptions motivated by experts' knowledge and future experimental and numerical results. As observed in Figure 12.5, the stress field is highly sensitive to temperature gradients, which may explain why the distribution of the Von Mises stress is bimodal for the given dataset of temperature fields drawn from a mixture of two Gaussian distributions.

14.2 Validation

Table 14.2: Error indicators for the evaluation of the ROM-net on 20 new thermal loadings.

Error indicator	Errors on p^o_{cum}	Errors on σ_{eq}
Mean L^2 relative error on Ω	1.14%	0.84%
Mean L^2 relative error on Ω'	0.75%	1.46%
Mean L^∞ relative error on Ω	1.11%	1.09%
Mean L^∞ relative error on Ω'	1.05%	2.60%
Mean relative error on value averaged over Ω'	0.50%	0.89%
Mean distance between maxima	0	0

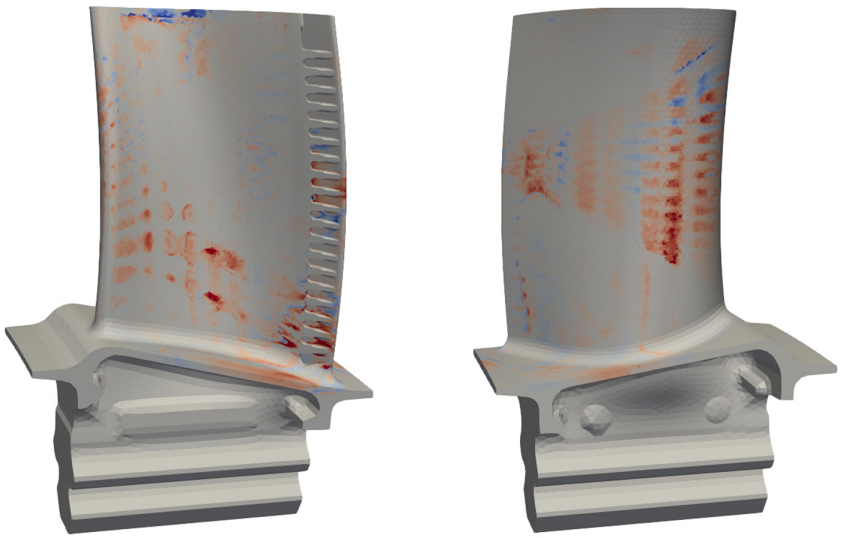

Figure 14.2: Errors on the quantity of interest p^o_{cum}. The red (*resp.* blue) color is used for zones where the quantity of interest is overestimated (*resp.* underestimated).

For validation purposes, the accuracy of the ROM-net is evaluated on 20 Monte Carlo simulations with 20 new thermal loadings. These thermal loadings are generated by randomly sampling points from the uniform distribution on the 5D unit hypercube, and applying the transformation given in Equation (13.1). The reduced simulations are run on single cores with Safran's module of *Mordicus* code. The total computation time for generating a new thermal loading on the fly, selecting a local ROM, running one reduced simulation and reconstructing the quantities of interest is 4 minutes on average. As a

comparison, one single high-fidelity simulation with *Zset* [79] with 48 subdomains takes 53 minutes, which implies that the ROM-net computes 13.25 times faster. However, one high-fidelity simulation requires 48 cores for domain decomposition, whereas the ROM-net works on one single core. Hence, using 48 cores to run 48 reduced simulations in parallel, 636 reduced simulations can be computed in 53 minutes with the ROM-net, while the high-fidelity model only runs one simulation. In addition to the acceleration of numerical simulations, energy consumption is reduced by a factor of 636 in the exploitation phase. In spite of the fast development of high-performance computing, numerical methods computing approximate solutions at reduced computational resources and time are particularly important for many-query problems such as uncertainty quantification, where the intensive use of computational resources is a major concern. Model order reduction and ROM-nets play a prominent role toward *green* numerical simulations [171]. Of course, the number of simulations in the exploitation phase must be large enough to compensate the efforts made in the training phase, like in any machine learning or model order reduction problem.

Figures 14.3 and 14.4 show the results for two simulations belonging to cluster 0 and cluster 1 respectively. These figures give the difference between the current temperature field and the reference one, *i.e.* the field $T - T_{\text{ref}}$, and the resulting variations of the quantity of interest predicted by the ROM-net and the high-fidelity model, *i.e.* $p_{\text{cum}}^{o,\text{ROM}}(T) - p_{\text{cum}}^{o,\text{HF}}(T_{\text{ref}})$ and $p_{\text{cum}}^{o,\text{HF}}(T) - p_{\text{cum}}^{o,\text{HF}}(T_{\text{ref}})$. The signs and the positions of the variations of the quantity of interest seem to be quite well predicted by the ROM-net.

Table 14.2 gives different indicators quantifying the errors made by the ROM-net: the L^2 relative errors on the whole domain Ω and on the zone of interest Ω', the L^∞ relative errors on Ω and Ω', the relative errors on $\overline{p}_{\text{cum}}^{o}$ and $\overline{\sigma}_{\text{eq}}$, and the errors on the locations of the points where the fields p_{cum}^{o} and σ_{eq} reach their maxima. All the relative errors remain in the order of 1% or 2%, which validates the methodology. In addition, the ROM-net perfectly predicts the position of the critical points at which p_{cum}^{o} and σ_{eq} reach their maxima. Figure 14.2 shows errors on the quantity of interest.

* *
*

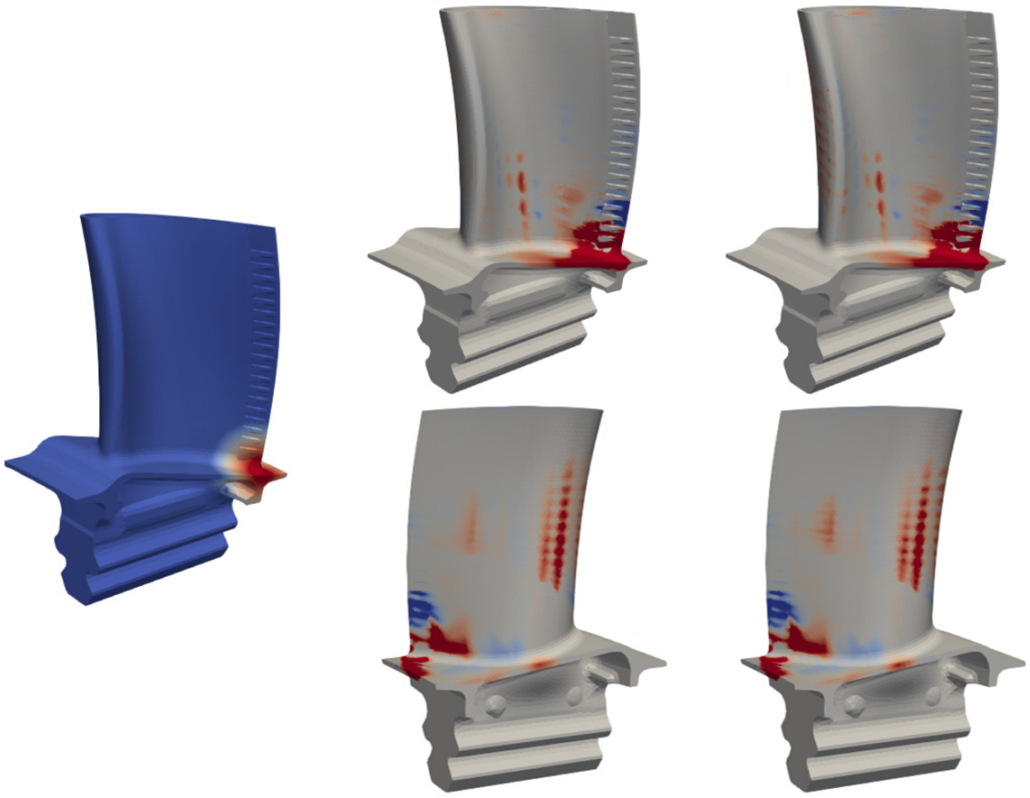

Figure 14.3: Comparison between high-fidelity predictions (middle column) and ROM-net's predictions (right-hand column). The field on the left represents the difference between the current temperature field (belonging to cluster 0) and the reference one. The other fields correspond to the increments of the quantity of interest p^o_{cum} with respect to its reference state obtained with the reference temperature field.

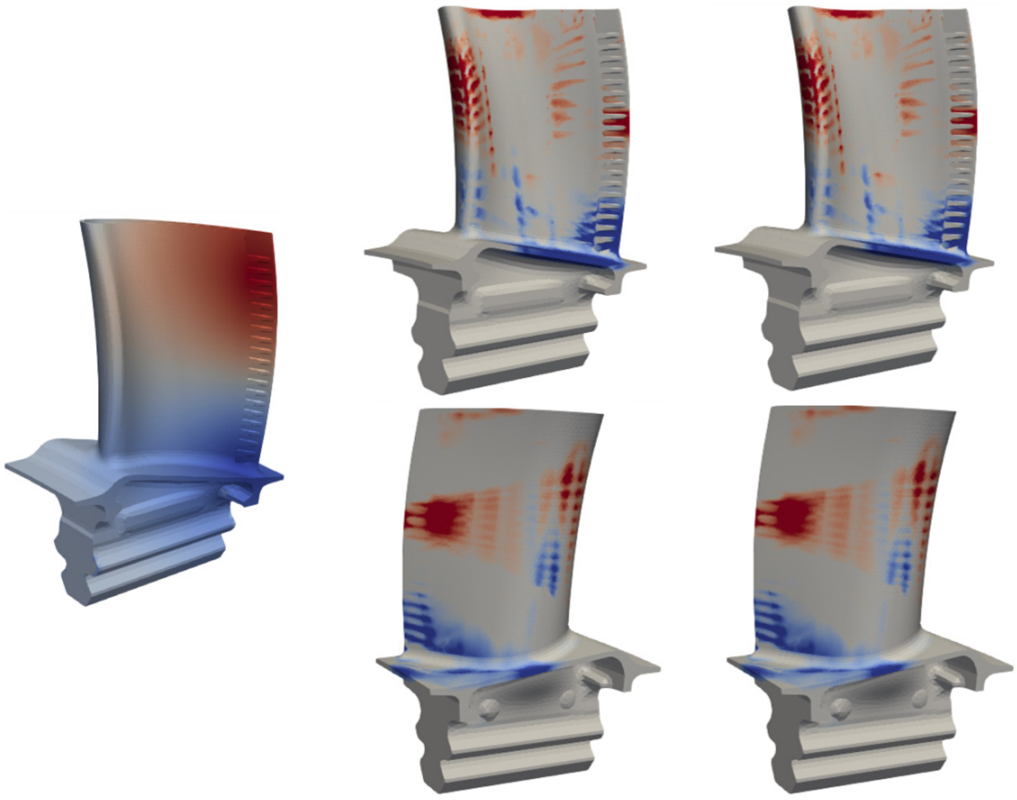

Figure 14.4: Comparison between high-fidelity predictions (middle column) and ROM-net's predictions (right-hand column). The field on the left represents the difference between the current temperature field (belonging to cluster 1) and the reference one. The other fields correspond to the increments of the quantity of interest p^o_{cum} with respect to its reference state obtained with the reference temperature field.

Chapter 15

Conclusion

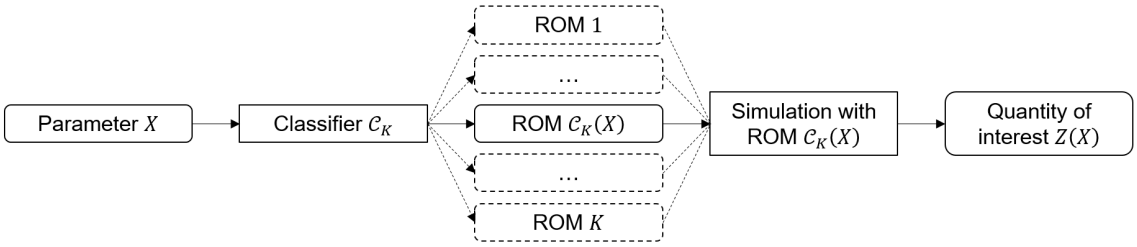

Figure 15.1: Exploitation phase of a dictionary-based ROM-net.

This thesis report summarizes our contributions to nonlinear model order reduction assisted by machine learning methods, already presented in our papers [47, 48, 49, 50, 51] on ROM-nets. Dictionary-based ROM-nets combine a dictionary of local reduced-order models (local ROMs [40, 41]) with a classification algorithm for automatic model recommendation, like in [42, 43], see Figure 15.1. This methodology adapts the choice of the ROM according to the state and the environment of the physical system, which enables accelerating numerical simulations while maintaining reasonable levels of error for many-query problems such as uncertainty quantification. A dictionary-based ROM-net has been successfully applied to a real industrial problem, namely the quantification of uncertainties on the mechanical behavior of a high-pressure turbine blade in an aircraft engine, under nonparametrized variabilities of the thermal loading. On this example, our methodology enabled computing 636 times faster with errors in the order of 1% to 3%.

In our methodology, the solution manifold is partitioned with a physics-informed clustering procedure involving simulation data, with a new ROM-oriented dissimilarity measure that is suitable for nonlinear model order reduction. However, the possibility to use different local ROMs in a single time-dependent simulation as in [40, 41] has not been explored yet within this framework. In addition to this contribution in the field of clustering for the construction of local ROMs, we have developed a new data augmentation algorithm for classification on simulation data, a variant of the mRMR feature selection algorithm, and an a priori efficiency criterion that can be used for hyperparameters calibration.

Beyond uncertainty quantification, industrial needs in terms of numerical methods include algorithms for topology optimization, which is another example of many-query problem where accelerating simulations is essential. Dealing with nonparametrized geometrical variabilities is of paramount importance for the design of mechanical parts and

for the certification of systems with shape uncertainties generated by their manufacturing processes. The application of ROM-nets to nonparametrized geometrical variabilities is a challenging outlook that could be studied in the future. More generally, the incorporation of model order reduction and machine learning in industry needs further developments on error indicators and error estimators, robust methods, and on-the-fly model enrichment from various sources of data (experimental and numerical data, partial data, low-fidelity predictions, streaming data collected by sensors, ...).

This work is part of a global trend in the computational physics research community, consisting in combining numerical methods for physics modeling with recent advances in machine learning. Mixing both worlds is crucial to be able to exploit data while preserving knowledge that has been accumulated over centuries by researchers in physics.

* *
*

References

[1] J. Lumley. The structure of inhomogeneous turbulent flows. *Atm. Turb. and Radio Wave. Prop.*, pages 166–178, 1967.

[2] L. Cordier and M. Bergmann. Proper Orthogonal Decomposition: an overview. In *Lecture series 2002-04, 2003-03 and 2008-01 on post-processing of experimental and numerical data, Von Karman Institute for Fluid Dynamics, 2008.*, page 46 pages. VKI, 2008.

[3] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[4] C.C. Aggarwal. *Neural Networks and Deep Learning*. Springer, 2018.

[5] M. Guo and J.S. Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, 345:75–99, 2019.

[6] R. Dupuis, J.-C. Jouhaud, and P. Sagaut. Surrogate modeling of aerodynamic simulations for multiple operating conditions using machine learning. 12 2019.

[7] G. Ortali, N. Demo, and G. Rozza. Gaussian process approach within a data-driven POD framework for fluid dynamics engineering problems. *arXiv preprint: 2012.01989*, 2020.

[8] Q. Wang, J.S. Hesthaven, and D. Ray. Non-intrusive reduced order modeling of unsteady flows using artificial neural networks with application to a combustion problem. *Journal of Computational Physics*, 384:289–307, 2019.

[9] Z. Wang, D. Xiao, F. Fang, R. Govindan, C. C. Pain, and Y. Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.

[10] S.M. Rahman, S. Pawar, O. San, A. Rasheed, and T. Iliescu. Nonintrusive reduced order modeling framework for quasigeostrophic turbulence. *Physical Review E*, 100, 11 2019.

[11] R. Maulik, A. Mohan, B. Lusch, S. Madireddy, P. Balaprakash, and D. Livescu. Time-series learning of latent-space dynamics for reduced-order model closure. *Physica D: Nonlinear Phenomena*, 405:132368, 04 2020.

[12] F. Gonzalez and M. Balajewicz. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. *arXiv preprint: 1808.01346*, 2018.

[13] S. Wiewel, M. Becher, and N. Thuerey. Latent-space physics: Towards learning the temporal evolution of fluid flow. *arXiv preprint: 1802.10123*, 2019.

[14] R. Maulik, B. Lusch, and P. Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3):037106, Mar 2021.

[15] J. Tencer and K. Potter. Enabling nonlinear manifold projection reduced-order models by extending convolutional neural networks to unstructured data. *ArXiv*, abs/2006.06154, 2020.

[16] R. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. *arXiv preprint: 1806.07366*, 2019.

[17] K. Lee and E. Parish. Parameterized neural ordinary differential equations: Applications to computational physics problems. *arXiv preprint: 2010.14685*, 2020.

[18] M. Drohmann and K. Carlberg. The ROMES method for statistical modeling of reduced-order-model error. *SIAM/ASA Journal on Uncertainty Quantification*, 3:116–145, 02 2015.

[19] S. Trehan, K. Carlberg, and L.J. Durlofsky. Error modeling for surrogates of dynamical systems using machine learning. *arXiv preprint: 1701.03240*, 2017.

[20] B. Freno and K. Carlberg. Machine-learning error models for approximate solutions to parameterized systems of nonlinear equations. 08 2018.

[21] S. Pagani, A. Manzoni, and K. Carlberg. Statistical closure modeling for reduced-order models of stationary systems by the ROMES method. 01 2019.

[22] F. Masi, I. Stefanou, P. Vannucci, and V. Maffi-Berthier. Thermodynamics-based Artificial Neural Networks for constitutive modeling. *Journal of the Mechanics and Physics of Solids*, 147:104277, Feb 2021.

[23] A. Baydin, B. Pearlmutter, A. Radul, and J. Siskind. Automatic Differentiation in Machine Learning: A Survey. *J. Mach. Learn. Res.*, 18(1):5595–5637, January 2017.

[24] Q. Hernandez, A. Badias, D. Gonzalez, F. Chinesta, and E. Cueto. Deep learning of thermodynamics-aware reduced-order models from data. *Computer Methods in Applied Mechanics and Engineering*, 379:113763, Jun 2021.

[25] M. Raissi, P. Perdikaris, and G.E. Karniadakis. 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, 2019.

[26] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint: 1912.00873*, 2019.

[27] E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362:112790, 2020.

[28] H. Kim, J. Kim, S. Won, and C. Lee. Unsupervised deep learning for super-resolution reconstruction of turbulence. *Journal of Fluid Mechanics*, 910, Jan 2021.

[29] J. Zhu, T. Park, P. Isola, and A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.

[30] A. Quarteroni and G. Rozza. *Reduced Order Methods for Modeling and Computational Reduction*. Springer Publishing Company, Incorporated, 2013.

[31] W. Keiper, A. Milde, and S. Volkwein. *Reduced-Order Modeling (ROM) for Simulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing*. Springer International Publishing, 2018.

[32] C. Rowley, T. Colonius, and R. Murray. Model reduction for compressible flow using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189:115–129, 01 2003.

[33] D. Ryckelynck. A priori hyperreduction method: an adaptive approach. *Journal of Computational Physics, Elsevier*, 202(1):346–366, 2005.

[34] J.A. Hernandez, M.A. Caicedo, and A. Ferrer. Dimensional hyper-reduction of nonlinear finite element models via empirical cubature. *Computer methods in applied mechanics and engineering*, 313:687–722, 2017.

[35] F. Casenave, N. Akkari, F. Bordeu, C. Rey, and D. Ryckelynck. 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, 2020.

[36] K. Lee and K.T. Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[37] K. Lee and K. Carlberg. Deep Conservation: A latent-dynamics model for exact satisfaction of physical conservation laws. *arXiv preprint: 1909.09754*, 2020.

[38] Y. Kim, Y. Choi, D. Widemann, and T. Zohdi. A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. *arXiv preprint: 2009.11990*, 2020.

[39] C.C. Aggarwal and C.K. Reddy. *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Taylor & Francis, 2013.

[40] D. Amsallem, M. Zahr, and C. Farhat. Nonlinear model order reduction based on local reduced-order bases. *International Journal for Numerical Methods in Engineering*, pages 1–31, 2012.

[41] K. Washabaugh, D. Amsallem, M. Zahr, and C. Farhat. Nonlinear model reduction for CFD problems using local reduced order bases. *42nd AIAA Fluid Dynamics Conference*, 2012.

[42] B. Peherstorfer, D. Butnaru, K. Willcox, and H.J. Bungartz. Localized Discrete Empirical Interpolation Method. *SIAM Journal on Scientific Computing*, 36, 01 2014.

[43] F. Nguyen, S.M. Barhli, D.P. Muñoz, and D. Ryckelynck. Computer vision with error estimation for reduced order modeling of macroscopic mechanical tests. *Complexity*, 2018.

[44] C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. In *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003*, pages 523–528, Aug 2003.

[45] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency,max-relevance, and min-redundancy. *IEEE transactions on pattern analysis and machine intelligence*, 27:1226–38, 09 2005.

[46] R. Everson and L. Sirovich. Karhunen–Loève procedure for gappy data. *J. Opt. Soc. Am. A*, 12(8):1657–1664, Aug 1995.

[47] T. Daniel, F. Casenave, N. Akkari, and D. Ryckelynck. Model order reduction assisted by deep neural networks (ROM-net). *Advanced Modeling and Simulation in Engineering Sciences*, 7(16), 2020.

[48] T. Daniel, F. Casenave, N. Akkari, and D. Ryckelynck. Data augmentation and feature selection for automatic model recommendation in computational physics. *Mathematical and Computational Applications*, 26(1), 2021.

[49] T. Daniel, A. Ketata, F. Casenave, and D. Ryckelynck. Physics-informed cluster analysis and a priori efficiency criterion for the construction of local reduced-order bases. *arXiv preprint: 2103.13683*, 2021.

[50] T. Daniel, F. Casenave, N. Akkari, D. Ryckelynck, and C. Rey. Uncertainty quantification for industrial design using dictionaries of reduced order models. *arXiv preprint: 2108.04012*, 2021.

[51] T. Daniel, F. Casenave, N. Akkari, and D. Ryckelynck. Optimal piecewise linear data compression for solutions of parametrized partial differential equations. *arXiv preprint: 2108.12291*, 2021.

[52] B. Jourdain. *Probabilités et statistique*. Ellipses, 2016.

[53] F.M. Dekking, C. Kraaikamp, H.P. Lopuhaä, and L.E. Meester. *A Modern Introduction to Probability and Statistics: Understanding Why and How*. Springer Texts in Statistics. Springer London, 2006.

[54] A.A. Borovkov. *Probability Theory*. Universitext. Springer London, 2013.

[55] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer series in statistics. Springer, 2009.

[56] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[57] E. Pineau and M. Lelarge. InfoCatVAE: Representation Learning with Categorical Variational Autoencoders. *arXiv preprint: 1806.08240*, 2018.

[58] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley, 2012.

[59] J. Mercer and A.R. Forsyth. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209(441-458):415–446, 1909.

[60] A. Alexanderian. A brief note on the Karhunen-Loève expansion. *arXiv preprint: 1509.07526*, 2015.

[61] H. Hotelling. *Analysis of a Complex of Statistical Variables Into Principal Components*. Warwick & York, 1933.

[62] P. Abrahamsen. *A Review of Gaussian Random Fields and Correlation Functions*. Norsk Regnesentral - Norwegian Computing Center, 1997.

[63] Par N. Aronszajn. La theorie des noyaux reproduisants et ses applications - premiere partie. *Mathematical Proceedings of the Cambridge Philosophical Society*, 39(3):133–153, 1943.

[64] N. Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[65] F. Casenave and N. Akkari. An error indicator-based adaptive reduced order model for nonlinear structural mechanics - Application to high-pressure turbine blades. *Math. Comput. Appl.*, 24(2), 2019.

[66] C. Soize and C. Farhat. A nonparametric probabilistic approach for quantifying uncertainties in low-dimensional and high-dimensional nonlinear models. *International Journal for Numerical Methods in Engineering*, 109(6):837–888, 2017.

[67] C. Farhat, A. Bos, R. Tezaur, Todd Chapman, P. Avery, and Christian Soize. A stochastic projection-based hyperreduced order model for model-form uncertainties in vibration analysis. 2018.

[68] S.S. Garud, I.A. Karimi, and M. Kraft. Design of computer experiments: A review. *Computers and Chemical Engineering*, 106:71–95, 2017. ESCAPE-26.

[69] N. Metropolis and S. Ulam. The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[70] R. A. Fisher. The design of experiments. 1935.

[71] G.E.P. Box, J.S. Hunter, and W.G. Hunter. *Statistics for Experimenters: Design, Innovation, and Discovery*. Wiley Series in Probability and Statistics. Wiley, 2005.

[72] M.E. Johnson, L.M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131–148, 1990.

[73] M.D. McKay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[74] M.D. Morris and T.J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402, 1995.

[75] V.R. Joseph, G. Evren, and S. Ba. Maximum projection designs for computer experiments. *Biometrika*, 102(2):371–380, 2015.

[76] I.M Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[77] Russel E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7:1–49, 1998.

[78] J.R. Rice and D.M. Tracey. On the ductile enlargement of voids in triaxial stress fields. *Journal of the Mechanics and Physics of Solids*, 17:201–217, 1969.

[79] Mines ParisTech and ONERA the French aerospace lab. Zset: nonlinear material & structure analysis suite. http://www.zset-software.com, 1981-present.

[80] R.E. Bellman. Adaptive control processes. *Princeton University Press*, 1961.

[81] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is nearest neighbor meaningful? *ICDT 1999. LNCS*, 1540, 12 1997.

[82] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.

[83] C.C. Aggarwal. *Data Mining: The Textbook*. Springer International Publishing, 2015.

[84] L. Kaufmann and P. Rousseeuw. Clustering by means of medoids. *Data Analysis based on the L1-Norm and Related Methods*, pages 405–416, 01 1987.

[85] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[86] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. pages 94–105, 1998.

[87] W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 186–195, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[88] U. von Luxburg. A tutorial on spectral clustering. *arXiv preprint: 0711.0189*, 2007.

[89] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 731–737, 1997.

[90] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint: 1801.07648*, 2018.

[91] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018.

[92] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[93] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. Patel, A. Tiwari, M. Er, W. Ding, and C. Lin. A review of clustering techniques and developments. *Neurocomputing*, 267, 07 2017.

[94] A. D. Gordon and J. T. Henderson. An algorithm for euclidean sum of squares classification. *Biometrics*, 33(2):355–362, 1977.

[95] L. Kaufman, P.J.R. Leonard Kaufman, and P.J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. A Wiley-Interscience publication. Wiley, 1990.

[96] L. Kaufman and P. Rousseeuw. *Clustering Large Data Sets*, pages 425–437. 12 1986.

[97] R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. Technical report, CAN, 1994.

[98] R. Ng and J. Han. CLARANS: A method for clustering objects for spatial data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 14:1003– 1016, 10 2002.

[99] E. Schubert and P.J. Rousseeuw. Faster k-medoids clustering: Improving the PAM, CLARA, and CLARANS algorithms. In G. Amato, C. Gennaro, V. Oria, and M. Radovanović, editors, *Similarity Search and Applications*, pages 171–187, Cham, 2019. Springer International Publishing.

[100] H.S. Park and C.H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36:3336–3341, 2009.

[101] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. *Efficient and Effective Clustering Methods for Spatial Data Mining*, 11 2000.

[102] V.N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[103] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2002.

[104] T. Lin, P. Goyal, R. Girshick, K. He, and P Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[105] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. 67(2):301–320, 2005.

[106] Rasmussen C.E. Gaussian processes in machine learning. *Advanced Lectures on Machine Learning*, 3176, 2004.

[107] C.E. Rasmussen and C. Williams. Gaussian processes for machine learning. *MIT Press*, 2006.

[108] J. Kleijnen. Kriging Metamodeling in Simulation: A Review. *European Journal of Operational Research*, 192:707–716, 02 2007.

[109] J. Berkson. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227):357–365, 1944.

[110] D.R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242, 1958.

[111] D.R. Cox. Some procedures connected with the logistic qualitative response curve. *Research papers in probability and statistics*, pages 55–71, 1966.

[112] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[113] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifier. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, 5, 08 1996.

[114] A.G. Ivakhnenko and V.G. Lapa. Cybernetic predicting devices. 1966.

[115] R. D. Joseph. Contributions to perceptron theory. phd thesis, cornell univ. 1961.

[116] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks : the official journal of the International Neural Network Society*, 61:85–117, 2015.

[117] L. Breiman, J.H. Friedman, R.A. Olshen, and R.A. Stone. Classification and regression trees. 1983.

[118] M. E. Maron. Automatic indexing: An experimental inquiry. *J. ACM*, 8(3):404–417, July 1961.

[119] H. Zhang. The optimality of naive bayes. volume 2, 01 2004.

[120] Cover, T. and Hart, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[121] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, 2006:161–168, 06 2006.

[122] S. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (Ljubljana)*, 31, 10 2007.

[123] M. Perez-Ortiz, S. Jimenez-Fernandez, P.A. Gutierrez, E. Alexandre, C. Martinez, and S. Salcedo-Sanz. A review of classification problems and algorithms in renewable energy applications. *Energies*, 9:607, 08 2016.

[124] S Buyukozturk and Omay Cokluk Bokeoglu. Discriminant function analysis: Concept and application. *Egitim Arastirmalari - Eurasian Journal of Educational Research*, 8:73–92, 01 2008.

[125] Dietrich Wettschereck, David W. Aha, and Takao Mohri. *A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms*, pages 273–314. Kluwer Academic Publishers, USA, 1997.

[126] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[127] L. Wang. Support vector machines: Theory and applications. *Studies in fuzziness and soft computing, v 177*, 302, 01 2005.

[128] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[129] H. Bisgin, T. Bera, H. Ding, H. Semey, L. Wu, Z. Liu, A. Barnes, D. Langley, M. Pava-Ripoll, H. Vyas, W. Tong, and J. Xu. Comparing svm and ann based machine learning methods for species identification of food contaminating beetles. *Scientific Reports*, 8, 12 2018.

[130] Liyang, W., Yongyi, Y., Nishikawa, R.M., and Jiang, Y. A study on several machine-learning methods for classification of malignant and benign clustered microcalcifications. *IEEE Transactions on Medical Imaging*, 24(3):371–380, 2005.

[131] F.F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[132] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 06 2014.

[133] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 02 2015.

[134] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[135] D.P. Kingma and J. Ba. Adam: a method for stochastic optimization. https://arxiv.org/abs/1412.6980, 2014.

[136] S. Haykin. Neural Networks - A Comprehensive Foundation. *Second edition*, pages 351–391, 1999.

[137] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[138] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint: 1409.1556*, 2014.

[139] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[140] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[141] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20:832–844, 1998.

[142] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[143] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, 1995.

[144] T.J. Hastie, S. Rosset, J. Zhu, and H. Zou. Multi-class adaboost. 2009.

[145] J. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29, 11 2000.

[146] J. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 02 2002.

[147] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent in function space. 06 1999.

[148] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. pages 512–518, 01 1999.

[149] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[150] B. Baharudin, L.H. Lee, K. Khan, and A. Khan. A review of machine learning algorithms for text-documents classification. *Journal of Advances in Information Technology*, 1, 02 2010.

[151] K. Kourou, T.P. Exarchos, K.P. Exarchos, M.V. Karamouzis, and D.I. Fotiadis. Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13:8 – 17, 2015.

[152] F. Fritzen, M. Fernández, and F. Larsson. On-the-fly adaptivity for nonlinear twoscale simulations using artificial neural networks and reduced order modeling. *Frontiers in Materials*, 6:75, 2019.

[153] R. Maulik, O. San, J. Jacob, and C. Crick. Sub-grid scale model classification and blending through deep learning. *Journal of Fluid Mechanics*, 870:784–812, 07 2019.

[154] M.G. Kapteyn, D.J. Knezevic, and K.E. Willcox. *Toward predictive digital twins via component-based reduced-order models and interpretable machine learning.*

[155] M.G. Kapteyn and K.E. Willcox. From physics-based models to predictive digital twins via interpretable machine learning. *arXiv preprint: 2004.11356*, 2020.

[156] R. Maulik, O. San, and J.D. Jacob. Spatiotemporally dynamic implicit large eddy simulation using machine learning classifiers. *Physica D: Nonlinear Phenomena*, 406:132409, 2020.

[157] C. Meneveau and P. Sagaut. *Large Eddy Simulation for Incompressible Flows: An Introduction.* Scientific Computation. Springer Berlin Heidelberg, 2006.

[158] F. Feyel. Multiscale FE2 elastoviscoplastic analysis of composite structures. *Computational Materials Science*, 16(1):344 – 354, 1999.

[159] F. Fritzen and O. Kunc. Two-stage data-driven homogenization for nonlinear solids using a reduced order model. *European Journal of Mechanics - A/Solids*, 69:201 – 220, 2018.

[160] D. Bertsimas and J. Dunn. Optimal classification trees. *Machine Learning*, 106, 04 2017.

[161] Phuong Huynh, D.B., Knezevic, D.J., and Patera, A.T. A static condensation reduced basis element method : approximation and a posteriori error estimation. *ESAIM: M2AN*, 47(1):213–251, 2013.

[162] J. Eftang, D. Huynnh, D. Knezevic, E. Ronqust, and A. Patera. Adaptive port reduction in static condensation. 01 2015.

[163] J. Eftang and A. Patera. Port reduction in parametrized component static condensation: Approximation and a posteriori error estimation. *International Journal for Numerical Methods in Engineering*, 96, 07 2013.

[164] K. Smetana and A. Patera. Optimal local approximation spaces for component-based static condensation procedures. *SIAM Journal on Scientific Computing*, 38:A3318–, 10 2016.

[165] S. Chaturantabut and D. Sorensen. Discrete empirical interpolation for nonlinear model reduction. *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference, CDC/CCC 2009, Proceedings of the 48th IEEE Conference*, pages 4316–4321, 2010.

[166] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen. Recent advances in convolutional neural networks. *Patter Recognition*, 77:354–377, 2018.

[167] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1):16 – 28, 2014.

[168] A. Janecek, W. Gansterer, M. Demel, and G. Ecker. On the relationship between feature selection and classification accuracy. *Journal of Machine Learning Research - Proceedings Track*, 4:90–105, 01 2008.

[169] C. Meyer. *Matrix Analysis and Applied Linear Algebra Book and Solutions Manual*. 01 2000.

[170] J. Vergara and P. Estevez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, 24, 01 2014.

[171] D. Ryckelynck, D.M. Benziane, A. Musienko, and G. Cailletaud. Toward "green" mechanical simulations in materials science. *European Journal of Computational Mechanics*, 19(4):365–388, 2010.

[172] F. Chinesta, P. Ladeveze, and E. Cueto. A short review on model order reduction based on Proper Generalized Decomposition. *Archives of Computational Methods in Engineering*, 18:395–404, 11 2011.

[173] F. Chinesta and E. Cueto. *PGD-Based Modeling of Materials, Structures and Processes*. 01 2014.

[174] C. Prud'homme, D. Rovas, K. Veroy, L. Machiels, Y. Maday, A. Patera, and G. Turinici. Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124:70, 03 2002.

[175] G. Rozza, D. Huynh, and A. Patera. Reduced basis approximation and a posteriori error estimation for affinely parametrized elliptic coercive partial differential equations. *Archives of Computational Methods in Engineering*, 15:1–47, 09 2007.

[176] T. Henri and J.P. Yvon. Convergence estimates of POD-Galerkin methods for parabolic problems. volume 166, pages 295–306, 01 2006.

[177] I. Gohberg, S. Goldberg, and M.A. Kaashoek. *Classes of Linear Operators*. Number vol. 1 in Classes of Linear Operators. Springer, 1990.

[178] C. Cheverry and N. Raymond. Handbook of Spectral Theory. Lecture, September 2019.

[179] S. Djouadi. On the optimality of the proper orthogonal decomposition and balanced truncation. pages 4221 – 4226, 01 2009.

[180] S. Djouadi and S. Sahyoun. On a generalization of the proper orthogonal decomposition and optimal construction of reduced order models. In *2012 American Control Conference (ACC)*, pages 1436–1441, 2012.

[181] L. Sirovich. Turbulence and the dynamics of coherent structures, Parts I, II and III. *Quarterly of Applied Mathematics*, XLV:561 – 590, 1987.

[182] A. Chatterjee. An introduction to the proper orthogonal decomposition. *Current Science*, 78:808 – 817, 2000.

[183] M. Meyer and H.G. Matthies. Efficient model reduction in non-linear dynamics using the Karhunen-Loève expansion and dual-weighted-residual methods. *Computational Mechanics*, 31:179–191, 05 2003.

[184] D. Ryckelynck, K. Lampoh, and S. Quilicy. Hyper-reduced predictions for lifetime assessment of elasto-plastic structures. *Computational Micromechanics of Materials*, 2015.

[185] M. Barrault, Y. Maday, N.C. Nguyen, and A.T. Patera. An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematiques*, 339(9):666–672, 2004.

[186] P. Astrid, S. Weiland, K. Willcox, and T. Backx. Missing point estimation in models described by proper orthogonal decomposition. *Proceedings of the IEEE Conference on Decision and Control*, 53(10):1767–1772, 2005.

[187] N.C. Nguyen, A.T. Patera, and J. Peraire. A best points interpolation method for efficient approximation of parametrized functions. *Internat. J. Numer. Methods Engrg.*, 73:521–543, 2008.

[188] K. Carlberg, C. Farhat, J. Cortial, and D. Amsallem. The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

[189] C. Farhat, P. Avery, T. Chapman, and J. Cortial. 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, 2014.

[190] M. Yano and A.T. Patera. An LP empirical quadrature procedure for reduced basis treatment of parametrized nonlinear PDEs. *Computer Methods in Applied Mechanics and Engineering*, 344, 2018.

[191] J. Fauque, I. Ramiere, and D. Ryckelynck. Hybrid hyper-reduced modeling for contact mechanics problems. *International Journal for Numerical Methods in Engineering*, 115(1):117–139, 2018.

[192] S. Feld-Payet, J. Besson, and F. Feyel. Finite element analysis of damage in ductile structures using a nonlocal model combined with a three-field formulation. *International Journal of Damage Mechanics*, 20(5):655–680, 2011.

[193] V. Davaze, N. Vallino, B. Langrand, J. Besson, and S. Feld-Payet. A non-local damage approach compatible with dynamic explicit simulations and parallel computing. *International Journal of Solids and Structures*, 2021.

[194] M. Yaghoobi, D. Wu, and M.E. Davies. Fast non-negative orthogonal matching pursuit. *IEEE Signal Processing Letters*, 22:1229–1233, 2015.

[195] M. Bachmayr and A. Cohen. Kolmogorov widths and low-rank approximations of parametric elliptic PDEs. *Mathematics of Computation*, 86(304):701–724, 2017.

[196] Angelo Iollo and Damiano Lombardi. Advection modes by optimal mass transfer. *Phys. Rev. E*, 89:022923, Feb 2014.

[197] J. Reiss, P. Schulze, J. Sesterhenn, and V. Mehrmann. The Shifted Proper Orthogonal Decomposition: A mode decomposition for multiple transport phenomena. *SIAM Journal on Scientific Computing*, 40(3):A1322–A1344, 2018.

[198] N. Cagniart, Y. Maday, and B. Stamm. Model order reduction for problems with large convection effects. *In: Chetverushkin B., Fitzgibbon W., Kuznetsov Y., Neittaanmäki P., Periaux J., Pironneau O. (eds) Contributions to Partial Differential Equations and Applications. Computational Methods in Applied Sciences*, 47, 2019.

[199] R. Zimmermann, B. Peherstorfer, and K. Willcox. Geometric subspace updates with applications to online adaptive nonlinear model reduction. *SIAM Journal on Matrix Analysis and Applications*, 39, 11 2017.

[200] T. Kim and D.L. James. Skipping steps in deformable simulation with online model reduction. *ACM Trans. Graph.*, 28(5):1–9, December 2009.

[201] M. Ohlberger and F. Schindler. Error control for the localized reduced basis multiscale method with adaptive on-line enrichment. *SIAM Journal on Scientific Computing*, 37(6):A2865–A2895, 2015.

[202] W. He, P. Avery, and C. Farhat. In-situ adaptive reduction of nonlinear multiscale structural dynamics models. *arXiv preprint: 2004.00153*, 2020.

[203] B. Peherstorfer and K. Willcox. Online adaptive model reduction for nonlinear systems via low-rank updates. *SIAM Journal on Scientific Computing*, 37(4):A2123–A2150, 2015.

[204] B. Peherstorfer. Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling. *SIAM Journal on Scientific Computing*, 42(5):A2803–A2836, 2020.

[205] P.A. Etter and K.T. Carlberg. Online adaptive basis refinement and compression for reduced-order models via vector-space sieving. *Computer Methods in Applied Mechanics and Engineering*, 364, 2020.

[206] D. Amsallem and C. Farhat. An online method for interpolating linear parametric reduced-order models. *SIAM Journal on Scientific Computing*, 33(5):2169–2198, 2011.

[207] T. Lieu and M. Lesoinne. Parameter adaptation of reduced order models for three-dimensional flutter analysis. *AIAA Paper 2004-0888*, 2004.

[208] T. Lieu, C. Farhat, and M. Lesoinne. POD-based aeroelastic analysis of a complete F-16 configuration: ROM adaptation and demonstration. *AIAA Paper 2005-2295*, 2005.

[209] T. Lieu and C. Farhat. Adaptation of POD-based aeroelastic ROMs for varying Mach number and angle of attack: Application to a complete F-16 configuration. *AIAA Paper 2005-7666*, 2005.

[210] T. Lieu, C. Farhat, and M. Lesoinne. Reduced-order fluid/structure modeling of a complete aircraft configuration. *Computer Methods in Applied Mechanics and Engineering*, 195:5730–5742, 2006.

[211] T. Lieu and C. Farhat. Adaptation of aeroelastic reduced-order models and application to an F-16 configuration. *AIAA Journal*, 45:1244–1257, 2007.

[212] D. Amsallem and C. Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *AIAA Journal*, 46(7):1803–1813, 2008.

[213] D. Amsallem, J. Cortial, and C. Farhat. Towards real-time computational-fluid-dynamics-based aeroelastic computations using a database of reduced-order information. *AIAA Journal*, 48(9):2029–2037, 2010.

[214] D. Amsallem, R. Tezaur, and C. Farhat. Real-time solution of linear computational problems using databases of parametric reduced-order models with arbitrary underlying meshes. *Journal of Computational Physics*, 326:373 – 397, 2016.

[215] R. Mosquera, A. Hamdouni, A. El Hamidi, and C. Allery. POD basis interpolation via Inverse Distance Weighting on Grassmann manifolds. *Discrete and Continuous Dynamical Systems, Series S.*, 12(6):1743–1759, 2018.

[216] R. Mosquera, A. El Hamidi, A. Hamdouni, and A. Falaize. Generalization of the Neville-Aitken Interpolation Algorithm on Grassmann Manifolds : Applications to Reduced Order Model. https://arxiv.org/pdf/1907.02831.pdf, 2019.

[217] Y. Choi, G. Boncoraglio, S. Anderson, D. Amsallem, and C. Farhat. Gradient-based constrained optimization using a database of linear reduced-order models. *Journal of Computational Physics*, 423:109787, 2020.

[218] Y. Maday and B. Stamm. Locally adaptive greedy approximations for anisotropic parameter reduced basis spaces. *SIAM Journal on Scientific Computing*, 35(6):A2417–A2441, 2013.

[219] S. Kaulmann and B. Haasdonk. Online greedy reduced basis construction using dictionaries. 2012.

[220] M. Drohmann, B. Haasdonk, and M. Ohlberger. Adaptive reduced basis methods for nonlinear convection–diffusion equations. volume 4, pages 369–377, 12 2010.

[221] M. Dihlmann, M. Drohmann, and B. Haasdonk. Model reduction of parametrized evolution problems using the reduced basis method with adaptive time partitioning. 01 2011.

[222] J. Eftang, A. Patera, and E. Ronquist. An "hp" certified reduced basis method for parametrized elliptic partial differential equations. *SIAM J. Scientific Computing*, 32:3170–3200, 09 2010.

[223] B. Haasdonk, M. Dihlmann, and M. Ohlberger. A training set and multiple bases generation approach for parametrized model reduction based on adaptive grids in parameter space. *Mathematical and Computer Modelling of Dynamical Systems*, 17:423–442, 08 2011.

[224] D. Amsallem, M. Zahr, and K. Washabaugh. Fast local reduced basis updates for the efficient reduction of nonlinear systems with hyper-reduction. *Advances in Computational Mathematics*, 41, 02 2015.

[225] S. Grimberg, C. Farhat, R. Tezaur, and C. Bou-Mosleh. Mesh sampling and weighting for the hyperreduction of nonlinear Petrov-Galerkin reduced-order models with local reduced-order bases. 08 2020.

[226] A. Buhr, L. Iapichino, and K. Smetana. *6 Localized model reduction for parameterized problems*, pages 245–306. De Gruyter, 2020.

[227] H. Proudhon, A. Moffat, I. Sinclair, and et al. Three-dimensional characterisation and modelling of small fatigue corner cracks in high strength al-alloys. *COMPTES RENDUS PHYSIQUE*, 13:316–327, 2012.

[228] A. Buljac, M. Shakoor, J. Neggers, M. Bernacki, P.O. Bouchard, L. Helfen, T.F. Morgeneyer, and F. Hild. Numerical validation framework for micromechanical simulations based on synchrotron 3d imaging. *Computational Mechanics*, 59:419–441, 2017.

[229] N. Akkari, F. Casenave, and V. Moureau. Time Stable Reduced Order Modeling by an Enhanced Reduced Order Basis of the Turbulent and Incompressible 3D Navier–Stokes Equations. *Mathematical and Computational Applications*, 24(2), 2019.

[230] D. Ryckelynck, T. Goessel, and F. Nguyen. Mechanical dissimilarity of defects in welded joints via Grassmann manifold and machine learning. Preprint, July 2020.

[231] A. Bjorck and G. Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27:123, 07 1973.

[232] J. Conway, R. Hardin, and N. Sloane. Packing lines, planes, etc.: Packings in grassmannian space. *Experimental Mathematics*, 5:139–159, 01 1996.

[233] D. Hua. A regularity result for boundary value problems on Lipschitz domains. *Annales de la Faculté des sciences de Toulouse : Mathématiques*, Ser. 5, 10(2):325–333, 1989.

[234] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 06 2014.

[235] N. Akkari, F. Casenave, M.E. Perrin, and D. Ryckelynck. *Deep Convolutional Generative Adversarial Networks Applied to 2D Incompressible and Unsteady Fluid Flows*, pages 264–276. 07 2020.

[236] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 01 2002.

[237] H. He, Y. Bai, E. Garcia, and S. Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. pages 1322 – 1328, 07 2008.

[238] L. Adrian, R.J. Adrian, and J. Westerweel. *Particle Image Velocimetry*. Cambridge Aerospace Series. Cambridge University Press, 2011.

[239] T. Chu, W. Ranson, and M. Sutton. Applications of digital-image-correlation techniques to experimental mechanics. *Experimental Mechanics*, 25:232–244, 09 1985.

[240] H. Mueller. Theory of photoelasticity in amorphous solids. *Physics*, 6(6):179–184, 1935.

[241] U. Fey and Y. Egami. Transition detection by temperature-sensitive paint. volume Chap. 7.4 of *Handbook with DVD-ROM*, pages 537–552. Springer Berlin, Heidelberg, New York, 2007.

[242] K. Ye and L.H. Lim. Schubert varieties and distances between subspaces of different dimensions. *SIAM J. Matrix Anal. Appl.*, 37(3):1176–1197, 2016.

[243] J.L. Chaboche. A review of some plasticity and viscoplasticity constitutive theories. *International Journal of Plasticity*, 24(10):1642 – 1693, 2008. Special Issue in Honor of Jean-Louis Chaboche.

[244] H.G. Matthies, C.E. Brenner, C.G. Bucher, and C. Guedes Soares. Uncertainties in probabilistic numerical analysis of structures and solids - Stochastic finite elements. *Structural Safety*, 19(3):283 – 336, 1997. Devoted to the work of the Joint Committee on Structural Safety.

[245] B. Sudret and A. Der Kiureghian. Stochastic finite element methods and reliability: A state-of-the-art report. 01 2000.

[246] H.G. Matthies and A. Keese. Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194(12):1295 – 1331, 2005. Special Issue on Computational Methods in Stochastic Mechanics and Reliability Analysis.

[247] B. N. Khoromskij, A. Litvinenko, and H. G. Matthies. Application of Hierarchical Matrices for computing the Karhunen-Loève Expansion. *Computing*, 84(1–2):49–67, 2009.

[248] D. Amsallem, M. Zahr, Y. Choi, and C. Farhat. Design optimization using hyper-reduced-order models. *Structural and Multidisciplinary Optimization*, 51:919–940, 04 2014.

[249] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[250] R.T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 1970.

[251] C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995.

[252] Z. He, L. Xie, X. Chen, Y. Zhang, Y. Wang, and Q. Tian. Data augmentation revisited: Rethinking the distribution gap between clean and augmented data. 09 2019.

[253] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019.

[254] M. Gonen and E. Alpaydin. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 07 2011.

[255] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, Jul 2017.

[256] C. Bovet, A. Parret-Freaud, N. Spillane, and P. Gosselet. Adaptive multipreconditioned FETI: Scalability results and robustness assessment. *Computers and Structures*, 193:1 – 20, 2017.

[257] C. Scarth et al. Random field simulation over curved surfaces: Applications to computational structural mechanics. *Comput. Methods Appl. Mech. Engrg.*, 2018.

[258] J.S.B. Mitchell, D.M. Mount, and C.H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987.

[259] V. Surazhsky, T. Surazhsky, D. Kirsanov, S.J. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph.*, 24(3):553–560, 2005.

[260] R. J. Asaro. Crystal Plasticity. *Journal of Applied Mechanics*, 50(4b):921–934, 12 1983.

[261] L. Meric, P. Poubanne, and G. Cailletaud. Single crystal modeling for structural calculations: Part 1 - Model presentation. *Journal of Engineering Materials and Technology*, 113(1):162–170, 01 1991.

[262] F. Gallerneau. Etude et modelisation de l'endommagement d'un superalliage monocristallin revetu pour aube de turbine. *PhD thesis - Mines ParisTech*, 1995.

[263] F. Gallerneau, D. Nouailhas, and J.L. Chaboche. A fatigue damage model including interaction effects with oxidation and creep damages. *Fatigue' 96: Proceedings of the Sixth International Fatigue Congress*, 2:861–866, 01 1996.

[264] I. Borg and P. Groenen. Modern multidimensional scaling - Theory and applications. *2nd edition, Springer-Verlag*, 2005.

[265] J. de Leeuw. Applications of convex analysis to multidimensional scaling. *in JR Barra, F Brodeau, G Romier, B van Cutsem (eds.), Recent Developments in Statistics*, pages 133–145, 1977.

RÉSUMÉ

La quantification d'incertitudes dans les simulations numériques requiert de lancer un grand nombre de calculs. Pour certaines applications industrielles, la complexité des modèles physiques utilisés est incompatible avec le nombre de simulations requises. Accélérer la résolution d'équations décrivant un problème physique donné est essentiel dans l'optique de concevoir des systèmes fiables et sûrs. Dans cette thèse, nous proposons une nouvelle méthode combinant la réduction de modèle et l'apprentissage statistique pour prédire rapidement une solution approchée d'une équation aux dérivées partielles stochastique. Le comportement mécanique d'une pièce d'un turboréacteur a ainsi pu être calculé 636 fois plus vite qu'avec le modèle haute-fidélité d'origine avec moins de 3% d'erreur, ce qui a permis de quantifier les incertitudes engendrées par la thermique.

MOTS CLÉS

Apprentissage statistique, Réduction de modèle, Simulation numérique.

ABSTRACT

Uncertainty quantification in computational physics requires running many simulations. For some industrial applications, the complexity of the numerical model is incompatible with the number of simulations to be run. Solving physics equations in a reduced computation time is therefore essential for the design of safe and reliable systems. In this thesis, we propose a new numerical method combining model order reduction and machine learning to compute an approximate solution of a stochastic partial differential equation in a reasonable computation time. With this method, the mechanical behavior of an aircraft engine's component is predicted 636 times faster than with the original high-fidelity model with less than 3% of errors, which enables quantifying the uncertainties generated by the thermal loading.

KEYWORDS

Machine learning, Model order reduction, Numerical simulations.