POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Study of Physics-Informed Neural Networks to Solve Fluid-Structure Problems for Turbine-like Phenomena

GAÉTAN RAYNAUD
Département de génie mécanique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie aérospatial

Août 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Study of Physics-Informed Neural Networks to Solve Fluid-Structure Problems for Turbine-like Phenomena

présenté par **Gaétan RAYNAUD**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

David VIDAL, président
Frédérick GOSSELIN, membre et directeur de recherche
Sébastien HOUDE, membre et codirecteur de recherche
Bruno BLAIS, membre

ACKNOWLEDGEMENTS

Firstly I would like to thanks my advisors Frédérick Gosselin and Sébastien Houde for their continuous support along these two years. Both have provided me with insightful perspectives, ideas and have also guided me into the world of academic research. Back in 2019, I remember my first call with Frédérick between Vancouver and Palaiseau, describing his ideas for research projects as well as his experience of living in Montreal. I am grateful to him for giving me the chance to come, learn, work and teach here in Polytechnique Montréal in a great lab. I met Sébastien at my arrival and despite the distance from Québec city to Montréal, he has been very present with his wise advice, support and precise feedback. I really enjoyed the few moments spent in Québec City in December 2019 when I met the team at LAMH and I wish I had the time to exchange more with them.

I sincerely acknowledge the support from the Tr-FRANCIS project, from the Natural Sciences and Engineering Research Council of Canada (NSERC) and from the Institut Trottier de l'Énergie. I also would like to thank the Simulation-Based Engineering Program for the organization of several engaging conferences that helped me get some perspective on my work as well as meeting colleagues and researchers.

I have a thought for Professors David Vidal and Bruno Blais who have accepted to be part of my jury and to read my thesis. Getting into someone else's work takes time and I would like to thank them for that.

Two years across the ocean goes both amazingly quick and slow, especially during these complicated times and with all the social restrictions that came up. I have a special thought for my family, my parents, my brother Valentin and some of my dearest friends who stayed in Europe (and for some who had to cancel their trip to Canada). Despite the distance, their backing never wavered. I am also grateful to my friends in Montréal whom I met at the lab, the orchestra, at the AÉCSP or during hikes, games or over a drink, for all the moments shared and the pleasant company. They were a moral support and guided me through Poly, the courses, research at the office and the life in Montréal. Finally, a special thought for Isa who has brighten up these last months with smiles and happiness.

RÉSUMÉ

Les réseaux de neurones informés de la physique (PINNs) décrivent une grandeur physique comme une fonction continue dans l'espace et le temps, par exemple les champs de vitesse et de pression pour un écoulement fluide. Cette méthode sans maillage permet de résoudre non seulement des équations aux dérivées partielles à partir de conditions initiales et aux limites (des problèmes directs) mais également de retrouver des paramètres inconnus dans le modèle théorique à partir de données fournies au préalable (expériences, simulations) en même temps que se fait la reconstruction de la solution. Dans ce dernier cas, on parle de problèmes inverses et les PINNs sont capables de les résoudre de la même manière que des problèmes directs grâce aux réseaux de neurones qui minimisent efficacement les écart aux données fournies ainsi que les résidus des équations différentielles.

Ce mémoire présente les fondamentaux mathématiques des PINNs qui permettent de les prendre en main, notamment dans le cas de la mécanique des fluides. Nous avons par la suite développé une approche modale afin de résoudre des phénomènes vibratoires avec une plus grande efficacité que les PINNs classiques. Nous avons également montré que les PINNs sont robustes dans une certaine mesure lorsqu'ils sont confrontés à des mesures incomplètes, éparses ou bruitées et dans les différents exemples nous avons tenté de montrer comment cela pourrait être utile pour améliorer la qualité et la quantité des données dans le cadre de résultats expérimentaux. Enfin, nous avons exploré plusieurs pistes pour prendre en compte les vibrations dûes aux interactions fluide-structure, bien que les premiers résultats soient à ce jour non conclusifs.

ABSTRACT

Physics Informed Neural Networks (PINNs) encode a physical quantity as a continuous function in space and time, for example velocity and pressure fields for a fluid flow. This meshless method allows to solve not only partial differential equations (direct problems) but also to find unknown parameters in the theoretical model from provided data (experiments, simulations) while reconstructing the solution field. In the latter case, we are referring to inverse problems and PINNs are able to solve them in the same way as direct problems thanks to neural networks that efficiently minimize the distance from the provided data as well as the residuals of the differential equations.

This thesis presents the mathematical basis of PINNs in order to be able to use them especially in the context of fluid mechanics. We have then developed a modal approach to solve vibration effects with a higher efficiency than the classical PINN approach. We have also shown that PINNs are robust when confronted with incomplete, sparse or noisy simulated measurements. In the test cases along this thesis we put a stress on how PINNs could be useful to improve the quality and quantity of data from an experimental set-up. Finally, we have explored several avenues to take into account vibrations due to fluid-structure interactions, although the first results are so far non-conclusive.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF SYMBOLS AND ACRONYMS

List of acronyms

AI Artificial Intelligence

BC Boundary Conditions

CFD Computational Fluid Dynamics

CPU Central Processing Unit

FSI Fluid Structure Interactions

GPU Graphics Processing Unit

IC Initial Conditions

ML Machine Learning

MLP Multi Layer Perceptron

(A)NN (Artificial) Neural Network

NS Navier-Stokes (Equations)

ODE Ordinary Differential Equation

PINN Physics-Informed Neural Networks

PDE Partial Differential Equation

RAM Random Access Memory

VIV Vortex Induced Vibrations

List of symbols

$\mathbf{F} = (F_x, F_y), \mathbf{df} = (df_x, df_y)$	Total and local forces of the fluid on a given boundary.
$\mathcal{L}$	Loss function, usually computed from the output of a neural network.
$\mathcal{L}_m$	Part of the loss that is expressed as a mean square error between labelled data and the output of the NN.
$\mathcal{L}_{eq}$	Part of the loss that does not require labelled data and which is formulated as a mean square error of residuals of the PDE operator $\mathcal{N}$ computed with the output of the NN.
$\mathcal{N}$	Mathematical differential operator of a space and/or time function, possibly non-linear.
N_{in}, N_m, N_{bc}	Number of points chosen for the sampling of the domain V_{in}, boundary conditions V_{bc} or picked out of measurements data V_m.
$Re = \frac{UL}{\nu}$	Reynolds number: defines the ratio of the inertial forces over viscous forces. It is computed with the typical scales of velocity U, length L and the kinematic viscosity of the fluid ν.
$S_t = \frac{fL}{U}$	Strouhal number: defines the dimensionless frequency of vortex shedding f on a fixed cylinder.
$U_r = \frac{T_{\text{solid}}}{T_{\text{fluid}}}$	Reduced velocity: defines the ratio between time scales of solid displacement T_{solid} and convection time $T_{\text{fluid}} = L/U$.
V, V_{in}, V_m, V_{bc}	Sampling of input coordinate points where to compute the output of a PINN and the associated losses. They can be separated in a component for equations penalization, distance to fitting data or boundary condition.
W_i, W	Matrices used in the affine transformations inside the neural network. Usually called Weights.
b, b_i, $\mathbf{b}$	Vectors used in the affine transformations inside the neural networks. Usually called biases.
$\mathbf{n} = (n_x, n_y)$, $\mathbf{t} = (t_x, t_y)$	Normal and tangential vectors. By convention, $\mathbf{n}$ is oriented from the inside to the outside of the solid and $\mathbf{t}$ goes along the orientation of the border with respect to s.
p	Pressure field

q, q_m	Generic unknown variable to be solved, usually stands for either u, v or p. When associated with the subscript $_m$, it refers to data that are provided for fitting.
$\mathbf{u} = (u, v)$	2D velocity field with its x and y components. For a 3D flow, the z component is denoted with w.
s	Curvilinear abscissa that defines a parametric border.
$x_{BC}(s), y_{BC}(s)$	Coordinates of points on a given boundary, functions of the curvilinear abscissa s.
$x^0, x^t, \Omega^0, \Omega^t$	Superscript used in the case of variables or domains expressed in a fixed reference frame ^{0} or its associated frame at a given instant ^{t}.
$\Omega, \partial\Omega, \partial\Omega_{FS}$	Domain where the PDE is defined with its border denoted using the symbol ∂. When dealing with several types of borders, a subscript can be used to specify it (like FS for the interface between a fluid and a moving structure).
η	Learning rate of gradient-descent optimizer. Can designate more widely the hyper-parameters of an optimizer.
θ	Set of scalar parameters that defines a neural network. It is usually composed of matrices W and vectors b that are concatenated into a vector.
$\boldsymbol{\xi} = (\xi_x, \xi_y), \boldsymbol{\eta} = (\eta_x, \eta_y)$	Direct and inverse extension operators.
$\left[\frac{\partial u}{\partial x}\right]_y$	Partial derivative of u with respect to x when y is kept constant. This notation is used when there might be an ambiguity.

CHAPTER 1 INTRODUCTION

According to the energy regulator in Canada [5], production of electricity rests upon hydraulic turbines for approximately 95% in 2019 in Québec. Hydroelectric plants are low-emissions means of production. They are flexible in their use and, associated with water reservoirs, can store energy. This is of particular interest when coupled in an electricity grid with intermittent sources like photo-voltaic or wind turbines. However, this flexible use with repeated starts, stops and spin no-load conditions are far from the operating point these hydraulic turbines have been designed for. Consequently this may cause severe vibrations and damages that can result in a shortened life expectancy, frequent maintenance shut down and an increased risk of accidents [6].

In that case, research is thus necessary to better understand these new operating modes. The Tr-Francis project focuses on fluid-structure interactions (FSI) in medium head Francis turbine in start-up and speed-no-load operation. The project intends to identify and parametrize the damaging FSI phenomena in those conditions using measurements and simulations of a 14.4:1 model turbine representing a unit currently in production at Hydro-Québec. The model is currently installed on a hydraulic turbine test stand at the Hydraulic Machines Laboratory of Université Laval. It features a unique runner design to yield FSI homologous to the full-size production unit. This project is conducted with industrial manufacturers and operators from the sector of hydraulic turbines. One of the main motivation behind Tr-FRANCIS is that current FSI simulations approaches often fail to predict measured stress levels in runner.

Over the past few years, machine learning and more generally the field of artificial intelligence has appeared as a support of traditional approaches to deal with large amounts of data, especially for physical sciences [7–9]. In this master thesis, we looked at new numerical methods using the formalism of neural networks that would give the possibility to bridge the gap between experimental data and theoretical models. We are therefore interested in confined incompressible flows inspired by those that can be found in hydraulic turbines. One of the long term goal is to be able to accurately predict the vibrations of the turbine with both experimental and theoretical data. This objective covers the fields of numerical methods, including modal approaches for the vibrations and the consideration of fluid-structure interactions. These area are discussed in the following chapters under the scope of Physics-Informed Neural Networks (PINNs).

CHAPTER 2 LITERATURE REVIEW

In this literature review, the concept of direct and inverse problems is introduced in section 2.1 and the classical numerical methods to solve them are recalled, with a focus on the particularity of fluid-structure interactions. As these methods face some limitations, especially for inverse problems with optimization through shooting method, another approach using machine learning features emerged in the context of fluid mechanics and is presented in section 2.2, a solution of which leads to PINNs that are finally qualitatively introduced in section 2.3. The missing parts in the literature on PINNs are discussed in section 2.4. After a summary of the key points of the literature review in section 2.5, the objectives and the plan of the thesis are outlined in section 2.6.

2.1 Review of current numerical methods and some open challenges

Predicting accurate quantities of interest in a mechanical systems is usually achieved by experimentation or by numerical simulations. Numerical methods depend on the resolution of discretized partial differential equations. This approach consists in solving specific equations where initial and boundary conditions, fluid properties and geometry are known. We can add that the problem is mathematically well-posed as defined by Hadamard and recalled in chapter 4 of Wendt et al. [10]. It means that there exists a unique solution[1] and that its behaviour changes continuously with the initial conditions. Most of classical numerical methods, which are briefly recalled in sub-section 2.1.1, are well suited for this kind of problems.

Using partial differential equations in an engineering problem solving process often implies finding the parameters that yield specific solutions meeting global performance targets. A ratio lift over drag can be targeted for a wing profile, some efficiency through a given operating range can be maximized in hydraulic turbines. The goal is to determine the causes (geometry, boundary conditions, a parameter in the equations or a control strategy) that lead to some consequences (often expressed as performance indicator in an engineering solution). Mathematically, these are called inverse problems and may often be ill-posed (still in the sense of Hadamard) [12]. An overview of such problems can be found in the collection of pa-

[1] The existence of the solution depends not only on the boundary and initial condition but also on the mathematical nature of the equation itself. Especially, the incompressible Navier-Stokes equations are not proven (yet) to have solutions. This is one of the Millennium Prize Problems from Clay Mathematics Institute [11].

pers *Inverse Problems in Engineering Mechanics* [13]. There are very few cases where these situations can be formulated into the resolution of a forward system of equations that can be directly reversed. More often, this kind of inverse problems is formulated as an optimization process solved iteratively. A direct problem is solved for a sequence of test inputs and these inputs are tuned so that the results of numerical simulations are as close as possible to the given target. Other techniques such as adjoint-based optimization provides tools to obtain the sensibility of the optimization target with the input variables [14, 15].

Among these types of inverse problems, flow field reconstruction refers to the recovery of a complete solution described by a physical field like velocity, displacement, pressure, etc [16–18]. It has some interests especially for enhancing experimental data which are local (discrete locations in time and space), only partial (they do not cover all the regions in space) and potentially imperfect (noise, problems of calibration). Field reconstruction aims at extracting hidden information in the data, an example of which is the prediction of forces using distant measurements of velocity in a flow. This can also be used to correct data when a part of the information is missing or corrupted by noise, delays or imprecision regarding the position of the probe. This fits in the domain of data assimilation, illustrated in meteorology with the use of temperature probes and satellites cartography (which are too sparse to simply do interpolation) to recover information and then predict the weather for the next days [19, 20].

This information recovery is more general than the formulation of a direct problem since usually it is not formulated as a set of initial and boundary conditions. Field reconstruction is rather defined using sparse data sampled at given locations in the domain where the solution is defined and these data can be of different nature since they may combine concurrent information from sensors of force, velocity, pressure, displacement, heat, etc.

Flow reconstruction also leads the way to active control strategies [21]. Indeed in optimal control, the objective is to find the most efficient level of an actuator in order to optimize a target (formulated as a reward or a loss) given only an estimation of the state of the system through limited sensors. To predict the effect of the next actuation, it could be of interest to understand the effect of this actuator using the field reconstruction with the information provided by the limited set of sensors.

2.1.1 Overview of classical numerical methods

In computational fluid and solid dynamics, several methods are available to approximately solve the governing equations. We propose a short recap of the available current methods used, especially in the field of fluid dynamics. The content in this subsection is intended to be a concise summary of the main ideas based on the textbooks of Wendt [10], Allaire [22], Rao [23] and Moukalled [24]. We also point out some challenges that appear when coupling fluids with solid deformations in simulations.

If we consider an unsteady problem governed by a general possibly non-linear differential operator $\mathcal{N}_x$ of a function f that depends of time and one dimension in space x, the strong form of PDE can be written as

$$\frac{\partial f}{\partial t} = \mathcal{N}_x(f). \tag{2.1}$$

The differential operators can be approximated by replacing all derivatives by a finite difference (FD) when the limit of $dx \to 0$ is replaced by a small (though finite) value of Δx in $\frac{df}{dx}(x) = \lim_{dx\to 0} \frac{f(x+dx)-f(x)}{dx} \approx \frac{f(x+\Delta x)-f(x)}{\Delta x}$. Practically, by storing the values of f on a grid of points at a given time step, we construct a discretized approximation of $\mathbf{f}$ as a vector of values at each point location. Then we can transform $\frac{\partial}{\partial x}$ as a matrix operator and then compute $N(\mathbf{f})$ that approximates $\mathcal{N}_x(f)$ at each point location on the grid. Then, the integration in time can be performed using a wide variety of integration schemes that discretize the time dimension. For instance the explicit forward Euler formulation consists of:

$$\frac{\mathbf{f}(t_{n+1}) - \mathbf{f}(t_n)}{\Delta t} = N(\mathbf{f}(t_n)), \tag{2.2}$$

but is conditionally stable. More robust schemes exist but may require more computational resources [22].

Finite element method (FE) appeared in the middle of the 20^{th} century, formally introduced by Clough in 1960 [23, 25]. It consists in solving a PDE on cells, called elements, possibly unstructured which brings several simplification compared to finite different schemes that works best with structured mesh. Inside each cell, the solution is split into a sum of basis local functions. FE are therefore an application of Galerkin's methods. Thus, FE provides a continuous interpolation on the domain based on values at nodes in each cells. Besides, FE usually solves the weak form of PDE (instead of the strong form in FD). The system of

equations to be solved is obtained by the integral of PDEs on the whole domain weighted by test functions.

In the early 1970s, Finite volume method (FV) has been proposed to keep the unstructured approach of cells but to simplify it by carrying the computations in the physical space of system coordinates, as recalled in the textbook from Moukalled et al. [24]. From the beginning it was applied to flow problems [26]. Its specificity is to solve a system based on conservation equations (mass, momentum, energy...) on given volumes defined by the mesh, instead of the local equations with higher order derivatives.

Beyond these three methods, several others are found in the literature. Some methods do not explicitly use a discrete sampling of space and are called mesh-free. Among those, Vortex methods [27] and Smoothed Particle Hydrodynamics (SPH) [28] solve an equivalent dynamics but for particle motions. Lattice Boltzmann method (LBM) [29, 30] solves an equivalent problem for density of population functions. Also a wide variety of spectral methods use the principles of Galerkin formalism but with global basis functions that are not restrained to local elements. Hussaini et al. has provided a comprehensive review [31] which may be completed by a more recent textbook from Canuto et al. [32] .

2.1.2 Specific problems with fluid-structure interactions in CFD

The specific case of fluid-structure interactions provides its own set of challenges. Especially, the deformations of the solid boundary changes the fluid domain as well as the kinematic and dynamic equilibrium at the fluid-solid boundary. Based on the thesis from Hovnanian. [33], a review from Hou et al. [34] and a paper from Pfister et al. [35], several approaches stand to deal with this moving boundary. Among which these:

- Moving the solid boundary and re-meshing the fluid domain at each iteration (or at least regularly). This method can be quite expensive since meshing can be long, especially in 3D, and often need to be verified by a human expert. Besides an interpolation of the flow from one mesh to the following can introduce errors that lower the precision of the overall simulation.
- Using an **Arbitrary Langrangian-Eulerian** (ALE) method where an initial mesh for the fluid domain is deformed as an elastic solid to propagate the deformations of the fluid-solid boundary. An extension operator needs to be solved alongside the fluid and

solid quantities. This approach can be limited in the cases where the solid deformations are very large compared to the initial fluid domain size.

- Using a fixed mesh for the fluid domain but with an **Immersed Boundary Method** (IBM). Only the cells of the fluid domain that are outside the solid domain are activated and those that are both on fluid and solid domains are corrected with an added forcing term.

The two first solutions are part of the conforming mesh methods because of the tracking of the interface motion. On the other hand, IBM is a non-conforming mesh method.

Another distinction is related to the use of partitioned or monolithic solvers. Monolithic solvers compute both fluid and solid using one matrix that implicitly account for the interactions. Partitioned solvers use different matrix for the fluid and the solid domains that must be coupled explicitly using information-exchange algorithms.

2.2 Machine learning for fluid dynamics

Machine learning (ML) is a field of computer science that aims at building predictive models directly from experience without having being taught how to do so by a human expert [2]. As outlined in Figure 2.1, ML is a sub-field of Artificial Intelligence (AI) which objective is to design programmable objects that behaves like humans. Later ML has lead to a sub-class called Deep Learning, using mainly deeper Artificial Neural Networks (ANN) capable of more abstract operations in order to reduce the need for human data-labelling. Especially, the Multi-Layer Perceptrons (MLP) is a fundamental tool that is used in this thesis and explained in chapter 3 using mainly the textbook from Goodfellow et al. [2]. Wang et al. [36] also provides a historical point of view of modern deep learning techniques.

The use of ML skyrocketed in the past decade largely due to the availability of large amount of data and powerful computing units [2]. For example, consumer grade Graphics Processing Units (GPU) originally for screen display are now particularly well-adapted to scientific computing using powerful vector processors and high speed memory buses. GPU support in popular and open source ML libraries like TensorFlow [37] or PyTorch [38] made it accessible for a lot of developers, engineers and researchers. Since 1980s ML algorithms were used to achieve a number of remarkable success in several fields, among which computer vision [39,40], natural language processing [41] or reinforcement learning as proved by DeepMind [42] with

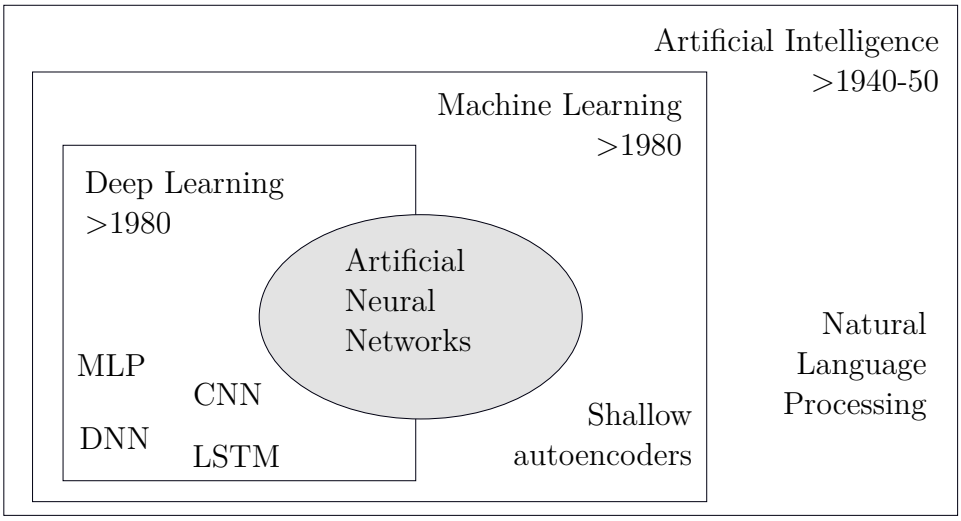


Figure 2.1 Classification of fields related to Machine Learning. Figure inspired from Fig. 1.4 in Goodfellow et al. [2].

a computer outperforming human on video games like Atari. These achievements lead to a renewed interest in this area of research (which means funding for academic and industrial research) and with the idea of applying such methods to more established domains like mechanical engineering.

At first sight, the way ML works may seem incompatible with traditional CFD. In classical CFD, the quantities of interest for engineering applications are predicted by numerical methods that solve an approximation of theoretical models. These models are often written as a set of partial differential equations derived from first principles. Navier-Stokes equations describe the conservation of mass, momentum and energy for continuum variables that depict the state of the flow. Conversely, ML typically uses directly data to infer the link between the parameter to be set (usually the input of the algorithm) and the quantity of interest (the output), without any knowledge of physical laws, as discussed by Brunton et al. [7] and Brenner et al. [8]. Consequently, the model built may not be as interpretable as the first principles where the physical meaning of each term in the fundamental equations can be explained by a human expert. This interpretability typically provides a greater level of confidence in the predictions as discussed by Beck et al. [43] in the context of turbulence modelling. Thus, one

challenge is to make sure that the predictions of ML algorithms are reliable and generalizable.

A second apparent incompatibility discussed by Brunton et al. [7] stands in the observation that ML algorithms can require extremely large amounts of data for training. Although it may be cheap to generate data for application like text translation, chess games or image recognition, it must be kept in mind that solving mechanical problems, for instance the complete solution in time and space of a turbulent flow, can be at best expensive and possibly unreachable. Despite these difficulties at first sight, there are some interesting leads to use machine learning in the field of mechanical engineering that are recalled in the next paragraphs.

To this day, machine learning in fluid dynamics has found applications for flow modelling, understanding some complex phenomena and reducing the complexity of models that describe it, to build active control strategies and for several optimization problems [7, 44]. Most of the proposed examples in the following paragraphs use the formalism of Artificial Neural Networks (ANN or simply NN). ANN are mathematical tools that can be defined as universal approximators. It maps an input vector to an output vector by constructing an explicit function constructed with tunable scalar coefficients. Then, all the scalar coefficients are optimized so that the output of the neural network minimizes a given loss function.

Numerical simulations are carried out using a large number of degrees of freedom, often exceeding millions for three dimensional turbulent flows. An idea is to simplify the detailed representation of the flow by finding patterns in the data. Among these techniques, modal decomposition are long-used methods to reduce the dimensions of the problem. Fourier decomposition is a way of splitting the time dependency of a periodic phenomena with a set of harmonic functions associated with coefficients that varies with space. This decomposition is orthogonal in time. Another modal method is the Proper Orthogonal Decomposition (POD) and uses orthogonal functions of space. It consists in decomposing any quantity $q(x,t)$ as a sum

$$q(x,t) = \sum_k a_k(t)\varphi_k(x) \tag{2.3}$$

with $\int_\Omega \varphi_j\varphi_k = \delta_{jk}$ where δ_{jk} is the Kronecker-delta function. The mode shapes φ_k can be obtained by solving an eigenvalue problem and then the time coefficients a_k are obtained by direct projection. The eigenvectors ordered by their eigenvalues are optimal in terms of energy. Taira et al. [45] present an overview of several types of modal analyses for fluid

flows including POD and its derived forms alongside other techniques like Dynamic Mode Decomposition (DMD).

It has been showed that POD is equivalent to an Auto-encoder which is a one-hidden layer neural network with linear activation functions trained to recover its own field [46]. A natural extension of this NN formalism of POD is to use non-linear activation functions and deeper neural network to map all the degrees of freedom of a flow as recalled by Brunton et al. [7] and illustrated by Milano et al. [47] where they use deep auto encoders to perform flow reconstruction of near wall turbulent flows with a gain in data compression.

POD, DMD and other types of modal decomposition find applications for flows that depicts periodicity [45] because this offers efficient representations for vibration phenomena. Especially within the framework of the Tr-Francis project, taking into account the flow-induced vibrations in hydroelectric machinery is critical during the phases of design and operation as reviewed by Dörfler et al. [48] and more generally by Blevins [49]. This approach has already been used (outside of the machine learning field) for understanding dynamic behaviours in Francis turbines [50, 51] and can suit both periodic and transient phenomena. As Machine Learning aims at finding patterns in the data, there is a common basis between modal representations and Machine Learning techniques.

Several engineering problems involve finding the governing equations for an unknown dynamics. These can have different objectives: deriving reduced order models, finding the coefficients or the physical constants from a set of raw data, or being able to extrapolate in time. Physics-Informed Neural Networks (PINNs) are a possible way of doing such things and are introduced in section 2.3 of this thesis. But other tools can perform comparable tasks within a different formalism. For instance, the Sparse Identification of Nonlinear Dynamics (SINDy) framework formalized within S. Brunton's team [52] makes it possible to find governing equations of an unknown phenomena. It consists in solving as sparsely as possible the fitting of a matrix of coefficients that links the time derivative of variables with linear, quadratic and a set of other nonlinear and coupled terms. Fukami et al [53] mixed this approach with the deep auto-encoders to find the time evolution of two latent variables that stands for the coefficient of deep-NN modes. They applied this to the two dimensional flow over a cylinder, including transient regime.

Another application of flow modelling is to be found in closure problems. Considering that

Direct Numerical Simulations (DNS) of the Navier-Stokes equations is prohibitively expensive for large Reynolds number flows, as discussed in chapters 8 and 9 from Pope [54], simplifications were developed to tackle complex flow on available computing resources. For instance, in the Reynolds Averaged Navier-Stokes (RANS) equations, one does only solve the statistical average velocity and pressure fields. All the terms of the RANS equation depend on the averaged flow quantities expect one, the Reynolds stresses tensors which represent the effect of the turbulent fluctuations on the averaged flow. The Reynolds stresses lead to a closure problem that require the use of semi-empirical models, the turbulence models. Typically, turbulence models are derived from analytical consideration and limited experimental measurements of flow related turbulent quantities. As such, their scope is often limited to certain classes of turbulent flows [55]. Using already computed turbulent flows with DNS, Machine Learning have powerful tools to generate data-driven turbulence closure models that could help improve RANS computations. Duraisamy et al. [56] address this subject and, for example, Font et al. [57] used a convolutional neural network to perform the closure in Spanwise Averaged Navier-Stokes equations (SANS) leading to estimations of forces on bluff-bodies with a low computational cost and a high precision. Besides the pre-trained CNN have been used for other geometries of bluff bodies as well as other Reynolds number.

In a design process involving CFD, being able to interact with the shape of an object and obtaining quickly the response of the flow around it might be of prime interest. Instead of using a classical CFD solver for each iteration of the design, Guo et al. have proposed a convolutional neural network that, once trained, perform a quick prediction of the steady flow two orders of magnitude faster than a CFD code in their test case [58]. This make it possible to test a variety of designs much larger in a reasonable time, taking into account that every engineering design process have to deal with compromises between the time spent during the study to optimize the solution to get a better result, and the need to finish the design in a given time.

Flow control and optimization have found new tools using artificial neural networks and especially Deep Reinforcement Learning (DRL) with control strategies that can perform better than more conventional optimal control strategies [7]. It is possible to train an agent given an estimation of the flow state to perform some actions, such as control tasks to maximize a reward. This technique have been applied to reduce the drag of a flow over a cylinder using two jets on the two sides of the cylinder, reducing by a factor 20 the force oscillations [59]. An other application within the FSI field appears in the work of Verma et al. [60] where a simulated fish learnt how to deform its body to swim the most efficiently in the wake of

another fish so that it harnesses the most energy from the vortical structures. Mathematical background and a review of literature on the use of deep reinforcement learning for fluid dynamics have been written by Garnier et al. [61].

The use of ML tools alongside classical methods and for applications for fluid and solid mechanics is still an open challenge. The examples provided in this section unveils quite promising leads. However there are still some concerns about the ability of ML tools to be reliable and generalized so that their use in the field of mechanical engineering can lead to certification for use in the process of design and predictive maintenance. In that context, the use of PINNs as presented in the following sections of this thesis could address these problems by taking into account the knowledge from data as well as physical laws.

2.3 Physics Informed Neural Networks

The principle of Physics-Informed Neural Networks (PINNs) is to use NN as an approximator of the function that solves a partial differential equation. This comes from a mathematical property presented and proved by Hornik et al. [62] in 1989 that we can summarized by quoting the abstract :

> *"Standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators."*
>
> —Hornik et al., [62]

In more simple terms, this means that a neural network $NN(\cdot)$ can be built so that when fed with an input vector $\mathbf{x}$, the output $\mathbf{y} = NN(\mathbf{x})$ can be as close as desired to any values $f(\mathbf{x})$ provided by a function $f : X \to Y$ with a given norm[2] $\|\cdot\|$. The function needs to be Borel measurable[3] which in practice is verified for classical physical applications such as

[2] A given norm which is not specified since all norms are equivalent in finite dimension.

[3] This means that for any underlying set E_y in the σ-algebra of Y (the collection of subsets of Y containing Y itself) that is measurable with a Borel measure, its preimage $f^{-1}(E_y)$ is also measurable with a Borel measure. Borel measure are mathematical measures that are defined on all the open sets.

fluid dynamics[4]. Then the second important point in this quote is that the degree of accuracy reachable for the approximation of f is limited by the size of the NN once a non linear activation function is given. At that point, there is no clue given on the required size nor the optimal ratio between depth and width of neural networks for obtaining the targeted precision. Nor does the method to obtain the coefficients, by contrast to methods where you can do direct projections on a set of orthogonal basis.

In the mid 1990s, Dissanayake et al. [63] followed a few years later by Lagaris et al. [64] proposed to use that property of neural networks to solve ordinary and partial differential equations. Their approach is focused on the direct problem (solving a PDE given a domain, a forcing and boundary conditions). These authors insist on some key-points to demonstrate the interest of PINNs: especially the fact that a NN can build a function that is close in form to an analytical solution and may require far fewer degrees of freedoms than classical FE simulation. Besides, the natural differentiability of the NN was already an asset compared to classical numerical methods in the 1990s with limited differentiability. Their approach already used back-propagation to estimate the residuals of PDE on a set of penalization points that discretises the domain (or collocation points as they named them). With an example on a two dimensional Poisson equation with a forcing term, Lagaris et al. [64] showed that a one hidden layer NN performed several orders of magnitude more accurate predictions than a FE simulation at the same number of degrees of freedoms (which is the number of parameters to tune in the model) and for an equivalent or lower computation time (see Figures 16 and 17 in [64]).

A renewed interest for this method came back with the work of Raissi, Perdikaris and Karniadakis in 2018 [65]. The basic PINN formulation is recalled with clear examples and with leads for data-driven studies in the paper *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations* published in 2019 [66]. The novel approach consists in using PINNs not only for direct solving of PDEs which already existed but dit not get a lot of success, but for inverse problems.

To go further than the examples provided by Raissi et al., several authors quickly applied PINNs to other fields of physics and mechanical engineering. Models for non-Newtonian fluids [67], for Reynolds-stresses in turbulent Couette flows [68] or for subsurface flows [69] have

[4]The functions at least piecewise continuous are Borel measurable which includes functions with a finite number of discontinuities as observed in hydraulic jump or shock waves.

been found using PINNs. The use of PINNs for unknown dynamics and constitutive relations is also discussed by Tipireddy et al. [70]. Flow reconstruction is addressed in some specific cases like high-speed flows [71] or vortex induced vibrations of a cylinder [72]. In the field of solid mechanics, similar approaches have been presented: reconstruction of deformation and stress fields [73] with parameter identification [74], modelling of brittle fracture with the phase field formalism [75], on the use of wave propagation for seismic data [76] or for a non intrusive crack detection method [77]. In other domains than strictly mechanical engineering, PINNs have been applied to solve wave guide problems in electromagnetism [78] and for nano-optic and meta materials [79]. Several promising outlooks for PINNs are pictured in the review papers by Cai et al. [80] and Karniadakis et al. [81].

More technical developments have been made to understand the difficulties of convergence of PINNs and give some leads to improve their robustness. Among them, Wang et al. use an approach to dynamically adapt the weights of the loss function while training PINNs to address convergence failures [82]. Almost the same research group proposed a method to reduce some gradient pathologies by adapting the architecture of PINNs [83]. In a slightly different way, Jagtap et al. [84] used adaptive activation functions that seem to improve the convergence rate at the beginning of the training. Finally some papers focus on the mathematical properties of PINNs as a preparatory work for further improvements [85, 86].

2.4 Gap in the current scientific literature on PINNs

The drawback of that recent profusion of scientific papers on PINNs is that these articles are more about applying PINNs to solve new types of physical problems rather than to understand the effect of the NN parameters and the training strategies to increase the robustness. Yet, PINNs are succinct in their mathematical construction, but compared to other classical numerical methods, they rely on a larger number of hyper-parameters that affect the training and accuracy of the solution. Moreover, some of these hyper-parameters have less physical interpretation than their equivalent in classical numerical methods which induces greater difficulties when it comes to choosing and tuning them. For instance, the size of an NN that defines a PINN can be linked to the number of nodes in a mesh. However there is no known convergence rate for PINNs with the size of an NN as there is with the size of mesh elements which causes discretisation errors. In most of the papers listed earlier, it is complicated to reproduce the results if any of the training data or the code is not provided because most of the papers do not present all these technical details. In addition, there is

no evidence that these convergence properties and the strategies to set the hyper-parameters could be independent of the considered test case and thus be generalized. The absence of any textbook providing proper definitions and advices on how to set the hyper-parameters on PINNs is typical of a technique that is not mature (yet). And during the redaction of this thesis, the first reviews on PINNs [80, 81] were not yet peer-reviewed or even published as preprints. To address this issue, we gathered in chapter 2 several methodological aspects in order to clarify our approach and help a new reader understand how PINNs work. These methodological aspects were obtained by both trials and errors and synthesis of good practices found in codes from the literature. This chapter 2 tries to answer this (non-exhaustive) list of questions:

- How to chose the size of the NN and the activation functions specifically in the context of PINNs?
- What is the minimum number of data points? How can we classify the types of problems to be solved given the structure of fitting data?
- How to sample points to penalize the equations?
- What is the influence of the training strategy?
- How do unknown parameters in the equation affect the training?
- How does the quality of the data, and especially the uncertainty regarding noises, can affect the accuracy of the predictions performed by PINNs?

This last point is guided by the will to perform flow reconstruction using experimental data with a precision limited by the accuracy of the sensors. On the contrary, many articles demonstrate results with dense synthetic data. An intermediate path would be to simulate the presence of imperfections in the data by adding uncertainty and noise and to quantify the link between the level of this uncertainty and the accuracy of the predictions. In addition, as experimental data is often local and sparse, the expected change in performance of PINNs with sparse fitting data compared to dense data is important for future industrial applications. However this area is still under-explored (at the time this thesis is written). This has motivated the format of some results presented in chapters 3 and 4 as well as a discussion and clarifications on the different types of fitting data in chapter 2.

Moreover, to our knowledge, there is no method to address modal decomposition and general fluid-structure vibrations within the framework of PINNs. These are however key components to perform efficient predictions of vibrations in hydraulic turbines. This gap in the literature has guided the developments presented in chapters 3 and 4.

2.5 Synthesis

Designing an engineering solution involves solving several inverse problems in order to determine the best input parameter for a specific goal. With classical numerical methods, it consists in solving one direct problem for several sets of inputs and comparing the obtained results to the target. The loop for converging the inputs can be expensive in computational resources. However PINNs deal with inverse problems more intrinsically because the flow field is set to verify concurrently the equations as well as a large type of targets, including fitting data. PINNs are based on ML techniques which are high dimensional and non convex optimization techniques well suited to large data. PINNs are though slightly apart in the family of ML because they can work in the small data regime and they offer a measure of accuracy with the convergence of PDE residuals. Though, they are not very mature yet, especially the convergence properties and the robustness is still to be demonstrated and methods are missing in order to address modal decomposition for FSI vibrations.

2.6 Objectives and plan of the thesis

The goal of this master thesis is to explore the use of PINNs to solve FSI problems using a modal representation. The specific objectives to achieve this goal are:

- Handling a simple PINN code to solve basic incompressible flow fields using physical laws and exterior data (from analytical solutions, previous numerical simulations or experimentation).
- Study the influence of a PINN's parameters and its training method on the accuracy and time required for computation.
- Testing the ability of PINNs to deal with sparse and imperfect data.
- Looking for adaptations of the PINN architecture to take into considerations fluid-structure interactions as well as vibrations in modal space.

Taking into account the relative youth of PINNs and the lack of maturity, we focused on simple test-cases consisting of two dimensional flows that are described further.

The structure of this thesis is the following:

1. Chapter 3 provides an insight of the fundamentals of PINNs for flow problems. We investigate the mathematical formulation of Multi Layer Perceptrons and their extension to PINNs as well as some techniques related to the training of PINNs that are useful to understand and improve the convergence.

2. Chapter 4 describes the enforcement of a modal representation directly into the PINN structure to take into account vibrations more efficiently. This part, alongside studies on the handling of noisy and out of synchronization data is wrapped up in a paper which is the main part of chapter 4. In unpublished appendices, we examined other modal approaches as well as two slightly different test cases that illustrates how PINN can apply to less trivial geometries.

3. In chapter 5, some ideas on how to adapt a PINN to the difficulties of fluid-structure interactions are listed. This chapter provides several hints and early results based on the idea of extension operator. These approaches did not deliver positive results yet but could be the starting point of future research works.

CHAPTER 3 FUNDAMENTALS OF PINNS FOR FLUID DYNAMICS

3.1 Mathematical foundations of Multi Layer Perceptrons (MLP)

Physics-Informed Neural Networks use the formalism of Deep Neural Networks with dense connections between the layers. Here we recall the mathematical construction of a multi-layer perceptron (MLP) and how it is used in the context of Physics-Informed Neural Networks. For more details on MLP and PINNs, one can refer to Goodfellow et al. [2] and Raissi et al. [72] respectively and this chapter synthesizes several notions from these references.

3.1.1 Structure of an MLP

An MLP is defined by a sequence of symbolic operations that transform an input quantity x into an output quantity y. Its first goal is to approximate any given function $y = f(x)$. It aims at solving the limitations of basic approximations techniques like linear models $y = ax + b$, as depicted in Figure 3.1a, by introducing a non linear function. Inspired by the behaviour of the human brain, a threshold can be used to take into account some non linear effects. With the threshold function $\sigma(x) = \max(0, x)$, we can introduce an intermediate variable $z = ax + b$ and then compute $y = c\sigma(z) + b$. The equivalent graph of operation would be plotted at Figure 3.1b. Several intermediate variables can be used in the meantime and the output y can be expressed as a sum of nonlinear terms of the input x

$$y(x) = c_1\sigma(a_1x + b_1) + c_2\sigma(a_2x + b_2) + d, \tag{3.1}$$

as in Figure 3.1c. This model is in fact a simple version of a basic MLP.

In these examples, respectively 2, 4 and 7 scalar coefficients need to be tuned so that the model approximates as accurately as possible the provided data. We will simply denote these coefficients as an equivalent unknown vector θ and the constructed model as a function $y^{MLP}(\cdot, \theta)$. One way of finding the parameters of the model for the illustrated case is to use an optimization algorithm that minimizes a loss function $\mathcal{L}$. In the previous example, the loss function used is defined as a Mean Square Error (MSE) :

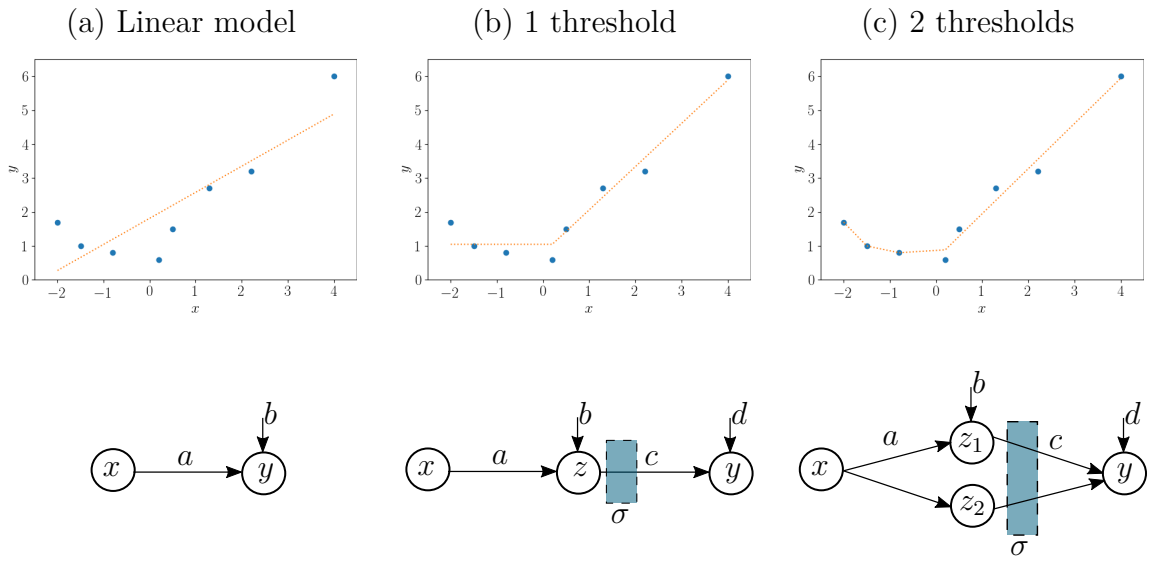


Figure 3.1 Illustration of the use of a MLP to approximate and fit any given data points

$$\mathcal{L} = \frac{1}{N}\sum_{k=1}^{N}\left(y^{MLP}(x_{data};\theta) - y_{data}\right)^2, \tag{3.2}$$

which computes the L^2 squared distance between the prediction y^{MLP} and the data y_{data}. Several techniques exist in order to minimize this loss. These often consists in computing the derivative of the loss with respect with the optimized parameters $\frac{\partial \mathcal{L}}{\partial \theta}$ and then performing gradient-based minimization. These techniques are discussed into further details in the next sub-section of this chapter.

Using the graph representation (as in the second line of Figure 3.1) helps picturing the structure of the mathematical operations that are carried out to compute the output of the MLP. If it is possible to write explicitly all the terms in the model in equation 3.1 for Figure 3.1.b or c, it starts to be quickly unreadable as the number of intermediate variables increases or when other layers are added.

This non-linear mapping presented in equation 3.1 can be generalized with more degrees of freedom which can lead potentially to a better accuracy. There are two possibilities :

1. increasing the number of intermediate variables. In equation 3.1, only two intermediate variables (z_1 and z_2) are used, but this number can be chosen arbitrarily. The more

intermediate coefficients (or artificial **neurons**), the **larger** is the neural network.

2. using the first set of intermediate variable (a **layer**) as an input to a similar transformation consisting of an affine transformation and non linear function σ of each coefficient. This increases the number of layers. The more layers a MLP has, the **deeper** it is.

We can define more generally an MLP, or a feed forward neural network, as a sequence of, alternatively, affine transformation and activation functions that transforms an input vector (or input layer) to an output vector (or output layer). This succession of affine transformation can also be understood as a way of shifting and scaling every quantities before applying σ which acts as a threshold. Mathematically, if we take the input $x \in \mathbb{R}^{n_0}$ and the output $y \in \mathbb{R}^{n_{p+1}}$ we can build a p-hidden layers deep neural network with a series of layers $z_k \in \mathbb{R}^{n_k}$ with :

$$\begin{aligned} z_1 &= \sigma\left(W_1 x + b_1\right) \in \mathbb{R}^{n_1} \\ &\vdots \\ z_{k+1} &= \sigma\left(W_{k+1} z_k + b_{k+1}\right) \in \mathbb{R}^{n_{k+1}} \\ &\vdots \\ y &= W_{p+1} z_p + b_{p+1} \in \mathbb{R}^{n_{p+1}}. \end{aligned} \tag{3.3}$$

In that example, the last transformation is linear so that y can be scaled without any restrictions from the range of the activation functions. Every matrices $W_k \in \mathbb{R}^{n_k \times n_{k-1}}$ and vectors $b_k \in \mathbb{R}^{n_k}$ are called **weights** and **biases** respectively. They are composed of coefficients that need to be initialized and then optimized so that the approximated outputs minimize the loss function. For the sake of brevity, these are referred to θ which is a vector containing all the variables in $(W_0, b_0, ..., W_{p+1}, b_{p+1})$.

Usually a neural network is presented with its general properties (or **hyper-parameters**). These contain its structure (number of layers and number of neurons per layers summarized by a list $[n_0, n_1, ..., n_{p+1}]$) and the list of activation functions σ. These functions can be different between each layers. In this thesis, all the presented NN use the same activation function for each layer. Nonetheless it can be set otherwise and according to a few tests that we performed, no significant improvements were noticed and this has not been studied in more details afterwards.

At this point it can be noted that there is a link between Galerkin's method and a one hidden

layer neural network. Indeed in Galerkin's methods, a continuous problem is discretized by decomposing the variable y on a set of basis functions ϕ_k so that $y = \sum_k a_k \phi_k(x)$. Then a loss is obtained usually by injecting this decomposition into a PDE operator $\mathcal{N}$ often written in its weak form and then projecting it against the set of functions and integrating in order to get a system of equations to solve:

$$\int_\Omega \mathcal{N}(\sum_k a_k \phi_k(x))\phi_j(x)dx \rightarrow 0 \forall j.$$

For a linearized system, this can be solved using direct or iterative solvers. But the coefficients a_k can also be determined using optimization techniques (which also fits in the sets of iterative solvers). In a one hidden layer NN, the decomposition is very similar excepted that the set of functions itself is tunable : $y = \sum_k a_k \sigma(w_k x + b_k)$. This limits the possibility of creating a set of linear system of equations but it is still possible to solve these parameters using optimization techniques. Besides, it can allow a more compact representation of a given function since some degrees of freedom can be "saved" by choosing where to "refine" using the first affine transformation that shifts and scale the equivalent ϕ_k. This could be compared in terms of efficiency to an unstructured adaptive mesh for every direction in space and time.

In optimization techniques, and especially those based on gradients, a starting point is usually required. For an NN, model's parameters θ must be initialized. For gradient vanishing issues, it appeared to be a bad solution to initialize them to zero. Finding the best initialization strategy is not an easy task since it depends on each problem to approximate, to the structure of the NN and its activation functions. Some optimizers also use stochastic methods so a given initialization can lead to different trained NN parameters. As recalled in chapter 8.4 from Goodfellow et al. [2], an important property of the initialization is to break the symmetry between two neurons in the same layer so that they can be updated differently using a deterministic optimizer. In the end, a common practice through the ML field is to use an initialization based on a given random distribution, often a Gaussian sampling. One difficulty is to scale the deviation of the Gaussian law. In the following sections of this thesis we used Xavier initialization from the PINN Python codes of several papers [66, 72]. It consists in scaling a truncated normal law centred on zero with a standard deviation per layer equal to $\sqrt{2/(\text{Input dim.} + \text{Output dim.})}$. Moreover it discards the values that are greater or lower than two standard deviation to overcome problems of saturation with activation functions like hyperbolic tangents at large values. This method has some similarities with some contents detailed in chapter 8.4 in Goodfellow et al. [2]. Especially a strategy from Glorot et al. [87] prescribes initializing the weights with a uniform law that has a radius

of $\sqrt{6/(\text{Input dim. + Output dim.})}$. In addition, biases vectors are often initialized at zero since they do not affect the propagation of gradients in the same way.

3.1.2 Loss construction, training and validation

In an MLP, and more specifically in supervised learning (when we know the function to be approximated), the goal of the NN training is to approximate as precisely as possible a given set of labelled data. If we have a set of training data x_{data}, y_{data} of size N_{data} to train an MLP $y^{MLP}(x;\theta)$, where x is the input and θ its parameters, a quadratic loss can be written

$$\mathcal{L} = \frac{1}{N_{data}} \sum_{k=1}^{N_{data}} \left[y^{MLP}(x_{data}, \theta) - y_{data} \right]^2, \tag{3.4}$$

and this loss is to be minimized by tuning the vector θ.

Using this simple mathematical operations (difference, square, average) allows to keep the symbolic graph of operations and, most importantly the automatic differentiation. This means that one can obtain the exact derivative of any quantity in the graph with respect to another that is linked to it. This is possible thanks to the implementation of back-propagation in modern ML libraries. Back propagation consists of using the chain rule for derivation $(f(g))' = g' \times f'(g)$ to compute $\frac{\partial \mathcal{L}}{\partial \theta}$ from the parameters at the last layer to those at the first layer (which explains the "back" in back-propagation in opposition to the usual output that is feed-forward the set of layers). If we use the same general formalism as in equation 3.3, it is possible to obtain the sensitivity of the loss with respect to one coefficient W_k^i of the weights matrix W_k by doing the chain rule. From the loss function to the output layer, the derivation propagates as:

$$\frac{\partial \mathcal{L}}{\partial W_k^i} = \frac{1}{N_{data}} \sum_{k=1}^{N_{data}} 2 \left[y^{MLP}(x_{data};\theta) - y_{data} \right] \times \frac{\partial y^{MLP}}{\partial W_k^i}(x_{data};\theta). \tag{3.5}$$

Then from the output of the MLP to the previous layer:

$$\frac{\partial y^{MLP}}{\partial W_k^i}(x_{data};\theta) = W_{p+1} \frac{\partial z_p}{\partial W_k^i}, \tag{3.6}$$

since the last layer is linear. Then from any hidden layer to the previous one, the general relationship writes as:

$$\frac{\partial z_p}{\partial W_k^i} = \frac{\partial}{\partial W_k^i}\left[\sigma\left(W_p z_{p-1} + b_p\right)\right] = W_p \frac{\partial z_{p-1}}{\partial W_k^i} \times \sigma'\left(W_p z_{p-1} + b_p\right). \tag{3.7}$$

This relationship is propagated backward up to the layer k where the wanted coefficient belongs:

$$\frac{\partial z_k}{\partial W_k^i} = \sigma'\left(W_k z_{k-1} + b_k\right). \tag{3.8}$$

The same algorithm is carried out for any component of a biases vector b_k^i. After doing this for every coefficient in the weights and biases, one can obtain the vector of sensitivity $\frac{\partial \mathcal{L}}{\partial \theta}$ which is used for gradient-descent optimization.

Gradient-based optimization is an iterative process that is quite efficient when the number of parameters is high and when derivatives are available. In NN those two criteria are verified since the length of θ can easily exceed $10^3 - 10^4$ and gradient free techniques that consists in trial and error can have a prohibitive cost: for a uniform mapping of the parameters space, the number of evaluation of the loss scales exponentially with the number of parameters. Gradient-based optimization consists in computing a sequence of parameters $(\theta_i)_{i=0,1,...,N}$ starting from the initialized value θ_0 and using a recurrent scheme $\theta_{n+1} = f(\theta_n)$. At the first order:

$$\mathcal{L}(\theta_{n+1}) = \mathcal{L}(\theta_n) + \left(\theta_{n+1} - \theta_n\right)^T \frac{\partial \mathcal{L}(\theta_n)}{\partial \theta}. \tag{3.9}$$

It can be noted that if $\mathcal{L}(\theta_{n+1}) < \mathcal{L}(\theta_n)$, which is what is wanted in order to reduce the loss, the scalar product $\left(\theta_{n+1} - \theta_n\right)^T . \frac{\partial \mathcal{L}(\theta_n)}{\partial \theta}$ needs to be negative so that the direction of evolution in the parameter space is qualitatively in the opposite direction than the one of the gradient $\partial \mathcal{L}/\partial \theta$. We introduce η that quantifies the length of the step in the direction $-\partial \mathcal{L}/\partial \theta$ so that the basic gradient descent can be written as:

$$\theta_{n+1} = \theta_n - \eta \frac{\partial \mathcal{L}}{\partial \theta}(\theta_n). \tag{3.10}$$

This step size η is often referred as a **learning-rate** in the field of ML. As explained by Allaire [22] for convex problems, an optimal step can be found at each iteration by solving $\min_{\eta \in \mathbb{R}} \mathcal{L}\left[\theta_n - \eta \frac{\partial \mathcal{L}}{\partial \theta}(\theta_n)\right]$. This is also called steepest descent and it is illustrated in Figure 3.2c from Goodfellow et al. [2]. However for simplicity or because for issues related to the non-convex properties as in NN, this second optimization of the learning rate is not carried

out and instead, a given learning rate is used. This learning rate can be fixed all along the optimization. But it is expected to do smaller steps once a minima is reached than at the beginning even if $\partial\mathcal{L}/\partial\theta$ has low values near a local-minima.

The relation written in equation 3.10 is the basis of Stochastic Gradient Descent (SGD) where the parameters are updated using a sequence of learning rate $(\eta_k)_{k=0,1,...,n,...}$ instead of a fixed η. In SGD, only a mini-batch (a random selection of data points in (x_{data}, y_{data})) is used to compute the gradient of the loss at each iteration so that it adds some uncertainty. Moreover the computational cost associated with one optimization step can be reduced and be independent of the size of the training set.

Several algorithms provide some improvements to deal with issues related with large oscillations in the parameters space as illustrated in Figure 3.2a. Among others, momentum algorithm [88] introduces a velocity term that smooths the direction of descent. In Figure 3.2b, $\partial\mathcal{L}/\partial\theta$ is depicted with black arrows and the inertia effect of the momentum allows a faster convergence. Other optimizers use adaptive learning-rate decay based on SGD or momentum formulation. Among them :

- the AdaGrad algorithm [89] nearly uses a decay based on the inverse of the accumulated gradient norm : $\eta_n = \frac{\epsilon}{\delta+\sqrt{r_n}}$ where ϵ is the global learning rate, δ a small value ($\sim 1 \times 10^{-7}$) and r_n is the accumulated gradient squared-norm : $r_n = \sum_{k=1}^{n} \left|\frac{\partial\mathcal{L}}{\partial\theta}(\theta_k)\right|^2$
- RMSProp [90] is similar to AdaGrad but uses a geometric decay ρ (which needs to be fixed by the user as well) for $r_{n+1} = \rho r_n + (1-\rho)\left|\frac{\partial\mathcal{L}}{\partial\theta_n}\right|^2$
- Adam [91] is an adaptive momentum algorithm with a similar policy than RMSProp but uses also a correction of bias.

These algorithms are explained into further details by Goodfellow et al. [2] alongside technical pieces of advice on how to use them efficiently. However at that time, there is no consensus on which of these algorithms works best. It may be specifically related to the problem that is adressed and also to the ability of the user to chose these hyper-parameters $(\eta_0, \rho, \epsilon, ...)$.

A different family of optimizers are approximated second-order methods. If we write the second order approximation of the new loss function

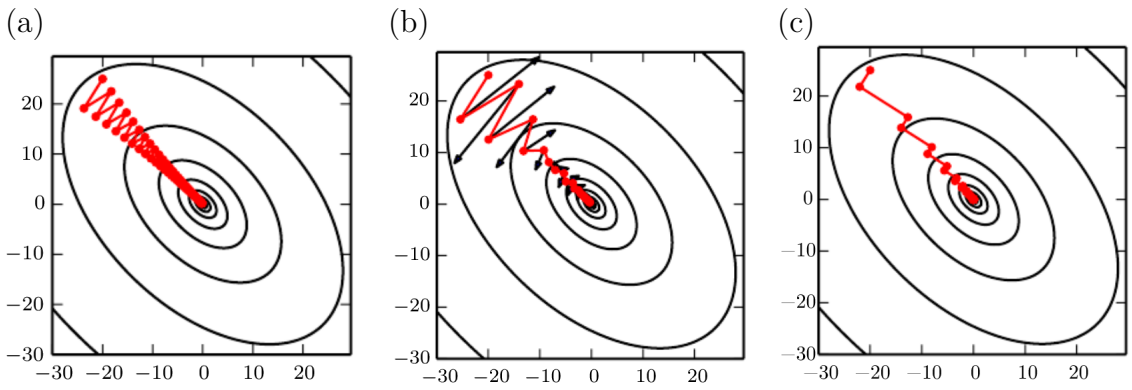


Figure 3.2 Illustration of some common optimization problems when the hessian matrix is poorly conditioned. Here contours of a quadratic loss are plotted in a 2D space of parameters using direct gradient descent (a), momentum with the direction of the gradient depicted with black arrows (b) and the steepest descent (c). Figures are reproduced from chapter 4 and 8 of Goodfellow et al. [2]

$$\mathcal{L}(\theta_{n+1}) = \mathcal{L}(\theta_n) + (\theta_{n+1} - \theta_n)^T \frac{\partial \mathcal{L}(\theta_n)}{\partial \theta} + \frac{1}{2} (\theta_{n+1} - \theta_n)^T \left[\frac{\partial^2 \mathcal{L}(\theta_n)}{\partial \theta_i \partial \theta_j} \right]_{i,j} (\theta_{n+1} - \theta_n), \quad (3.11)$$

the Hessian matrix $H = \left[\frac{\partial^2 \mathcal{L}(\theta_n)}{\partial \theta_i \partial \theta_j} \right]_{i,j}$ appears. Supposing that we reach the minima at the next step, we can make the assumption that $\frac{\partial \mathcal{L}(\theta_{n+1})}{\partial \theta} = 0 = \frac{\partial \mathcal{L}(\theta_n)}{\partial \theta} + (\theta_{n+1} - \theta_n)^T H$ at the first order. If we are able to compute the inverse matrix of the Hessian, then we can get:

$$\theta_{n+1} = H^{-1} \frac{\partial \mathcal{L}(\theta_n)}{\partial \theta}. \quad (3.12)$$

In an exact quadratic problem, this method would converge exactly in one iteration. However it might still work for non quadratic and even sometimes in non convex optimization. Though, the problem is that computing and storing this matrix is expensive (for a number $N \sim 10^3 - 10^4$ of coefficients in θ, computing H requires N^2 second order derivatives). And even more, finding the inverse is usually an operation which complexity is in N^3 (with Gaussian elimination for instance) which is even more expensive and which should be done at every iteration. In practical applications, NN can not afford these computations. Nonetheless, there are methods that approximates the Hessian and its inverse using only first order derivatives. Among them, conjugate gradient and Broyden-Fletcher-Goldfarb-Shanno (BFGS) are two methods that avoid computing the inverse of the Hessian by using iteratively equivalent quantities. BFGS in particular uses a sequence of matrices that are refined to approximate

directly H^{-1}. In this thesis, a variant called L-BFGS (L- for Low memory) is used and thanks to some assumptions on the initial guess of the approximated inverse, it helps reducing the number of coefficients to store.

The succession of operation when creating and training a Neural Network can be summarized in the algorithm 1. After the steps of construction of the model and the loss, preparation of training data and initialization of θ_0 that were discussed previously in this section, the training loop begins. It consists in applying successively the optimizer strategy to refine the parameters θ. A criteria must be given for stopping the training. It can be a limit of time, a threshold value for the loss function or based on the fact that the loss stagnates.

Another detail which is not discussed here is the possibility to split the training into batches so that the optimization is performed with smaller data set. Also, a part of the data should be kept apart and not used for the training in order to validate the model in the end with "new" points that have not been used for optimization. For PINNs and in this thesis we call this the **validation data set**. But it must be said that in ML in general, this is called the **testing data set** because the validation data set is used during the training alongside the training data set. The choice to keep the formulation **validation data set** aims to better match the standard wording in CFD where the validation is performed at the end.

Algorithm 1: Algorithm to create and train a NN

Input: Hyper-parameters of the NN (size, σ) and optimization (method, learning rate η, training limit ...)
Result: A trained NN
Construct the structure of the NN $x, \theta \rightarrow y^{MLP}(x;\theta)$;
Construct the loss function $(x_{data}, y_{data}), \theta \rightarrow \mathcal{L}\left[(x_{data}, y_{data});\theta\right]$;
Prepare training data (x_{data}, y_{data});
Initialize the parameters of the model θ;
while *training limit is not reached* **do**
    Prepare the batch ;
    Compute the loss $\mathcal{L}\left[(x_{data}, y_{data});\theta\right]$;
    Compute loss derivatives $\frac{\partial \mathcal{L}}{\partial \theta}\left[(x_{data}, y_{data});\theta\right]$;
    Update parameters θ;
end
Compute loss on validation data;
Other outputs and predictions;

3.2 Physics-Informed Neural Networks as an application of MLP

Physics-Informed Neural Networks (PINN) are direct applications of feed forward NN that were presented in the previous section. The specificity of PINNs compared to a general MLP is to be found in two points:

- PINNs approximate a physical quantity as a symbolic and continuous function of physical coordinates (i.e. space and time). If a fluid or solid domain of space $\Omega \subset \mathbb{R}^3$ is considered, then the input of the MLP is the list of space time coordinates composed at most of the 3 coordinates in space $(x, y, z) \in \Omega$ and one in time $t \in [T_{min}, T_{max}] \subset \mathbb{R}$. The output is a continuous field like a component of velocity, a displacement, a pressure field, etc.
- An additional term that considers previous knowledge of physical principles is added to the loss. These physical principles can be transcribed as a set of partial differential equations (PDE). The added term in the loss is a discrete approximation of the integral of the residuals of this PDE over the domain Ω.

The first point reduces drastically the size of the NN to train since the number of inputs and outputs are limited to a very low number of scalar (between 1 to 10). This means that for getting the value of the approximated field at some point, one need to provide the coordinates in space and time of the point. Consequently, the PINN encodes the physical phenomena without having an explicit discretisation of the domain. In other terms, PINNs are mesh-free techniques. However it is limited to one solution for a given configuration.

The second points is interesting when there is little data to fit because it gives a rule on how the NN should interpolate between two data points. An illustrative example is displayed in Figure 3.3. It consists of a 1D function $y(x)$ of a 1D coordinate x that is to be retrieved with only the knowledge of two measurements (depicted by red squares on the graph). If we use any kind of fitting, we can easily find a function that goes by these two points. But this choice of fitting is quite arbitrary and there is a low confidence in the capacity of this fitting to generalize to other values (either interpolation or extrapolation). However if we know that this function describes a phenomena that has an exponential behaviour (with the ordinary differential equation (ODE) $dy/dx = \alpha y$), we can try to find a shape of function that fits the measurements and this equation. This ODE is evaluated at several points, that are called penalization points and we sample them all along the domain. They are depicted with blue cross in Figure 3.3. In that example, the parameter α was unknown but it can still be

optimized as an auxiliary variable during the training. This example is trivial since we could have also directly fitted an exponential. However, this formalism can deal with redundancy of the information and ill-posed problem. For instance, if we are provided 4 data points for only 2 degrees of freedom in an exponential model, this leads to an over-constrained problem. The optimization framework of the PINN deals directly with it. Besides it might be more robust regarding noises in data than a direct technique that would use only 2 points over the 4 to obtain a well-posed problem. Secondly, in fluid application, there is usually no analytical solution that is previously known so a PINN is one of the most general type of fitting which is compatible with an estimation of the error on the PDE.

Mathematically, the penalization of the equations uses the same technique of automatic-differentiation than for the optimization of the loss function. It is simply applied to the relation between the output and the input, which, in a PINN, have physical interpretations. Given a (potentially non linear) differential operator $\mathcal{N}$ for our PINN $x \rightarrow y(x;\theta)$, the different terms in the expression of $\mathcal{N}$ can be summed using any order derivatives of y with respect to x. In a PDE where $\mathcal{N}(y) = 0$ on a domain Ω, the residuals of the integral quantity are expected to converge toward zero:

$$\int_{\Omega} \left|\mathcal{N}\left[y\right](x)\right|^2 dx \rightarrow 0. \tag{3.13}$$

This integral is numerically approximated by a Riemann sum using Monte-Carlo method

$$\int_{\Omega} \left|\mathcal{N}\left[y\right](x)\right|^2 dx \approx \frac{1}{N_{in}} \sum_{x_{in} \in V_{in}} \left|\mathcal{N}\left[y\right](x_{in})\right|^2 \rho(x_{in}), \tag{3.14}$$

where $x_{in} \in V_{in}$ are a set of points that discretizes the domain Ω. This sum is weighted by $\rho(x_{in})$ which quantifies the importance of a point in the estimation of the integral. For a uniform sampling, the weighting of points is constant and equal to the measure of the domain (length, surface or volume in 1D, 2D and 3D respectively) : $\rho(x_{in}) = |\Omega|$. But when strong variations are expected, other strategies of sampling can be used. For example one can sample points using a Probability Density Function (PDF) that is directly linked with the norm of the gradients dy/dx or the Hessian d^2y/dx^2 or the local values of the residuals. Then using the PDF of the sampling of V_{in}, the weight scales as the inverse of the PDF $\rho(x_{in}) = \frac{1}{PDF(x_{in})}$. However for practical reasons and because we are not interested in the exact value of the integral of the residuals, we do not always take into account the effects of this non uniform weighting. The loss term associated with equation penalizations is finally

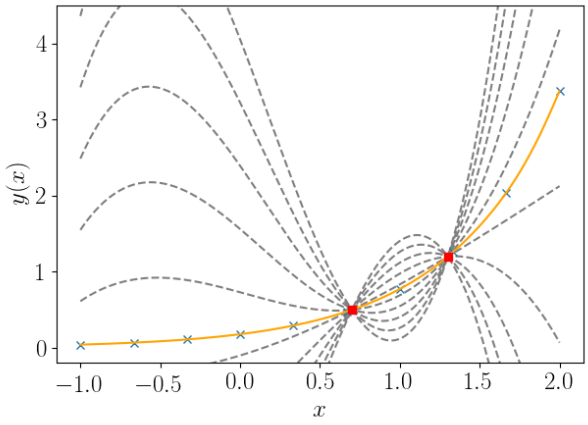


Figure 3.3 Illustration of the interest of physical regularization in PINNs. Here two fitting points are provided (red squares) and a basic PINN is trained to fit it as well as an exponential ODE (orange solid line). Penalization points of the ODE are distributed regularly (blue cross) and polynomial fittings are displayed for illustrative purpose.

written as

$$\mathcal{L}_{eqs} = \frac{1}{N_{in}} \sum_{x_{in} \in V_{in}} \left| \mathcal{N}\left[y\right](x_{in}) \right|^2, \tag{3.15}$$

and in a PINN, the complete loss is

$$\mathcal{L} = \mathcal{L}_{eqs} + \mathcal{L}_m, \tag{3.16}$$

where $\mathcal{L}_m$ is the exact same mean square error than previously described for general MLP. Here, the subscript m stands for measurements. But this part of the loss also includes boundary conditions and initial conditions if any of these are provided. In the case of a Dirichlet boundary condition, for instance $y(x) = y_0(x)$ for $x \in \partial\Omega$ where y_0 is a known function, there will be

$$\mathcal{L}_m = \mathcal{L}_{bc} = \frac{1}{N_{bc}} \sum_{x_{bc} \in V_{bc}} \left[y^{MLP}(x_{bc}; \theta) - y_0(x_{bc}) \right]^2, \tag{3.17}$$

where V_{bc} is a discrete sampling of $\partial\Omega$ with N_{bc} points in it. This loss term can also take into account the derivatives of the PINN when using Neumann boundary conditions using the same tools as in the penalization of the equations. Also it can be noticed that for a PINN which solves an unsteady problem (with inputs containing space and time dimensions), the

time dimension is strictly equivalent to another space dimension. In that sense, an initial condition is directly equivalent to a boundary condition on the border of $[T_{min}, T_{max}]$. Consequently, it is direct to use a final condition (or a condition at any time instant in $t_{bc} \in [T_{min}, T_{max}]$ with the formalism of $\mathcal{L}_m$. This is why PINNs are in design well suited for non direct problem that would otherwise be solved using the shooting method.

Finally, the summation $\mathcal{L} = \mathcal{L}_{eqs} + \mathcal{L}_m$ is a special case of $\mathcal{L} = \lambda_{eqs}\mathcal{L}_{eqs} + \lambda_m\mathcal{L}_m$ where λ_{eqs} and λ_m are weights associated with each type of loss. In some cases, different weighting can be used if these two quantities are of different order of magnitudes or have different levels of importance. It is also possible to use an adaptive weighting as discussed by Wang et al. [82] to solve convergence failures in PINNs. Nonetheless in this thesis, all the fields are made dimensionless using relevant quantities so that all the fields and their first derivatives are of order of magnitude 1. Then every sums in the loss are averaged by the number of terms. This should lead to approximately equivalent terms for each part. This is why $\lambda_{eqs} = \lambda_m = 1$ have been kept in the following sections of this thesis.

3.2.1 Types of measurements data

PINNs are very flexible with the quantity and the type of data that is provided in the loss term $\mathcal{L}_m$. Here we list different classical approaches for reconstructing the solution of a PDE with PINNs:

- The direct problem which consists in providing only the boundary (in space and/or time) conditions, without any measurement point inside the domain Ω. The physical regularization (penalization of PDE) is therefore used to interpolate inside the domain using a solution that verifies physical laws. An example is provided in Figure 3.4a where penalization points are also plotted with a random sampling of V_{in}.

- A dense data set can be used when some measurements are available everywhere in the domain as illustrated in Figure 3.4c with red points randomly sampled. This definition is qualitative and requires a certain level of interpretation. However it means that there are no large area without any other information than the physical regularization. This type of problem is usually easier to solve with PINNs than a direct problem. Indeed optimizing with a lot of data a shape that is largely known is easier than finding an unknown shape with indirect information provided by a PDE. On the other hand, this type of problem may have a lesser interest for practical applications since it means

that there is already a lot of information on the field to be reconstructed. One goal of this can be to correct the information: for example there can be noise or errors in PIV measurements and these imperfect data can be input into a PINN that can correct them and smooth them using the physical regularization. Another topic that will be discussed afterwards is the possibility to retrieve a field using dense measurements of another variable that is indirectly linked to this first quantity through a coupled PDE.

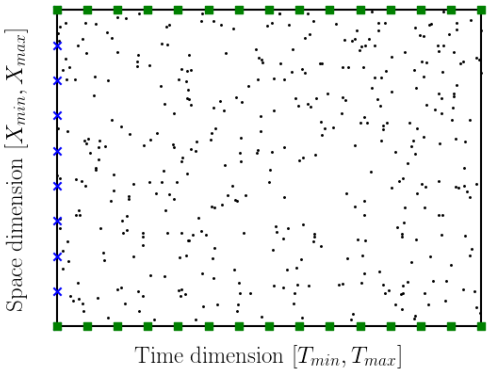
- When only a very specific area of the domain is accessible for measurements, the obtained data set can be sparse. This is the case if a recording is done with a local probe, even on a moving path as illustrated in Figure 3.4d and looked at in chapter 4. Also large area without accessible measurements can fit in the sparse data type and is illustrated with the examples of an array of cylinders in appendix A.2.2. This is a very interesting application for PINNs compared to classical methods since classical methods often need to perform a shooting technique. That consist of carrying a simulation from an initial unknown boundary and initial condition, then integrating in time, computing the difference with the provided data and then optimize the boundary/initial conditions in order to minimize this error. Therefore, this requires several time-integration of the PDE over the domain (or "shots") which is expensive while running an optimization over that. In a PINN, the shot is itself an optimization and both are done concurrently.

Reconstruction of an indirect variable is a practical application that has an engineering interest and which is highlighted in some studies. Here are some examples for incompressible flows:

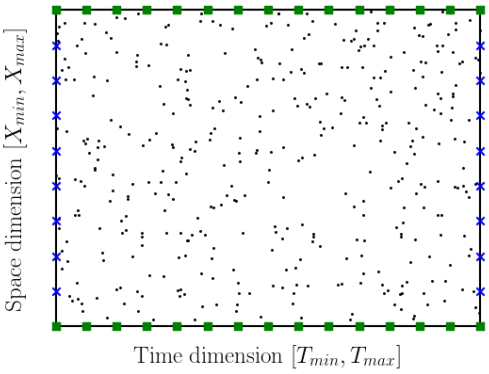
- In this thesis (and in other articles [66]), the pressure field have been reconstructed without using pressure data but only using the coupling between the pressure-gradient and other terms of velocity in the momentum equation of Navier-Stokes. This reconstruction can only be achieved up to an unknown constant for the pressure field. This is why in this thesis we have used a condition at some point to fix the constant value of the pressure. Reconstructing p leads to the estimation of the local forces on the frontiers with a solid which, then, allows the prediction of unsteady drag and lift.

- In *Deep Learning of Vortex Induced Vibrations* [72], Raissi et al. used some snapshots of the concentration of a passive scalar $c(x, y, t)$ alongside movements of a border to retrieve velocity and pressure fields. In addition of the NS equations, an advection-diffusion equation is added and the link between c and velocities is to be found in the advection term.

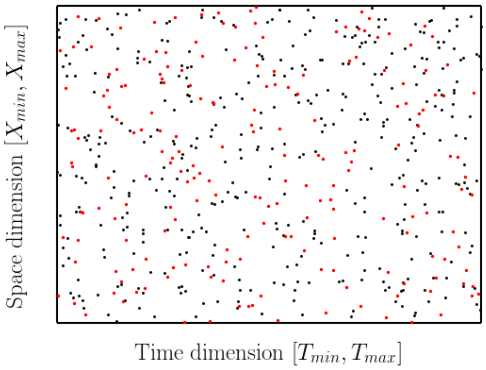
(a) Direct problem



(b) Over-constrained direct problem



(c) Dense data



(d) Sparse data

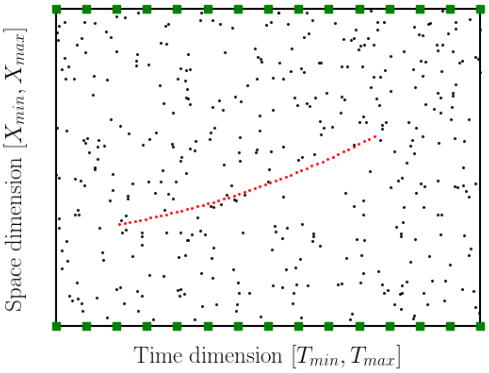


Figure 3.4 Illustration of the types of data that can be fed inside the measurement loss term $\mathcal{L}_m$. Boundary conditions are depicted with green squares (for space) and blue cross (for time), penalization points for the equations are plotted with small black points and direct measurements data inside the domain are in middle-size red dots.

- A set of 2D snapshots of temperature over an espresso cup using Schlieren videos leads to the reconstruction of the 3D, unsteady and continuous fields of velocity u, v, w, pressure p and temperature T [92]. The only direct measurement is for the temperature field. Coupling between temperature and velocity appears in the advection-diffusion equation as well as the gravity term in the momentum equation. Pressure field is retrieved with the link between the pressure gradient and other terms of velocity in the momentum equation.
- This last example also illustrated the interest of being able to perform 3D and unsteady continuous reconstruction using only 2D information at discrete time steps. Some recent papers have discussed this but outside of the framework of PINNs (Pérez et al. [93] for instance) but PINNs could be promising candidates to succeed in that kind of challenge.

The objective of retrieving a field using indirect measurements can also be reached using other types of data, even sparse measurements. Nonetheless, the difficulty of the reconstruction will increase as soon as the quantity and the quality of the measurements decrease, the sparsity increases and the level of proximity of the value that is measured and the field that is reconstructed.

3.2.2 Optimization of auxiliary variables

In many situations illustrated in the previous subsection, the problem to solve is ill-posed. There are often some redundancy in the data that over-constrain the solution. Especially, there is no certainty that there is a solution that verifies exactly the PDE and the fitting. In the example depicted at Figure 3.4, only the first plot (a) can satisfy the condition for the problem to be well-posed (in the sense of Hadamard with existence, unicity and continuity of the solution with the initial conditions). For a classical numerical simulations formulated as a direct problem, this property is required for the solution to be computed. But it would not be possible for the conditions depicted at 3.4b where both initial and final conditions are provided for the time dimension. In a PINN, this is less of a problem since the solution of PDE and the distance to boundary conditions are optimized concurrently and incorporated in the loss in a similar way.

Moreover, this redundancy of information can be used to find some auxiliary parameter while the field is reconstructed. This is called parameter identification and it has several application. Here we give several examples of these:

- In the example illustrated in Figure 3.3, the objective is to fit two data points (x_{data}, y_{data}) using a PINN $y(x;\theta)$ which is informed that y verifies an exponential-type behaviour $y' = \alpha y$. At that point α is unknown. However, as the expected solution is of the form $y^{theo.} = \beta e^{\alpha x}$, there are only two degrees of freedom. With these two data points, finding β and α is a well-constrained problem under the conditions that the data points are compatible with an exponential shape (and especially that they have the same sign). This can be extended to the PINN where the loss function is :

$$\mathcal{L} = \frac{1}{2}\sum_{k=1}^{2}\left[y(x_{data};\theta) - y_{data}\right]^2 + \frac{1}{N_{in}}\sum_{x_{in}\in V_{in}}\left[y'(x_{in};\theta) - \alpha y(x_{in};\theta)\right]^2. \tag{3.18}$$

  In Tensorflow, α can also be set as a variable to be optimized as all the other variables of the models θ which leads to the computation of $\frac{\partial \mathcal{L}}{\partial \alpha}$ alongside $\frac{\partial \mathcal{L}}{\partial \theta}$.

- In a simple Poiseuille problem consisting of a 2D steady incompressible flow in a rectangular channel of width H and length L as pictured in Figure 3.5a, the expected profile is parabolic :

$$\begin{aligned} u(x,y) =& Re\frac{P_{in} - P_{out}}{2L}\left(\frac{H^2}{4} - y^2\right), \\ v(x,y) =& 0, \\ p(x,y) =& P_{in} - \frac{P_{in} - P_{out}}{L}x. \end{aligned} \tag{3.19}$$

  So we can train a PINN to solve this problem while providing it with some simulated measurement data using the analytical solution. In this example, we used a sampling for penalization of the unsteady equation V_{in} of size $N_{in} = 4 \times 10^4$, and a dense sampling of points for measurements (u_m, v_m, p_m) of size $N_m = 1 \times 10^3$ as well as boundary conditions sampled on V_{bc} (of size N_{bc} for each boundary). The complete loss function writes as

$$
\begin{aligned}
\mathcal{L} =& \frac{1}{N_{in}} \sum_{x_{in},y_{in} \in V_{in}} \left\{ \left[u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - Re^{-1}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\right]^2 \right. \\
&+ \left[u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - Re^{-1}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\right]^2 \\
&\left. + \left[\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right]^2 \right\}_{x=x_{in},y=y_{in}} \\
&+ \frac{1}{N_m} \sum_{x_m,y_m \in V_m} \left\{ [u-u_m]^2 + [v-v_m]^2 + [p-p_m]^2 \right\}_{x=x_m,y=y_m} \\
&+ \frac{1}{N_{bc}} \sum_{x_{bc},y_{bc} \in V_{bc}^{inlet}} [p-P_{in}]^2_{x=x_{bc},y=y_{bc}} + \frac{1}{N_{bc}} \sum_{x_{bc},y_{bc} \in V_{bc}^{outlet}} [p-P_{out}]^2_{x=x_{bc},y=y_{bc}} \\
&+ \frac{1}{N_{bc}} \sum_{x_{bc},y_{bc} \in V_{bc}^{wall}} \left[u^2+v^2\right]_{x=x_{bc},y=y_{bc}} .
\end{aligned}
\tag{3.20}
$$

where the PINNs approximates $x, y \rightarrow u, v, p$ and with Re and P_{in} that are unknown variables optimized concurrently. Here, using H, P_{in} and ρ as typical scales, the fields u, v, p and therefore the cost function are dimensionless. With as few measurement points as $N_m = 10$, the relative error ϵ on guessed Re and P_{in} (denoted with the subscript ^{guess} compared with the exact value with the subscript ^{exact}) were of the order of magnitude of 1×10^{-3}. But what is even more interesting to observe is the effect of noise on the identification. Therefore, we added a Gaussian noise $\mathcal{N}(\mu = 0, \sigma)$ with a zero-average and a given standard deviation σ. The noise is independently added to every measurement point for u_m, v_m and p_m and several PINNs are trained using different levels of σ. The results depicted in Figure 3.5b show that the relative error $\epsilon = \left|\frac{Re^{guess}-Re^{exact}}{Re^{exact}}\right| + \left|\frac{P_{in}^{guess}-P_{in}^{exact}}{P_{in}^{exact}}\right|$ is stable at low values for a noise level σ lower or similar to 1×10^{-2}. Even for $1 \times 10^{-2} < \sigma < 1 \times 10^{-1}$, the relative error for both predictions stay under 1% which tends to indicate that the identification is noise-robust on a large range of noise level.

- In chapter 4, we performed a study that shows that delays for out-of-synchronization signals can be retrieved in a periodic flow using the same formalism for parameters identification.

- A slightly different example of application is provided by Raissi et al. in *Deep Learning of Vortex Induced Vibrations* [72] where stiffness and damping coefficients of a spring-mounted cylinder are identified with a relative error below 1%.

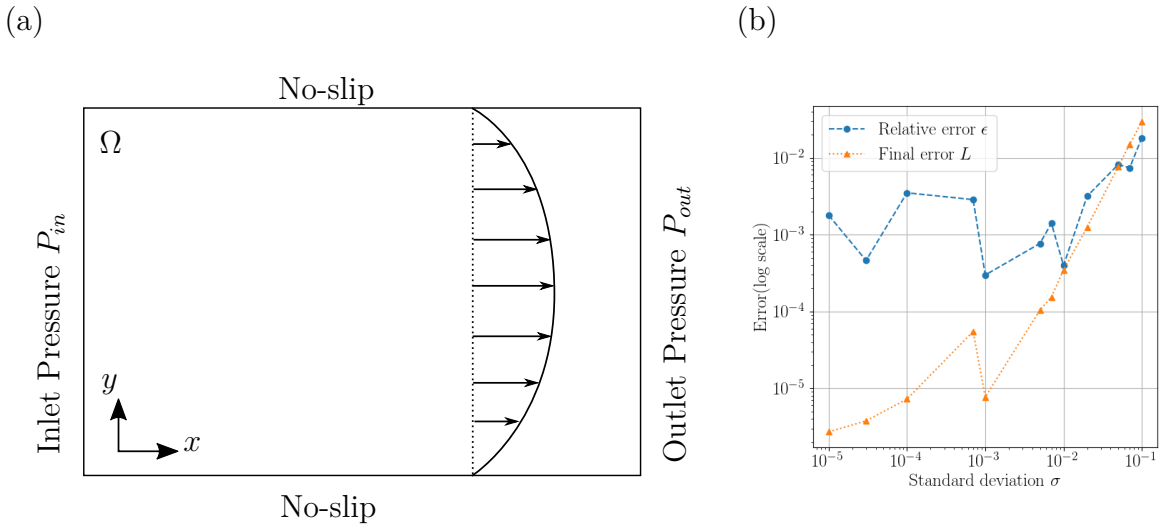


Figure 3.5 Identification of flow parameters in a Poiseuille flow: geometrical configuration used for the training of PINNs (a); and the sum of relative errors obtained for the identification of Reynolds number and pressure gradient as a function of noise in the measurements (b)

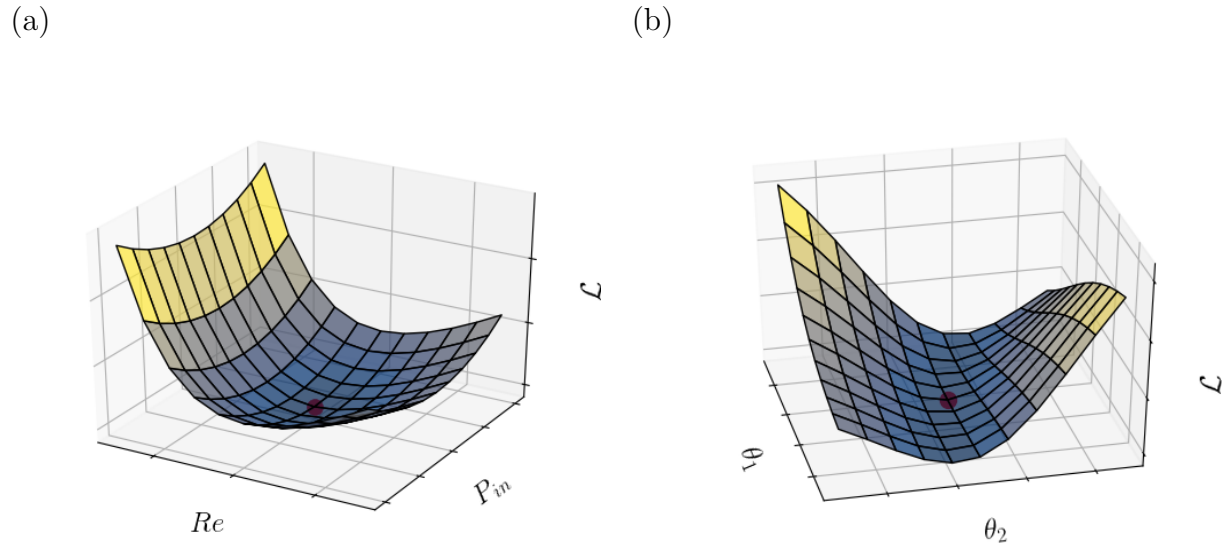


Figure 3.6 Shape of the loss function $\mathcal{L}$ of a PINN that solves a Poiseuille flow as described at page 34. Auxiliary variables Re and P_{in} in (a), and two parameters of the model θ_1 and θ_2 randomly selected in a weight matrix and a bias vector (b) have been changed around their optimum position (denoted with a red point). Other parameters values have been set to their optimized state.

The effect of auxiliary variables (the parameters that are identified) on the shape of the loss function is difficult to quantify in general. However, as depicted in Figure 3.6 for the Poiseuille flow where the total loss $\mathcal{L}$ containing penalization of equations, BC and a few measurements have been plotted for values of parameters around their optimized state, auxiliary variables might create a parameter space where $\mathcal{L}$ can be nearly convex near the optimum value. Moreover, as intuited[1] by Goodfellow et al. [2], even if neural networks are non-convex in general, it seems that in many applications the loss function often displays simple mathematical shape with nearly-convex properties. Besides, the effect of auxiliary variables (like the Reynolds number and the pressure gradient to retrieve in the Poiseuille flow) seems to be similar to those of other parameters that define the neural network.

In terms of redundancy, it may also be noted that optimizing a PINN is equivalent to finding optimal values for all the parameters in θ. In other words, it can be better to over-constrain the optimization by using a number of points sampled in the domain (for penalization of the equations and for measurements) that is greater than the number of degrees of freedom θ.

3.3 Specificity of PINNs for incompressible fluid flows

This thesis focuses on the use of PINNs for fluid dynamics (with pieces of fluid-structure interactions in the last chapter). In this section, we recall the equations and typical boundary conditions that will be used through the examples.

3.3.1 Equation penalization

In a two dimensional domain of space $x, y \in \Omega$ considered over a time period $t \in [T_{min}, T_{max}]$, an unsteady incompressible flow is fully described by the two components of the velocity u, v (respectively horizontal and vertical) and the pressure p. In their dimensionless form, the equations of motion for a mesoscopic particle of fluid can be written in their strong form using the three Navier-Stokes equations:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{3.21}$$

[1] See the discussion in the caption of Figure 8.2 in **Chapter 8 - Optimisation for training deep models** of *Deep Learning Book*, from Goodfellow et al. which is in open access at deeplearningbook.org/contents/optimization.html

which enforces the conservation of mass (or equivalently volume for an incompressible flow). Conservation of momentum in the x and y directions are written as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - Re^{-1}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f_x, \tag{3.22}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - Re^{-1}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = f_y, \tag{3.23}$$

where f_x and f_y are external forcing terms. Here the fluid is considered Newtonian with a fixed viscosity taken into account in the Reynolds number Re. Besides for incompressible flows, it is assumed that there are no coupling with the thermodynamic properties like energy, temperature and pressure that explains the low number of equations compared to compressible Navier-Stokes equations. This assumption is valid as far as the Mach number M which is the ratio of the velocity of the flow and the speed of sound, stays under values around $M < 0.3$. For the example that we are looking at in this thesis, we want to mimic the behaviour of water at very low values of velocity compared to its speed of sound (about 1480 m/s at 20 °C), which justifies the incompressible hypothesis.

3.3.2 Boundary conditions and computation of forces

In flow problems, here are the classical boundary conditions (BC) that we may apply on a border with normal and tangent vectors $\mathbf{n} = (n_x, n_y)$ and $\mathbf{t} = (t_x, t_y)$:

- The **no-slip** boundary where the velocity of the flow equals the velocity of the border. For a fixed solid, it simply consists in $\mathbf{u} = (u, v) = \mathbf{0}$. This is the standard BC for a wall-type boundary.
- **Slip** boundary conditions are less restrictive than the no-slip condition. Especially, free-slip allows a non-zero tangential velocity. However, no normal velocity is permitted ($\mathbf{n}.\mathbf{u} = 0$) since it would violates the principles of non penetration of the flow in the solid. In that case it can also be able to specify a shear rate or even more complex finite slip.
- A **symmetric** boundary condition can be considered for dividing the domain in half. In that case we have a zero gradient of the tangential component of velocity and pressure $\frac{\partial}{\partial \mathbf{n}} = \left(n_x\frac{\partial}{\partial x} + n_y\frac{\partial}{\partial y}\right) = 0$ for u, v and p as well as a non-penetration for the normal velocity $\mathbf{u}.\mathbf{n} = n_x u + n_y v = 0$.

- A uniform flow, especially for inlet condition $(u, v) = (u_{bc}, v_{bc})$.
- A given value of pressure, or a value for the average pressure as proposed by `ANSYS CFX` for outlets conditions.

For PINN reconstruction, it may also be possible not to penalize boundary condition, especially for the exterior border of the domain. Indeed the information of the effect of exterior BC on the interior flow can be implicitly given by dense measurements data distributed inside the domain. However this might generate some errors for the extrapolation of the fields outside of the spatial range of data. Nonetheless this is quite practical if we reconstruct only a part of the flow far from the actual boundaries (wall in a wind tunnel, or BC in CFD data). Consequently, there are no simple boundary conditions to apply in that case.

Other dynamical conditions are discussed especially in the fluid-structure chapter since the kinematic quantities (u and v) are related to the forces (shear stress, pressure). Besides, prediction of the forces from sparse measurements in the flow is also useful for the design of an engineering solution. In order to compute the forces, any derivative of the flow quantities along the boundary direction should be accessible for computation. To do so, a curvilinear abscissa is used to define a 1D border. This curvilinear abscissa s is made dimensionless and normalized so that the function $s \in [0, 1] \rightarrow x_{bc}(s), y_{bc}(s)$ defines the coordinates of the border. The two values $s = 0$ and $s = 1$ define the two extremities of this border and also define an orientation that will orient the normal and tangent vectors. Finally, tangent vector is obtained with the derivatives of these coordinates along the path:

$$\mathbf{t} = (t_x, t_y) = \left(\frac{\partial x_{bc}}{\partial s}, \frac{\partial y_{bc}}{\partial s} \right). \tag{3.24}$$

In a 2D problem, the normal vector can be found with a rotation of 90°:

$$\mathbf{n} = (n_x, n_y) = (-t_y, t_x) = \left(-\frac{\partial y_{bc}}{\partial s}, \frac{\partial x_{bc}}{\partial s} \right), \tag{3.25}$$

so that $\mathbf{n}.\mathbf{t} = n_x t_x + n_y t_y = 0$. In a 3D problem where the border would be a plane, one would have to obtain two independent tangential vectors and then compute the normal direction with a cross product.

Then, the local fluid force on the border consists of a pressure term $-p\mathbf{n}$ and a viscous term $\tau.\mathbf{n}$. This last one has both normal and tangential components. Here the normal is considered

from the solid to the fluid following the usual conventions. This detail must be considered while choosing the orientation of the curvilinear abscissa. The expression of the local force vector $\mathbf{df} = (df_x, df_y)$ writes as:

$$df_x = -pn_x + \frac{2}{Re}\frac{\partial u}{\partial x}n_x + \frac{1}{Re}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)n_y, \tag{3.26}$$

$$df_y = -pn_y + \frac{2}{Re}\frac{\partial v}{\partial y}n_y + \frac{1}{Re}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)n_x, \tag{3.27}$$

and to retrieve the complete force, it can be integrated all over the border

$$\mathbf{F}(t) = \int_{\partial\Omega_f} \mathbf{df}(x, y, t)dl = \int_{[0,1]} \mathbf{df}\left(x_{BC}(s), y_{BC}(s), t\right)\left|\frac{dl(s)}{ds}\right| ds, \tag{3.28}$$

with the change of variable s that introduces the length per unit of s: $dl = \left|\frac{dl(s)}{ds}\right| ds$. Then we approximate this quantity using the Monte-Carlo method as described previously at equation 3.14

$$\mathbf{F}(t) \approx \frac{1}{\mathrm{card}V_s}\sum_{s\in V_s} \mathbf{df}\left(x_{BC}(s), y_{BC}(s), t\right)\left|\frac{dl(s)}{ds}\right|, \tag{3.29}$$

by sampling points only for $s \in V_s \subset [0, 1]$ where V_s is a discrete sampling of size N_s. The same type of non uniform sampling strategies may apply here, especially when the local forces varies steeply along the border. In a uniform sampling, $\left|\frac{dl(s)}{ds}\right|$ would equal the length of the border.

3.4 Discussion on training and improvements

In this section are described technical details that are required to reproduce the results presented in this thesis. It seems that a judicious choice of these settings has a major influence on the convergence of PINNs. However, there is no practical guide that can be found in the scientific literature to our knowledge. The ideas presented through this sections are therefore derived from the experience of the test cases explored during this thesis and by reverse-engineering of several PINN codes from papers [66, 72] among others. When it is possible, some simple mathematical justifications are derived. Nonetheless, a more extensive guide could be of interest for future developments of PINNs. One can also refer to Monte-Carlo

techniques [94, 95] for a more general introduction.

3.4.1 Strategies of point sampling

Although PINNs are mesh-free methods since the definition of the solution do not rest upon elements or points attached at specific spatial location, the equations that we aim at enforcing need to be evaluated at actual locations. To do that, one needs to sample points which approximate integral quantities as discussed alongside equation 3.14 for the PDE residuals and equation 3.29 for the integration of local force values along a frontier. As demonstrated in section 3.1, this sampling is completely independent of the solution, it can be provided before or after the creation of the model and even changed during the optimization process. In this subsection are described some practical strategies to get a more accurate solution regarding the physical regularization term.

For equations penalization

For a given domain, a direct method to perform the sampling of penalization points is to use a random uniform distribution, especially when there is no a priori knowledge of the solution. This is the method performed by Raissi et al. [66] for several PDE. In practice, for a domain defined by a rectangle $x, y \in [L_{x,min}, L_{x,max}] \times [L_{y,min}, L_{y,max}]$, this sampling is obtained using an affine transformation

$$\begin{aligned} x_{in} &\sim L_{x,min} + (L_{x,max} - L_{x,min}) \times U(0,1), \\ y_{in} &\sim L_{y,min} + (L_{y,max} - L_{y,min}) \times U(0,1), \end{aligned} \tag{3.30}$$

where $U(0,1)$ is the uniform law in the interval $[0,1]$. If a body is present in the domain (like a cylinder), the points inside the body need to be removed from the set of generated points. A function that performs this discrimination is required. In the case of a cylinder which centre is x_c, y_c and which is of radius r_c, the threshold is :

$$\text{if } r = \sqrt{(x - x_c)^2 + (y - y_c)^2} < r_c, \text{ then discard the point.} \tag{3.31}$$

This technique can be extended if several bodies are present with the requirement of being outside every cylinder. But for non trivial shapes, this calculus may be less direct. In the

case of a turbine blade profile presented in appendix A.2.1, a function has already been constructed that gives the radius of the border with respect to a chosen centre and the angle θ. In that case, we can replace r_c by $r_c\left(\theta(x,y)\right)$ in equation 3.31. But for a more general context, one will need a function $h(x,y)$ with values that can be compared to a threshold to determine if the newly sampled points are inside or outside the bodies (and thus the fluid domain). This function can be a pre-trained NN on data from CAD. One example presented in appendix A.2.1 is $h(x,y) = \text{distance}[(x,y), \partial\Omega]$ the distance between a given point (x,y) and the nearest point on the border $\partial\Omega$ (if outside the body, 0 for points inside the body). In that case, only points that have $h(x_{in}, y_{in}) > 0$ would be conserved for training of the equations in the fluid domain.

In the case a mesh already exists, it is possible to use the points at the nodes for the penalization of the equations. However, it can be problematic not to be able to generate other points for validation purpose for instance. Also, depending on the activation function, it can be expected that a regular distance between the points (from a Cartesian grid for instance) might introduce a length scale that has no physical sense and oscillations at a given wavenumber could appear.

It is also possible to increase the number of penalization points in some area where we expect the flow to have more complex patterns, like the boundary layer with larger shear rate and possible detachment. This idea have been employed by Raissi et al. [72] for the unsteady flow around a cylinder. For high speed flows with shock-waves, the influence of the sampling is studied by Mao et al. [71]. An accurate reconstruction of these phenomena can lead to improvements in the estimation of the forces and for other flow features further down the wake (like some shedding of vortices that is influenced by the detachment of the boundary layer). A way of doing this is to reserve a given proportion of penalization points for a reduced area in the desired region. For a cylinder, the area defined by the points that are less than a radius away from the border can be used with that purpose in mind. The concentration of penalization points near the boundary is implemented in chapter 4. In the end, there is no absolute method to balance how many points in which area one needs to sample with no a-priori estimation of the flow features. Only experience might tell the good practices which result in a compromise between computational costs with the available resources, training time, desired accuracy and priority features.

For estimation of integral quantities

An accurate estimation of forces is of prime interest for engineering application. However, an exact computation for the integral is not available. An approximate solution is constructed as a generalization of the rectangle method. For a quantity $f(x)$ alongside a direction x in an interval $[a, b]$, the regular discretisation which is a Riemann sum consists in:

$$F = \int_a^b f(x)dx \approx \frac{b-a}{N} \sum_{k=0}^{N-1} f\left(a + \frac{k}{N-1}(b-a)\right). \tag{3.32}$$

In the case of a non-regular discretisation of $[a, b]$, the Riemann sum writes as

$$F = \int_a^b f(x)dx \approx \sum_{k=0}^{N-1} \Delta x_k f\left(x_k\right), \tag{3.33}$$

where $\Delta x_k = x_{k+1} - x_k$ is the step size. In a Monte-Carlo method, a very similar approach is used excepted that the $(x_k)_k$ is not an ordered discretisation of the domain like a grid in a mesh. Instead, $(x_k)_k$ is a random sampling of points in the interval $[a, b]$. Therefore $x_{k+1} - x_k$ does not make sense anymore since the values of x_k are not ordered. However, this term still quantifies the concentration of points at x_k. This is replaced by $\Delta x_k = 1/(N \times p(x_k))$ where p is the probability distribution function (PDF) of the random variable x. It can be proved that the estimator $I = \frac{1}{N} \sum_k f(x_k)/p(x_k)$ gives the right result on average by looking at the expectation of the random variable f/p:

$$E[f/p] = \int_a^b \frac{f(x)}{p(x)} p(x)dx = \int_a^b f(x)dx = F. \tag{3.34}$$

The choice of sampling probability function is however not easy. A uniform sampling provides a direct way, straightforward to implement in the code since $p(x) = \frac{1}{b-a}$. However, it may not be adapted to capture steep variation of f at some locations. In other words, only few points are enough to approximate the integral quantity using a rectangle method where f is nearly constant. However when $|df/dx|$ is large, the error is more important and there would be improvements if more points were sampled in that area. To understand it more mathematically, we take the example of a Riemann sum. If we assume that f is at least $\mathcal{C}^2$, which is the case since we use combinations of functions that are $\mathcal{C}^\infty$ in the NN, then we can write the second order Taylor expansion near a point x_k:

$$\begin{aligned}\delta F(x_k) &= \int_{x_k}^{x_k+\Delta x_k} f(x)dx, \\ &= \int_{x_k}^{x_k+\Delta x_k} \left[f(x_k) + x\frac{df}{dx}(x_k) + \frac{x^2}{2}\frac{d^2f}{dx^2}(x_k) + \mathcal{O}(x^3) \right] dx, \\ &= \Delta x_k f(x_k) + \frac{\Delta x_k^2}{2}\frac{df}{dx}(x_k) + \frac{\Delta x_k^3}{6}\frac{d^2f}{dx^2}(x_k) + \mathcal{O}(\Delta x_k^4).\end{aligned} \tag{3.35}$$

Now that $F = \sum_k \delta F(x_k)$ we have the error on the estimation of F that scales like

$$F - \sum_{k=0}^{N-1} \Delta x_k f\left(x_k\right) = \sum_{k=0}^{N-1} \left[\frac{\Delta x_k^2}{2}\frac{df}{dx}(x_k) + \frac{\Delta x_k^3}{6}\frac{d^2f}{dx^2}(x_k) + \mathcal{O}(\Delta x_k^4) \right]. \tag{3.36}$$

So we may want to balance a large value of $\frac{df}{dx}$ by a small value of Δx_k. Recalling that $\Delta x_k \sim 1/p(x_k)$, we could have a PDF that is defined using a power of $|df/dx|$.

In the case where a higher integration scheme is adapted, for example:

$$F \approx \frac{1}{N}\sum_{k=0}^{N-1} \Delta x_k f(x_k) + \frac{\Delta x_k^2}{2} f(x_k) = \frac{1}{N}\sum_{k=0}^{N-1} \frac{f(x_k)}{p(x_k)} + \frac{1}{Np^2(x_k)}\frac{df}{dx}(x_k), \tag{3.37}$$

the error would depend mainly of the second order derivative d^2f/dx^2 at the first remaining order. Thus, the sampling distribution could follow a probability p that scales with a power of $|d^2f/dx^2|$.

The practical interest in PINNs is that those derivatives are directly available, even on complex path like borders defined by curvilinear abscissa. This makes it quite direct to define such a sampling using an existing quantity related to the output of a PINN. However to our knowledge, it has not been used explicitly in any paper from the literature.

In the case of $p(x) = c\,|df/dx|^\alpha$ where c and α are 2 positive constants, it is possible to sample points using the rejection method [96], also found under the name accept-reject method. A simple version of this method consists in:

$$\text{Choose } x \sim U(a,b) \text{ and } y \sim U(0, \max_{[a,b]} p), \text{if } y < p(x) \text{ then keep } x, \text{ otherwise discard } x. \tag{3.38}$$

With the application of the computation of forces in mind, this can be translated with $s \in [0, 1]$ instead of $x \in [a, b]$.

Sampling of penalization points in several batches

Although it is not that useful to sample measurements data points into small samples in their is only a little data, it is however of interest to use as much as possible different training data points for equation penalization. However there are technical limitations in terms of memory capacity. Moreover as ML is designed for parallelized tasks on GPU architecture, the RAM on a GPU is today more expensive and can not be upgraded easily compared to the RAM associated with a CPU. Besides, a GPU is physically located far away from the CPU. This may cause memory transfer bottlenecks if the whole training set need to be sent in several packs between CPU RAM and GPU RAM at each iteration.

One way of dealing with this issue is to split the training set of equation penalization points into several batches and perform a turn over at a given rate between the batches. However there is a compromise to find regarding the actual size of the batch. If they are too small, the information is incomplete because physical equations are not penalized everywhere enough to properly average. Consequently the optimization steps might be erratic. On the other hand, the width of a batch is limited by the available space on the GPU RAM and with the associated increase in computational time.

Distribution of batches on several GPUs

Scaling to a larger configuration is actually limited by the available memory on a GPU. However, since the calculations on a GPU are massively parallel, it is also possible to distribute the training between several GPUs. In theory, it could even be possible to split the training tasks heterogeneously with mixed GPU/CPU on different workstations and with different sizes of batches on each of it. Each individual unit would therefore compute a part of the loss (either $\mathcal{L}_{eq}(V_{in}^k)$ on a given set of equations penalization points $(V_{in}^k)_k$, or a part of $\mathcal{L}_m$ for the k^{th}-unit) and the gradients $\partial\mathcal{L}/\partial\theta$. Then all contributions would be summed from all the individual compute units in the main CPU and the coefficients θ would be updated and sent to all compute units.

This principle is depicted in theory in Figure 3.7 where θ^n and η^n are the parameters of the model and the optimizer at the n^{th} iteration and where V_1 and V_2 are two batches for the data points where to compute the loss. Also the structure of the NN is stored in each compute unit. The advantage of this method is that the memory flow is quite low (no data set is transferred between iterations and only the updated parameters are propagated between compute units), so that it could efficiently scale up to larger tasks. However we were not able to effectively implement it and it would require low-level programming skills to efficiently perform that distribution.

3.4.2 Convergence criteria and choice of NN size

It is of practical interest to have a convergence criteria to choose when to stop the training. It can be complicated to interpret directly the absolute value of $\mathcal{L}$ obtained at the end of training. Different approaches are listed below to get an estimation for the convergence criteria. It consists in compairing $\mathcal{L}$ with:

- The machine epsilon which quantifies the smallest relative difference between two floats that the computer is able to differentiate. Considering that ML applications are often performed in half-precision (sometimes mixed-precision) which has a machine epsilon $\epsilon_{FP16} \sim 5 \times 10^{-4}$, and that some CFD runs with double precision ($\epsilon_{FP64} \sim 1 \times 10^{-16}$), we ran most of our codes in simple precision as a compromise over precision and memory bandwidth limitations when working on a GPU. In simple precision $\epsilon_{FP32} \sim 1 \times 10^{-7}$. Therefore, one should not expect any values of $\mathcal{L}$ significantly under $10^{-7} - 10^{-6}$.

- The square root of the loss $\sqrt{\mathcal{L}_{eq}}$ corresponds to a weighted L^2 error on equations. It can be compared to Root Mean Square (RMS) targets in classical CFD codes. However, it is non-direct to perform these comparison since some commercial CFD codes use proprietary normalization that are not publicly available. For instance in `ANSYS CFX`, the convergence criteria is either an RMS or a maximum value for the residuals of each of the equations but there is a normalization which is space-dependant and that we were not able to fully understand. In our case, all the physical fields are dimensionless and of order of magnitude 1, but a non uniform sampling may result in a different loss value if this weight is not taken into account as discussed before.

- The evolution of residuals during optimization often displays a plateau. For instance at Figure 3.8a, the history of the PDE loss is plotted in the case of the Poiseuille flow described at Figure 3.5a. In that case, no measurements data were provided and

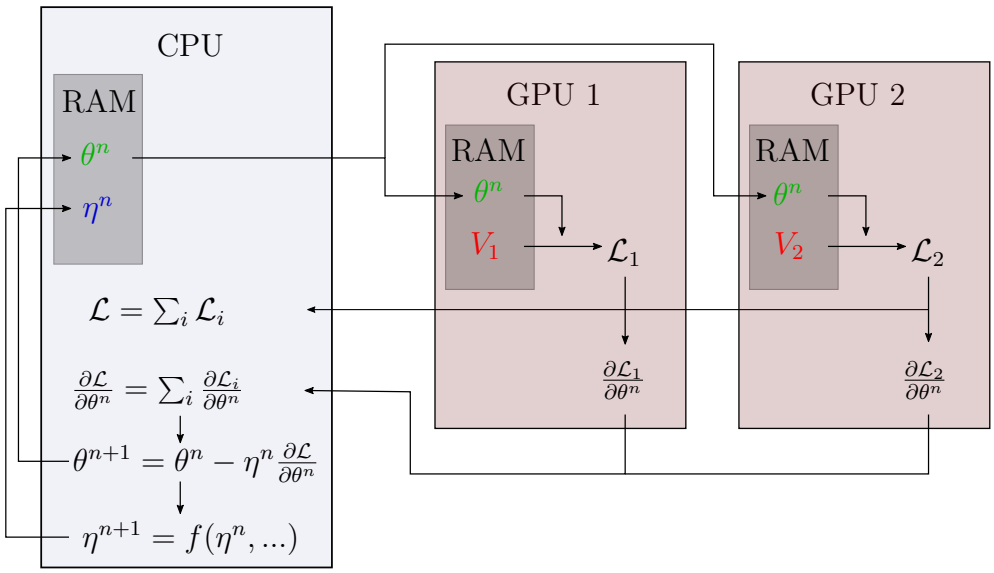


Figure 3.7 Distribution of training on several heterogeneous computational units

convergence of losses are compared for several NN sizes. One can see that for a specific test case, converged NN are those for which the final value of the training loss went under $\sim 10^{-5}$. In other cases discussed later for more complex flows and others read in recent papers, it seems that an "acceptably" converged PINN has an error on its PDE around 10^{-3} or under. But this value may depend upon the actual test-case and configuration so it is difficult to give general advices. In most papers, the final value of $\mathcal{L}_{eq}$ is not mentioned. In Mao et al. [71], it seems that $\mathcal{L}_{eq}$ converges around $10^{-2} - 10^{-3}$ whereas in Jin et al. [97], this plateau is around $10^{-3} - 10^{-5}$. But the difference of test-cases makes it complicated to draw general conclusions.

However, the fitting part of the loss $\mathcal{L}_m$, and especially the one for validation, is more direct to interpret since we can compute an L^2 relative error by dividing $\sqrt{\mathcal{L}_m}$ by the L^2 norm of our fields. Empirically we have observed that an acceptable error on the measurements can be found around a few percents of relative difference. As the fields are of order of magnitude 1, this usually means that for $\mathcal{L}_m \sim 10^{-4} - 10^{-3}$ or above, PINN predictions are in good accordance with the validation data.

In classical numerical simulation, one tool to quantify the error that comes from the discreti-

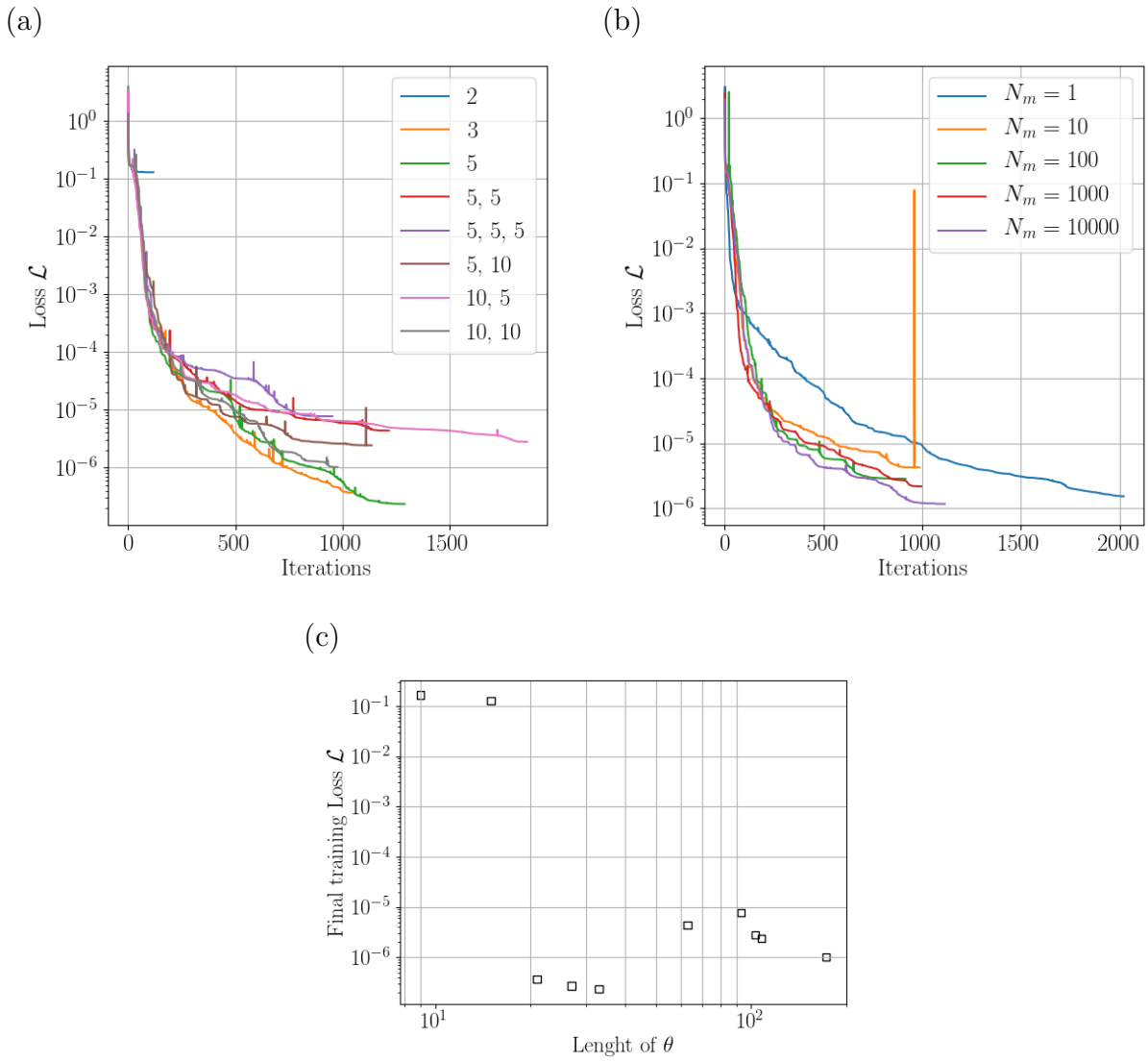


Figure 3.8 Loss history for the computation of a Poiseuille flow: (a) for several size of NN where $N_m = 0$; (b) with different numbers of measurements points N_m. In (c) a comparison of final training loss value extracted from (a) is plotted versus the number of scalar parameters in the NN.

sation of the problem is to perform convergence study. One way is to increase the number of points in the mesh, or using higher-order elements (especially for the FE method). There is no direct transposition of this to PINNs since the method does not rely on a mesh. However, NN with a larger number of degrees of freedom can be used. This is directly the number of scalar in the matrices and vectors used in the sequence of affine transformations that are tuned through optimization. It is also equal to the length of the vectors θ of all the parameters of the NN.

As pictured in Figure 3.8c, the dependency of the final value of loss function (the residuals of PDE) is not smooth in the case of solving the Poiseuille flow without measurement data. The two first points on the plot are PINNs $x, y \to u, v, p$ with a structure $[2, 1, 3]$ and $[2, 2, 3]$ neurons respectively, with hyperbolic tangent as activation functions. Residuals $\sim 10^{-1}$ are well above 10^{-3} and qualitatively flows are not converged even roughly. This can be explained because the elementary bricks are not tunable enough to approximate both u, v and p in the same graph. However, starting from a size $[2, 3, 3]$, the obtained residuals fall under 10^{-6} and are even lower than some runs with more degrees of freedom. This is a behaviour that have been encountered in other cases in this thesis and in literature (see for instance Figure 6.7 in Goodfellow et al. [2]) and that can be summarized as follow (like the ModalPINN in chapter 4): in simple applications, as soon as the number of degrees of freedom is large enough to represent accurately the shape of the function (in space or time, and this criteria can be verified qualitatively), increasing the number of degrees of freedom might not improve significantly the precision of the solution, at least when training time is fixed. This counter intuitive behaviour might be understood as a fragility in the optimization process. As the problem is non convex, it is easy to be trapped in a local minima, especially when the number of dimension increases with the size of the NN. From results presented in Figure 3.8a, we can also notice that there is no clear indication between increasing the width of the NN or its depth.

In other cases, in order to determine an adequate size of NN and the expected final loss values, we proceed by following these steps:

1. Verify that the size of the NN is large enough to qualitatively approximate the phenomena. To do so, training with dense data only (no equations penalization) was performed and we checked that the reconstructed field is nearly indistinctible from the data from numerical simulations.

2. Find the lower values for the NN size before there is the drop in loss residuals. This

drop can be steep as in Figure 3.8c or smooth as presented in chapter 4 for the vortex shedding using a ClassicPINN.

3. Perform flow reconstruction using both dense data and equation penalization. At the end, there is an order of magnitude of the expected residuals for $\mathcal{L}_{eq}$. At this point, the number of measurements points N_m may have only a limited influence as illustrated with the example in Figure 3.8b where the Poiseuille flow is solved using equation penalization as well as fitting data (of size N_m) from the analytical solution. From the evolution of the training loss across iterations it can be noted that the influence of N_m is not noticeable as soon as N_m is at least of the order of magnitude of 10. This means that only a few fitting points in the domain significantly help the reconstruction of the solution. For more complex applications, this typical value of N_m might be larger.

4. Use these criteria in more complex training configurations (sparse data, inverse problem, parameters identification).

3.4.3 Training strategy in terms of optimizers and initialization of parameters

Several types of optimizers have been introduced in section 3.1. In practice, inspired by the codes provided with some papers [65, 72, 97], we used a combination of two optimizers L-BFGS [2, 98] and Adam [91].

First, we used L-BFGS which is imported from the Scipy library interface. We do not have a direct control of each optimization step, apart from a callback function that allows outputting the loss value. The hyper-parameters that are set for training are both a limit of iterations and function calls (usually set to 5×10^4 but sometimes lower if it exceeds the available time for computation) and a tolerance parameter when the relative difference between two steps falls under it. By default this value was fixed near the machine epsilon. Details about the parameters can be found in the documentation [99].

Then, the Adam optimizer is used directly from the Tensorflow library. For this optimizer we have control on each iteration in the convergence loop so we can easily choose which batch, training limit and outputs to use. We used a main learning rate of $\eta = 1 \times 10^{-3}$ in most of the trainings and let the other values to the default which are described in the documentation [100] as well.

As discussed with the jury during the thesis presentation, there are still some questions about

the order of the optimizers. Indeed Adam offers more directly the possibility to perform training with mini-batches. This makes the loss function stochastic and therefore may prevent from getting stuck in a local minima (since the position of local minima can change between two consecutive batches). On the contrary, using L-BFGS-B with only one batch might lead to a solution that can not exit a local minima, even if followed by the Adam optimizers. One of the lead for future work would be to perform training with L-BFGS-B with several batches by fixing the maximum number of iterations for one batch to a value to be determined.

3.5 Technical details for PINNs training in this thesis

In order to reproduce the results used in our Python codes, the list of modules and libraries used is summarized at Table 3.1. Also, in order to be able to compare the performances between several hardware configurations (especially when the run is limited by a maximum duration of the training), relevant technical specifications are recalled in Table 3.2 for both training on the desktop computer and the computational cluster on Compute Canada.

Table 3.1 List of modules loaded on Compute Canada and Python packages used in the codes.

Package Name	Version	Documentation (with link)
Standard software environments	StdEnv/2020	Compute Canada wiki
Nix package manager	nixpkgs/16.09	Compute Canada wiki
Python	python/3.7.4	Python.org and Compute Canada
Numpy	1.17.4	Numpy documentation
Scipy	1.3.2	Scipy documentation
Tensorflow CPU	1.15.0 (Desktop)	Tensorflow documentation
Tensorflow GPU	1.14.1 (Graham)	Tensorflow documentation
Matplotlib	3.1.1	Matplotlib documentation

Table 3.2 Hardware configuration for runs launched on desktop (a) and on a computational cluster (b). More detailed documentation and specific availability of python packages can be found in Compute Canada wiki [1]

(a) Hardware configuration on desktop computer

Property	Value
CPU ref	Intel i9-9900k
CPU type	8 cores/16 threads at 3.6 GHz
RAM	32 Go
GPU ref	NVIDIA RTX 2060 SUPER
GPU RAM	8 Go
GPU Tensor Cores	272
GPU Cuda Cores	2176

(b) Hardware configuration on Compute Canada

Property	Value
Cluster	Graham
CPU type	Intel Xeon Gold / Xeon Silver
Number of CPU per job	2
RAM per job	50 Go
GPU ref	NVIDIA T4 Turing or V100 Volta
GPU RAM (T4)	16 Go GDDR6
GPU RAM (V100)	16 Go / 32 Go HBM2 (depends of the node)
GPU Tensor Core (T4)	320
GPU Tensor Core (V100)	640
GPU Cuda Core (T4)	2560
GPU Cuda Core (V100)	5120

CHAPTER 4 ARTICLE 1 - MODALPINN: AN EXTENSION OF PHYSICS-INFORMED NEURAL NETWORKS WITH ENFORCED TRUNCATED FOURIER DECOMPOSITION FOR PERIODIC FLOW RECONSTRUCTION USING A LIMITED NUMBER OF IMPERFECT SENSORS

This paper was submitted to Journal of Computational Physics on July, 20th 2021.

Authors : Gaétan Raynaud[a], Sébastien Houde[b], Frédérick P. Gosselin[a]
[a] Département de Génie Mécanique, Laboratory for Multiscale Mechanics (LM2), Polytechnique Montréal, Montréal, QC, Canada
[b] Hydraulic Machinery Laboratory LAMH, Faculty of Science and Engineering, Laval University, Québec, G1V 0A6, Canada

Contribution : GR carried out the work, wrote the article and did the corrections, FG and SH supervised the work and reviewed the draft.

This paper presents an integration of a Fourier modal decomposition into the architecture of a PINN. It also presents results of a quantitative study on the robustness of PINNs regarding the quality of provided data (noise, out of synchronization) while using sparse data. These studies are illustrated with a laminar flow over a cylinder. This example is limited in complexity in order to maintain brevity. Appendix A.1 discuss the choice of modal decomposition and in appendix A.2 ModalPINN is applied to more complex flow patterns. These appendices are not included in the submitted paper.

4.1 Abstract

Continuous reconstructions of periodic phenomena provide powerful tools to understand, predict and model natural situations and engineering problems. In line with the recent method called Physics-Informed Neural Networks (PINN) where a multi layer perceptron directly approximates any physical quantity as a symbolic function of time and space coordinates, we present an extension, namely ModalPINN, that encodes the approximation of a limited number of Fourier mode shapes. In addition to the added interpretability, this representa-

tion performs up to two orders of magnitude more precisely for a similar number of degrees of freedom and training time in some cases as illustrated through the test case of laminar shedding of vortices over a cylinder. This added simplicity proves to be robust in regards to flow reconstruction using only a limited number of sensors with asymmetric data that simulates an experimental configuration, even when a Gaussian noise or a random delay is added, imitating imperfect and sparse information.

4.2 Introduction

Data-assimilation techniques helped bridge the gap between experimentation, numerical simulation, and modelling in order to design better engineering solutions. However when data is expensive to gather and therefore scarce, these conventional solutions lack physical accuracy. New algorithms such as Physics-Informed Neural Networks that are presented in this paper allow using prior knowledge from Partial Differential Equations (PDE) to reconstruct continuous fields and predict quantities of interest in the small-data regime. This leads the way to improve the design of mechanical systems, monitor operations and perform predictive maintenance to reduce operating cost and increase overall efficiency.

Along with a series of advances in various fields [2] such as computer vision [39] and natural language processing [41], artificial intelligence has found numerous applications in fluid dynamics. Especially, deep learning took advantage of the massive amount of experimental data and high-fidelity simulations [101] with applications in flow estimation [58], active control [102] or complex optimisation like collective swimming [60].

From the field of dynamical systems and sparse regression [52, 103], another branch has emerged and aims at generalising the idea of test functions using multi layer perceptrons that directly approximate any physical quantity as a symbolic function of spatial and temporal coordinates. This technique, called Physics-Informed Neural Networks (PINN) finds its origins in the early work of Dissanayake et al. [63] and Lagaris et al. [64]. It was met with renewed interest since 2018. PINNs allow the reconstruction of hidden variables using different types of data without preliminary processing, identification of PDE parameters and resolution of complex direct and inverse problems governed by these types of equations [66].

The conciseness of implementation of this method and the profusion of PDE-governed phe-

nomena has lead to numerous works applying the concepts of PINNs to diverse fields such as fluid dynamics with non-newtonian fluids [67] and high-speed flows [71], but also in material sciences [74, 77], electromagnetism [78] or nano-optic and meta-materials [79].

Improvements to the mathematical basis of PINNs have been proposed to increase the efficiency and robustness of training. For instance our paper uses a technique called prior-dictionary [104] that allows enforcing prior knowledge about the solution such as boundary conditions. Other papers discuss techniques to improve the activation functions by adding degrees of freedom [84], or address deeper issues like an adaptive weighting of loss parameters [82] and gradient-related issues in the optimisation process [83].

Modal approaches have been a matter of interest in flow modelling. They allow lighter representations by extracting physically important patterns from raw data obtained by numerical simulation or experimentation [45]. This thematic has been addressed under the scope of machine learning with, for instance, Fourier content that is learned from the geometry in order to improve prediction performances during the design phase [105]. Spectral methods for high randomness have been enforced in PINNs governed by stochastics PDE [106].

Our paper is positioned in the continuity of Raissi et al. [72] where a vortex-induced vibration (VIV) phenomena is modelled using a classical PINN. Inspired by harmonic balance techniques (HBT), we aimed at directly enforcing this oscillatory phenomena in the way PINN represents information so that it gains in interpretability. For simplicity reasons, structural movement has not been considered in the presented results and only the fluid flow has been reconstructed.

From an experimental point of view, flow reconstruction might be a difficult and expensive challenge. Some techniques make it possible to obtain flow information at discrete points in a volume or on a plane with Particle Image Velocimetry (PIV) and its tomographic and holographic variants. Other techniques give only information on traverses such as Laser Doppler Velocimetry (LDV) or at single points like pitot probe or hot-wire anemometry [107]. These techniques require a substantial time for calibration and may not be available everywhere or at the same time. Moreover, recording with one pitot probe at several location results in a set of asynchronous data. In the area close to a wall, the large gradients of velocity and the heat-loss through the wall create flaws in PIV [108] and hot-wire measurements [109]. Some complex geometries like inter-blade regions in hydraulic turbines can be difficult to

access and limit the area of visualisation with optical techniques [110]. Other defects can also appear with the tracking particles such as peak-locking in PIV [111] which can corrupt the data. In this context, PINNs can help fill the gap by interpolating between sparse data like plane measurements to reconstruct 3D fields from 2D measurements [92]. There are still some questions about the ability of PINN to deal with and correct these imperfections and to extrapolate outside of the available measurement window.

The objective of this paper is to propose a simpler representation of PINNs for oscillating phenomena, namely a ModalPINN, and quantitatively show that this simplification provides robustness regarding sparsity, noise and lack of synchronisation in the provided data. The following section recalls the mathematical grounds of PINNs and present our ModalPINN. The vortex-shedding that serves as a test case is introduced in the third section with the required adaptations. Then several training configurations are run from dense and perfect data to flow reconstruction using sparse and artificially corrupted time signals.

4.3 Method

4.3.1 Theoretical background about physics-informed neural networks

ModalPINN is based on the concept of physics-informed neural network (PINN). Its formulation is presented in the next subsection alongside some precision on the use of prior-dictionary to enforce boundary conditions and unsteady force computation with PINN.

Physics-Informed Neural Networks

We consider a physical problem where an unknown variable $\mathbf{q}(\mathbf{x}, t) \in \mathbb{R}^n$ is defined as a solution of a partial differential equation. This variable $\mathbf{q}$ is a function defined on a spatial domain Ω and on a time interval $[t_0, t_f]$. The set of equations also contains a boundary term on $\partial\Omega$ and initial conditions:

$$\begin{cases} \mathcal{N}(\mathbf{q}, t) = & \mathbf{f}(\mathbf{x}, t) \quad \forall \mathbf{x}, t \in \Omega \times [t_0, t_f], \\ \mathbf{q}(\mathbf{x}, t) = & \mathbf{h}(\mathbf{x}, t) \quad \forall \mathbf{x}, t \in \partial\Omega \times [t_0, t_f], \\ \mathbf{q}(\mathbf{x}, t_0) = & \mathbf{q}_0(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega, \end{cases} \tag{4.1}$$

where $\mathcal{N}$ is a differential operator with respect to spatio-temporal coordinates, and which can be non linear.

The idea behind a PINN is to approximate the physical solution $\mathbf{q}$ with a neural network. The neural network (NN) defined by its set of parameters $\theta \in \mathbb{R}^p$ is considered as a function of physical coordinates (here space and/or time). The approximated solution $\tilde{\mathbf{q}}$ is obtained with

$$\tilde{\mathbf{q}}(\cdot) = NN(\theta; \cdot) \approx \mathbf{q}(\cdot), \tag{4.2}$$

and is completely specified once all parameters θ are set. In other words, the approximation is continuously defined without any mesh required. For the purpose of concision, the tilde is dropped and $\tilde{\mathbf{q}}$ is referred as $\mathbf{q}$ from here on. It can also be noted that having time as one of the input coordinates $(\mathbf{x}, t)$ is strictly equivalent as having an additional spatial dimension in $\mathbf{x}$.

The neural network $NN(\theta; \cdot)$ designates a symbolic graph of operations consisting of, alternatively, a matrix-vector product and a sum, and a non-linear activation function $\sigma : \mathbb{R}^j \to \mathbb{R}^j$. For a neural networks of depth k defined with the set of parameters $\theta = \{W_0, \mathbf{b_0}, ...W_k, \mathbf{b_k}\}$, where W_i are matrices and $\mathbf{b_i}$ vectors, one can obtain from an input $(\mathbf{x}, t)$ the output $\mathbf{q}$ with the following sequence of operations

$$\begin{aligned}
\mathbf{y_0} &= (\mathbf{x}, t) \in \mathbb{R}^{n_0}, \text{ usually } n_0 = 3 \text{ or } 4, \\
\mathbf{y}_1 &= \sigma\left(W_0 \mathbf{y_0} + \mathbf{b_0}\right) \in \mathbb{R}^{n_1}, \\
&\vdots \\
\mathbf{y}_{i+1} &= \sigma\left(W_i \mathbf{y_i} + \mathbf{b_i}\right) \in \mathbb{R}^{n_{i+1}}, \\
&\vdots \\
\mathbf{q} &= W_k \mathbf{y_k} + \mathbf{b_k} \in \mathbb{R}^{n_{k+1}}.
\end{aligned} \tag{4.3}$$

This sequence of operation is usually illustrated by a graph, as depicted in Figure 4.1. In this example, unknown quantities of a two dimensional incompressible flow $\mathbf{q} = (u, v, p)$ defined on a 2D cartesian domain $\mathbf{x} = (x, y)$, are solved using a PINN for each scalar variable. It is also possible to unite all the flow quantities in the same PINN. The choice of activation function and neural network size are both yet to be settled in the literature dealing with PINN. A comparison can be made with numerical methods such as finite elements where the number of degrees of freedom is linked to the number of parameters θ that quantifies the network's size. Besides, activation function σ can be viewed as a form function that will help approximate any complicated shape. For classical cases, sine and hyperbolic tangent proved to work in previous studies [65, 72]. We adopted the same choice by using $\sigma = \sin$ when there

is periodicity with one of the input coordinates, and tanh in other cases like mode shapes reconstruction.

Typical PINN algorithms optimise the set of parameters θ in order to minimise a specific loss function $\mathcal{L}$. For a PINN, the loss function is generally composed of two kinds of terms: $\mathcal{L} = \mathcal{L}_m + \mathcal{L}_{eq}$ where $\mathcal{L}_m$ and $\mathcal{L}_{eq}$ respectively represent:

1. The distance to measurements or Dirichlet boundary conditions. On a sample of coordinates V_m of size N_m, $\mathcal{L}_m$ represents the average squared distance to specific and known values $\mathbf{q}_m$:

$$\mathcal{L}_m = \frac{1}{N_m} \sum_{\mathbf{x}_m, t_m \in V_m} \left| \tilde{\mathbf{q}}(\mathbf{x_m}, t_m) - \mathbf{q}_m \right|^2, \tag{4.4}$$

   Using a quadratic norm allows smoother differentiation. Here $\mathbf{q}_m$ can be a sampling of measurements data as well as boundary conditions. In that last case, V_m would be a discrete sampling of $\partial\Omega \times I$ with $\mathbf{q}_m = h(\mathbf{x}_m, t_m)$.

2. The residuals of partial differential equations or Neumann boundary conditions:

$$\mathcal{L}_{eq} = \frac{1}{N_{in}} \sum_{\mathbf{x}_{in}, t_{in} \in V_{in}} \left| \mathcal{N}(\tilde{\mathbf{q}}(\mathbf{x}_{in}, t_{in})) - f(\mathbf{x}_{in}, t_{in}) \right|^2, \tag{4.5}$$

   where V_{in} is a sampling of the PDE domain $\Omega \times [t_0, t_f]$ where $\mathbf{q}$ is defined, and N_{in} its cardinal. For Neumann boundary conditions, V_{in} would be a sampling of $\partial\Omega \times [t_0, t_f]$ and $\mathcal{N}$ and f would be adapted consequently.

The second part benefits from automatic differentiation available with neural networks. Since every operation in the operation graph is known and differentiable, derivatives with respect to any variable in the graph can be computed exactly with most machine learning libraries such as TensorFlow [37]. Most of the time, machine learning makes use of this property to perform fast optimisation of parameters θ (with gradient descent for instance). But since the input has a physical signification in PINN, it makes sense to differentiate with respect to one of the input to compute gradients of the solution in physical space.

Once the loss function and sampling spaces V_m and V_{in} are defined, the model's parameters θ are optimised so that the approximated solution $\mathbf{q}$ fits best both equations and measurements. Several minimising algorithms are available. The quasi-Newtonian L-BFGS-B [112] followed by the Adam optimisers [91] are used and seemed effective from empirical observa-

tions. Technical details about training can be found in section 4.8.

At the end of the training, $\mathcal{L}_m$ is computed using a larger data set $V_m^{\text{valid.}}$ from simulations with new points that have not been used for optimisation. This provides a squared L^2 measure of the reconstruction error that is later referred as validation error. It should be noted that outside of PINNs literature, this quantity may be named testing loss in the field of machine learning.

Prior-dictionary

One way to take the boundary conditions into account is to penalise the error on a sampling of points. This is the method presented in the previous section and which is included in the $\mathcal{L}_m$ term. Nonetheless, it may slow down or prevent the algorithm from converging on a solution. This issue has been tackled by Peng et al. [104] who proposed a method called prior-dictionary.

The idea is to force the shape of the approximated solution to fit some criteria, especially Dirichlet conditions. If the condition to be satisfied is $\mathbf{q}(\mathbf{x}, t) = h(\mathbf{x}, t), \forall \mathbf{x} \in \partial\Omega$ which is independent of time, the approximated solution can be defined as

$$\tilde{\mathbf{q}}(\mathbf{x}, t) = NN(\theta; \mathbf{x}, t) \times f_{BC}(\mathbf{x}) + h(\mathbf{x}, t), \tag{4.6}$$

using a function f_{BC} which equals 0 at specific domain frontiers $\partial\Omega$. Following the example of a two-dimensional flow where $\mathbf{q} = (u, v)$ and $\mathbf{x} = (x, y)$, it is possible to impose a no-slip boundary condition on $y = 0$ by defining $f_{BC}(x, y) = \tanh y$. This choice is not unique. The difficulty is to select a f_{BC} that is almost flat on the entire domain except at specific boundaries in order to minimise the deformation of the neural network output $NN(\theta; \mathbf{x}, t)$.

Unsteady force computations

For the computation of forces on any border, a parametric definition is used. For instance a 1D-frontier in a two-dimensional domain can be defined by $s \in [0, 1] \rightarrow (x_{BC}(s), y_{BC}(s)) \in \mathbb{R}^2$. This is a symbolic function, either analytically defined by equations (as lines, circles, parabola...) but it can also be defined by an auxiliary neural network, pre-trained to fit any regular border. This allows a PINN to use more complex border shapes. Besides, this

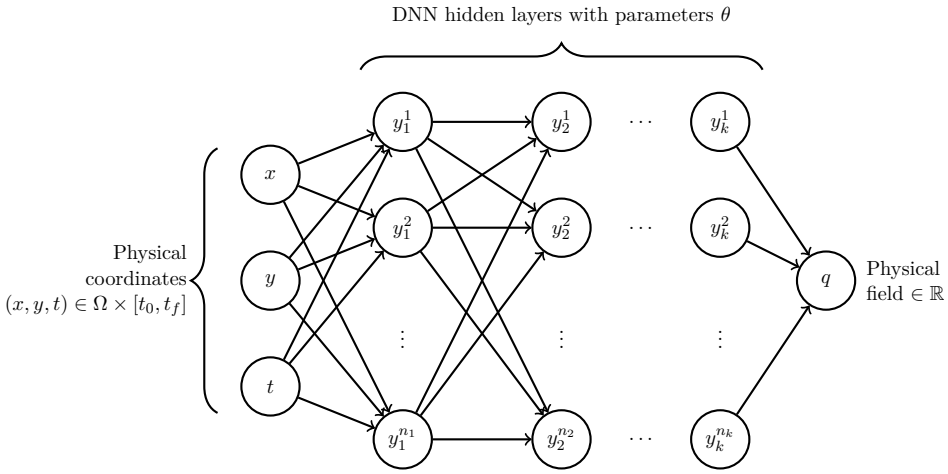


Figure 4.1 Physics Informed Neural Network classical structure for approximating a field q as a function of spatio-temporal coordinates (x, y, t). If q were to be a vector, for instance velocity components and pressure (u, v, p), all these quantities could go along in the output of one PINN or in separated neural networks.

parametric approach can be generalised to higher dimensions, such as a surface defined by $(s, \xi) \in [0, 1]^2 \to (x_{BC}, y_{BC}, z_{BC}) \in \mathbb{R}^3$. Moreover for non canonical shapes, for instance with discontinuities, it is possible to define several borders that can be separately approximated by a symbolic function.

Once a symbolic function of the border is available, computation of the normal vector is made possible using the automatic differentiation of neural networks with respect to s. In a two-dimensional problem, the normal vector is given by:

$$\vec{n}(s) = (n_x(s), n_y(s)) = \left(-\frac{\partial y_{BC}}{\partial s}(s), \frac{\partial x_{BC}}{\partial s}(s) \right). \tag{4.7}$$

Then, the total forces $\vec{F}$ on a border can be estimated using an empirical average with the Monte Carlo method in order to integrate local forces $\vec{df}(x, y)$:

$$\vec{F} = \int_{\partial\Omega_f} \vec{df} dl. \tag{4.8}$$

In the case of a two-dimensional incompressible flow, local forces $\vec{df} = (df_x, df_y)$ are given by

$$df_x = -pn_x + \frac{2}{Re}\frac{\partial u}{\partial x}n_x + \frac{1}{Re}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)n_y, \tag{4.9}$$

$$df_y = -pn_y + \frac{2}{Re}\frac{\partial v}{\partial y}n_y + \frac{1}{Re}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)n_x, \tag{4.10}$$

where Re is the Reynolds number quantifying the ratio between inertial and viscous forces, u, v and p are the dimensionless velocity and pressure fields. Then the integral in equation 4.8 is approached by the symbolic parametrization and a Monte-Carlo method

$$\begin{aligned}\vec{F}(t) = \int_{\partial\Omega_f} \vec{df}(x,y,t)dl = \int_{[0,1]} \vec{df}\left(x_{BC}(s), y_{BC}(s), t\right)\left|\frac{dl(s)}{ds}\right|ds, \\ \approx \frac{1}{\text{card}V_s}\sum_{s\in V_s} \vec{df}\left(x_{BC}(s), y_{BC}(s), t\right)\left|\frac{dl(s)}{ds}\right|,\end{aligned} \tag{4.11}$$

where V_s is a sampling of $[0,1]$ which is then mapped to the coordinates of the points on the boundary using the parametrization $s \to x_{BC}(s), y_{BC}(s)$. This sampling V_s can be uniform, in which case s is the curvilinear abscissa divided by the length of the border L and $\left|\frac{dl(s)}{ds}\right| = L$. But in case of strong variations in the integrand, an adaptive sampling can be used with Monte-Carlo method. In that case, $\left|\frac{dl(s)}{ds}\right|$ is calculated using the probability distribution function of the random variable s.

4.3.2 ModalPINN : enforcing Fourier modes in the neural architecture

Periodicity occurs for a wide range of phenomena in nature and in engineering processes. The mathematical tools and models can be adapted to use this property to significantly speed-up calculations. The following subsections present how a truncated modal representation can be directly included into the neural network architecture and how it allows another type of physical regularisation based on modal equations.

Modal decomposition encoded in ModalPINN

For a physical case where the observed phenomena is periodic for one space-time coordinate, it can be convenient to decompose the solution with a modal approach. For example, consider a real function of space and time $q(\mathbf{x}, t)$ periodic in time with a fundamental frequency

$f_0 = 2\pi\omega_0$. It can be transformed with Fourier decomposition such as:

$$q(\mathbf{x}, t) = \sum_{k=0}^{\infty} \hat{q}_k(\mathbf{x}) e^{ik\omega_0 t} + c.c., \tag{4.12}$$

where $\hat{q}_k \in \mathbb{C}$ are the modal coefficients at frequency $2\pi k\omega_0$ with $k \in \mathbb{N}$. These coefficients are functions of space only, which removes time as a variable needed to solve the problem.

In some circumstances, it is possible to obtain an acceptable approximation of $q(\mathbf{x}, t)$ with a finite number of modes. The obtained level of accuracy may depend on the presence of high frequency phenomena. Besides, high order harmonics may be required when non-linear features in the governing equations lead to interactions between modes at different frequencies. Given a number of modes N, a PINN with prior dictionaries aiming at approximating these modal shapes is constructed:

$$\mathbf{x} \in \Omega \xrightarrow{NN(\theta;\cdot)\times f_{BC}(\cdot)} (\hat{q}_0, ..., \hat{q}_N) \in \mathbb{C}^{N+1}. \tag{4.13}$$

The complete approximated solution is recovered by the sum :

$$\tilde{q}(\mathbf{x}, t) = 2Re\left(\sum_{k=0}^{N} \hat{q}_k(\mathbf{x}) e^{ik\omega_0 t}\right), \tag{4.14}$$

all this can be done in the computational graph of the neural network, as illustrated in Figure 4.2 and summarised in algorithm 2.

Loss construction with physical and modal equations

Since a modal sum can be considered as an auxiliary neural network, derivatives of q with respect to time and space are available. Consequently, one direct way of computing a loss function to penalise equations residuals is to use the same formalism as in the classical PINN approach. Thus, the modal sum is used as input to the partial differential operator. In that case, sampling space V_{in} provides a sampling in both space and time. For the penalisation to be satisfactory, the number of points required is significantly higher than for a space-only problem. Though, for a time periodic solution, the time domain can be reduced to $[0, \frac{2\pi}{\omega_0}]$. This will be referred to as physical equations.

To go beyond this space and time sampling, an advantage can be drawn from the availability

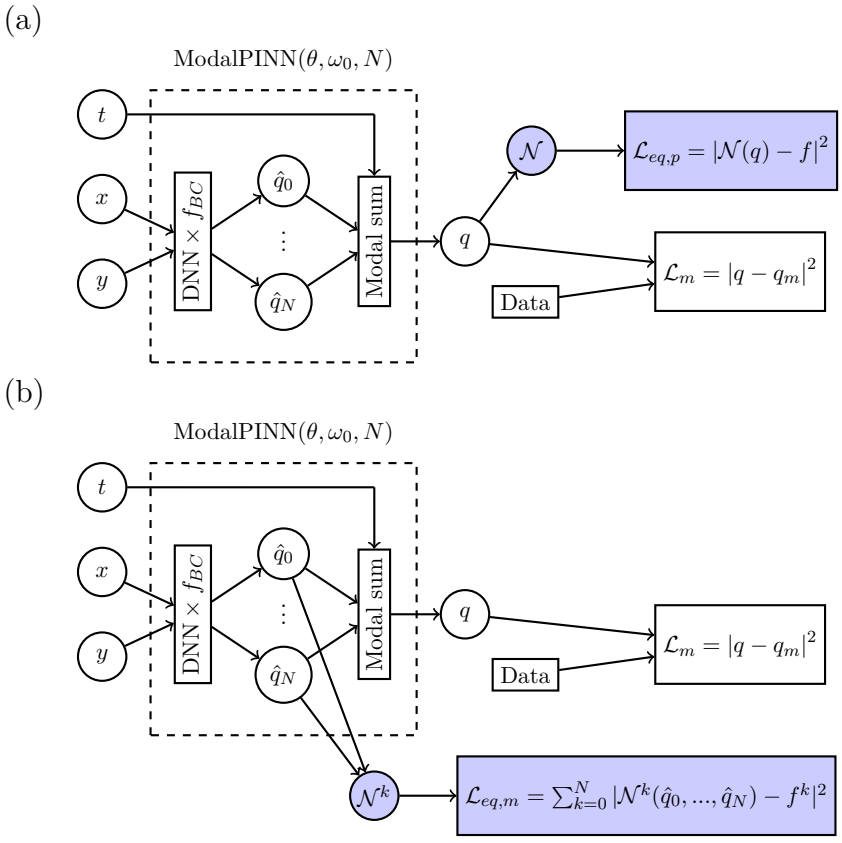


Figure 4.2 Schematic structure of the ModalPINN with access to mode shapes as well as spatio-temporal solution to compute partial-differential equations: (a) in the physical space; (b) in modal space.

Algorithm 2: Algorithm to create and train a ModalPINN

Input: Hyper-parameters of the Neural Network (size, σ) and optimisation (method, learning rate η, training limit ...)
Input: Number of modes N and fundamental frequency ω_0
Result: Modal decomposition encoded in a PINN
Construct the structure of a dense Neural Network $x, y; \theta \rightarrow NN(x, y; \theta)$;
Apply Prior-Dictionary to compute mode shapes : $\hat{q}_0, \hat{q}_1, ..., \hat{q}_N$;
Construct the modal sum : $q(x, y, t) = \sum_{k=0}^{N} \hat{q}_k(x, y) e^{ik\omega_0 t}$;
Construct the fitting part of the loss function $\mathcal{L}_m [V_m; \theta]$;
Construct equation penalisation loss $\mathcal{L}_{eq} [V_{in}; \theta]$;
Construct the total loss function for training $\mathcal{L} = \mathcal{L}_m + \mathcal{L}_{eq}$;
Prepare training data set V_{in}, V_m;
Initialise the parameters of the model θ;
while *training limit is not reached* **do**
  Prepare the batch $\tilde{V}_{in}, \tilde{V}_m \subset V_{in}, V_m$;
  Compute the loss $\mathcal{L}\left[\tilde{V}_{in}, \tilde{V}_m; \theta\right]$;
  Compute loss derivatives $\frac{\partial \mathcal{L}}{\partial \theta}\left[\tilde{V}_{in}, \tilde{V}_m; \theta\right]$;
  Update parameters θ using the optimiser strategy;
end
Compute loss on validation data;

of modal shapes. By projecting the equation on a basis of oscillatory function, one can obtain modal operators :

$$\mathcal{N}^k(\hat{q}_0, ..., \hat{q}_N, \omega_0) = \int_0^{2\pi/\omega_0} \mathcal{N}\left(\sum_{j=0}^{N} \hat{q}_j(\mathbf{x}) e^{ij\omega_0 t} + c.c.\right) \times e^{-ik\omega_0 t} dt, \tag{4.15}$$

as well as modal forces obtained with a similar projection of the global forcing $\mathbf{f}(x, t)$ on the k^{th} frequency, as obtained for $\mathcal{N}^k$ in equation 4.15

$$f^k(\mathbf{x}) = \int_0^{2\pi/\omega_0} \mathbf{f}(x, t) e^{-ik\omega_0 t} dt. \tag{4.16}$$

The possibility is therefore given to optimise the model on physical equations residuals $\mathcal{L}_{eq,physical}$ as formulated in equation 4.5 or by using residuals of modal equations $\mathcal{L}_{eq,modal}$, as illustrated in Figure 4.2. This part of the loss function may be formulated as follows

$$\mathcal{L}_{eq,modal} = \sum_{k=0}^{N} \frac{1}{N_{in}} \sum_{\mathbf{x}_{in} \in V_{in}} \left| \mathcal{N}^k(\hat{q}_0(\mathbf{x}_{in}), ..., \hat{q}_N(\mathbf{x}_{in})) - \mathbf{f}^k(\mathbf{x}_{in}) \right|^2, \tag{4.17}$$

and for the purpose of conciseness it will be referred as $\mathcal{L}_{eq,m}$ (and $\mathcal{L}_{eq,physical}$ as $\mathcal{L}_{eq,p}$).

4.4 Laminar vortex-shedding around a cylinder : a non-linear test case for ModalPINN

We consider a two dimensional incompressible flow over a cylinder, where non-linear vortex shedding is known to occur when a critical Reynolds number is reached. In its dimensionless form, the diameter $d = 1$, the horizontal inflow is the typical velocity scale $(u_\infty, v_\infty) = (1, 0)$. In the present case, the Reynolds number is set at $Re = 100$. In this regime, periodic oscillations of velocity (u, v) and pressure p happen at a Strouhal number $St = \frac{fd}{u_\infty} \approx 0.17$ as documented by Fey et al. [113] (the relative error on the Strouhal with the proposed fitting is estimated at 3×10^{-4}). Numerical data are provided by Boudina et al. [3] for a 2D simulation of the incompressible flow over a fixed cylinder and are available for download [114]. These are obtained using the finite element solver Cadyf [115] that performs hp-adaptive backward differential formula methods [116] in order to keep the local truncation error under a given threshold.

Whereas the simulation domain in Boudina et al. [3] is a rectangle of $(x, y) \in [-40, 120] \times [-60, 60]$, the domain used for the ModalPINN reconstruction covers a limited area defined by $(x, y) \in [-4, 8] \times [-4, 4]$ as depicted in Figure 4.3a. In time, the simulation data used for reconstruction covers approximately 3 oscillation periods with 201 equally spaced time steps.

In order to impose boundary conditions on the cylinder, the following prior dictionary, as defined in equation 4.6, is used for velocities u and v

$$f_{BC}(x, y) = \tanh\left[\gamma\left(r - r_c\right)\right], \tag{4.18}$$

$$h(x, y) = 0, \tag{4.19}$$

where $r^2 = \left(x - x_c\right)^2 + \left(y - y_c\right)^2$ with $(x_c, y_c) = (0, 0)$ being cylinder's coordinates and $r_c = 1/2$ its radius. The slope of f_{BC} near the boundary is defined by the factor γ. In the present case $\gamma = 5$ which is a compromise between a short transition zone and finite gradients. This function is depicted in Figure 4.3b along its profile on centre line.

The equations which are to be solved by minimising the residuals are given by the three differential operators $\mathcal{N} = (\mathcal{N}_{div}, \mathcal{N}_x, \mathcal{N}_y)$ of the unknown $\mathbf{q} = (u, v, p)$. They stand for

conservation of mass (4.20) and momentum (4.21,4.22):

$$\mathcal{N}_{div}(\mathbf{q}) = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{4.20}$$

$$\mathcal{N}_x(\mathbf{q}) = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - Re^{-1}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0, \tag{4.21}$$

$$\mathcal{N}_y(\mathbf{q}) = \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - Re^{-1}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0. \tag{4.22}$$

The associated modal operators for equations 4.20, 4.21 and 4.22 are respectively noted as $\mathcal{N}^k = \left(\mathcal{N}^k_{div}, \mathcal{N}^k_x, \mathcal{N}^k_y\right)$. Modal representation of mass conservation writes as

$$\mathcal{N}^k_{div} = \frac{\partial \hat{u}_k}{\partial x} + \frac{\partial \hat{v}_k}{\partial y}, \forall k \in [\![0, N]\!]. \tag{4.23}$$

Momentum balance along the x axis of the k^{th} mode writes as

$$\begin{aligned}\mathcal{N}^k_x =& (ik\omega_0)\hat{u}_k + \frac{\partial \hat{p}_k}{\partial x} - Re^{-1}\left(\frac{\partial^2 \hat{u}_k}{\partial x^2} + \frac{\partial^2 \hat{u}_k}{\partial y^2}\right) + \sum_{l=0}^{k}\left(\hat{u}_l\frac{\partial \hat{u}_{k-l}}{\partial x} + \hat{v}_l\frac{\partial \hat{u}_{k-l}}{\partial y}\right) \\ &+ \sum_{l=k+1}^{N}\left(\hat{u}_l\frac{\partial \hat{u}^*_{l-k}}{\partial x} + \hat{u}^*_{l-k}\frac{\partial \hat{u}_l}{\partial x} + \hat{v}_l\frac{\partial \hat{u}^*_{l-k}}{\partial y} + \hat{v}^*_{l-k}\frac{\partial \hat{u}_l}{\partial y}\right), \forall k \in [\![0, N]\!],\end{aligned} \tag{4.24}$$

where $\hat{u}^*_k$ stands for the complex conjugate of the k^{th} modal component of u. And similarly for the y component of the momentum equation, the modal operator is obtained with

$$\begin{aligned}\mathcal{N}^k_y =& (ik\omega_0)\hat{v}_k + \frac{\partial \hat{p}_k}{\partial y} - Re^{-1}\left(\frac{\partial^2 \hat{v}_k}{\partial x^2} + \frac{\partial^2 \hat{v}_k}{\partial y^2}\right) + \sum_{l=0}^{k}\left(\hat{u}_l\frac{\partial \hat{v}_{k-l}}{\partial x} + \hat{v}_l\frac{\partial \hat{v}_{k-l}}{\partial y}\right) \\ &+ \sum_{l=k+1}^{N}\left(\hat{u}_l\frac{\partial \hat{v}^*_{l-k}}{\partial x} + \hat{u}^*_{l-k}\frac{\partial \hat{v}_l}{\partial x} + \hat{v}_l\frac{\partial \hat{v}^*_{l-k}}{\partial y} + \hat{v}^*_{l-k}\frac{\partial \hat{v}_l}{\partial y}\right), \forall k \in [\![0, N]\!].\end{aligned} \tag{4.25}$$

Penalisation of equations is conducted on a randomly generated sampling of points V_{in}. Different strategies of space sampling may be defined. From the basic uniform sampling to a sampling adapted to the solution's local complexity, the final choice depends on a compromise between calculation speed and precision. A 2 zones sampling is used here. It consists in distributing 80 % of points uniformly and concentrating the last 20 % within a given

distance around the cylinder, as depicted in Figure 4.3c. By doing this, the relative weight of the residuals located in the boundary layer of the cylinder increases in comparison of those in the rest of the fluid domain. Thus, the shear layer, its detachment and near pressure field are expected to be more accurate which should lead to an increased precision in the estimation of the forces. This spatial sampling has been used for results presented in Figures 4.8, 4.9, 4.10 and 4.12. However, results depicted in Figures 4.4, 4.5 and 4.11 used a uniform spatial sampling optimised for the validation of the equations in the domain. Those two different strategies are also linked with limitations in the number of penalisation points due to memory overflow problems. But from a theoretical point of view, the results are expected to converge through a similar solution as the number of penalisation points increases thanks to larger and better distributed computational resources.

4.5 Results

In this section, results on several training configurations are presented from the one with the largest training data to cases with sparse and flawed information. The first part aims at testing how a ModalPINN performs in comparison to a classical PINN. It also provides some insights about the use of modal equations. The last sections highlight the ability of ModalPINN to address ill-posed problems in simulated experimental conditions. For each result, run properties are recalled in Table 3.1.

4.5.1 Comparison between ModalPINN and classical PINN approach

To illustrate the simplicity brought by the ModalPINN, a comparison is performed with the classical PINN approach that approximates the entire field $x, y, t \rightarrow q(x, y, t)$ as a symbolic function of three coordinates. As the oscillatory nature of the phenomena is known, a sine function is chosen as the activation function between layers to ease convergence, as it has been done by Raissi et al. [72]. A training is performed with a time limit of 2 hours using equivalent computational resources (see section 4.8 and Table 3.1). Physical equations are used for both the classical PINN and ModalPINN. Time sampling for equations penalisation is performed over the simulation data range since the classic PINN is not able to extrapolate the periodic phenomena outside its trained time range.

In classical numerical simulations like finite elements, dependency of the precision of the

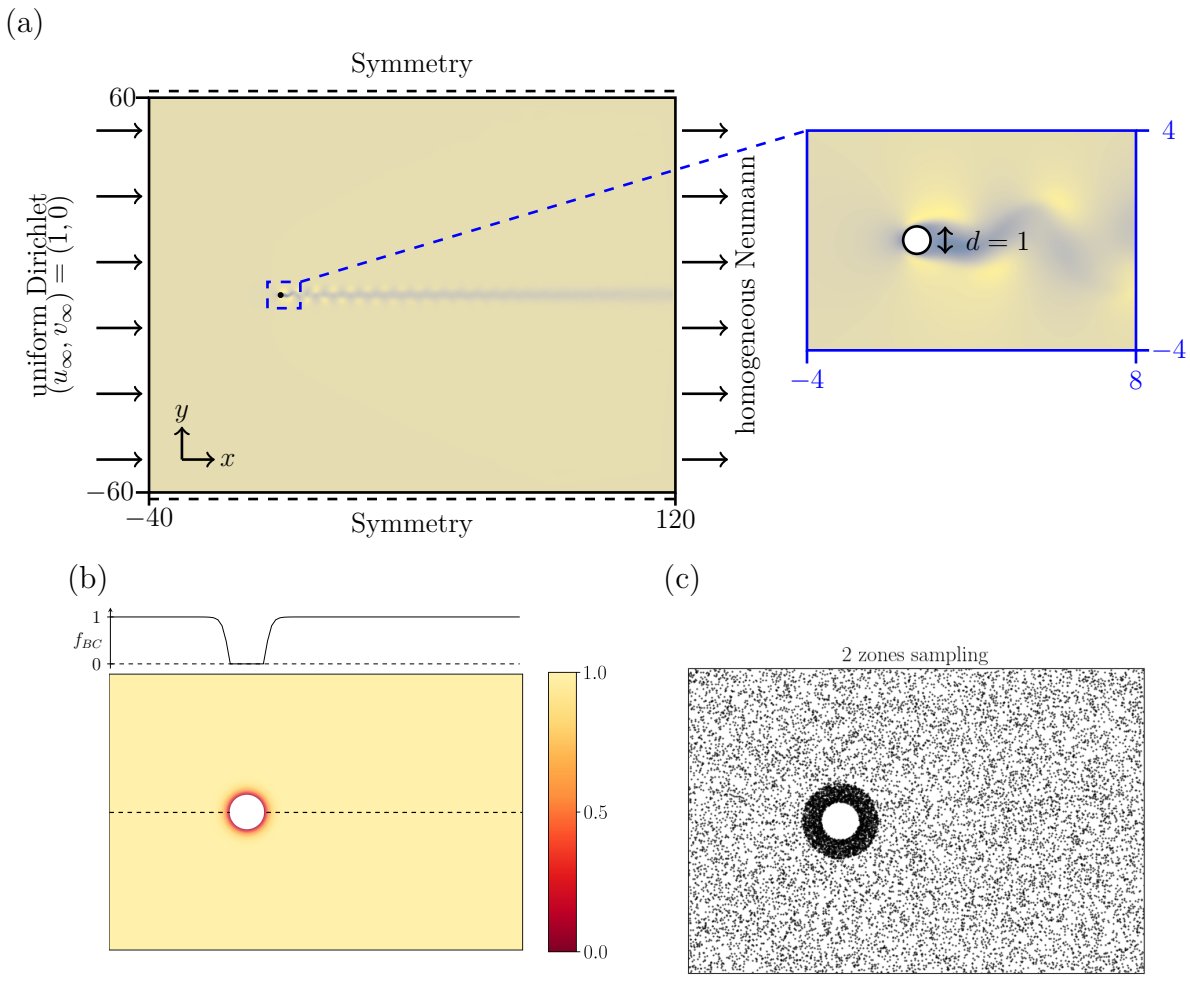


Figure 4.3 (a) Computational domain and ModalPINN reduced domain (blue rectangle) used for training. (b) Prior-Dictionary f_{BC} that enforces boundary conditions $u = v = 0$ on the cylinder border. Profile of f_{BC} along the centre line (dashed line) is plotted above. (c) Distribution of penalisation points for equations in 2 zones configuration with a number of points $N_{in} = 15 \times 10^3$.

solution with the size of the mesh is a key parameter to compare two algorithms. In a PINN, the similar quantity is the number of parameters to optimise. Their influence on precision have been examined for test purpose. To do so, the width of each hidden-layer is multiplied by a factor W_l according to the structure detailed in Table 3.1. Training is performed using physical equations on a set of randomly sampled points in the domain (with uniform probability) and $N_m = 5000$ measurements (u_m, v_m, p_m) at points (x_m, y_m, t_m) randomly picked out from simulation data.

Figure 4.4a depicts how the validation loss at the end of the training varies with the ModalPINN and the classic PINN using different numbers of degrees of freedom and different numbers of modes. Precision of the classic PINN increases with the size of the neural network. On the contrary, the ModalPINN's precision appears insensitive to the number of degrees of freedom if this number is sufficient to allow a correct representation of each mode shape. Loss convergence seems rather linked with the number of modes. Besides, for an equivalent neural network size and training time, there is an observed increase in precision up to 2 orders of magnitude for the ModalPINN. Or alternatively, to approximate the vortex shedding with the same precision as the ModalPINN with $N = 3$, one would need a significantly larger classic PINN than the tested range and with a consequent increase in training time.

To characterise the link between precision and the number of modes in a ModalPINN, the normalised mean squared error (NMSE) is plotted. NMSE is defined as

$$NMSE_q = \frac{\sum_{V_m} \left[q(x_m, y_m, t_m) - q_m\right]^2}{\sum_{V_m} q_m^2}, \tag{4.26}$$

where $q \in \{u, v, p\}$ is the output from the ModalPINN and q_m is data sampled at space-time coordinates (x_m, y_m, t_m). In Figure 4.4b, the NMSE is computed with the result of one training using successively the k^{th} first modes in addition to the steady state $\hat{q}_0$ ($N = 0$). The result indicates that the precision increases with the number of modes following almost a power-law behaviour. This convergence can be compared to other previous studies like Rosenfeld et al. [117] who plotted amplitude decrease of Fourier modes on a similar problem with a two order of magnitude over 6 modes. This is comparable to our result with approximately 2 orders of magnitude between 3 modes, taking into account the square norm.

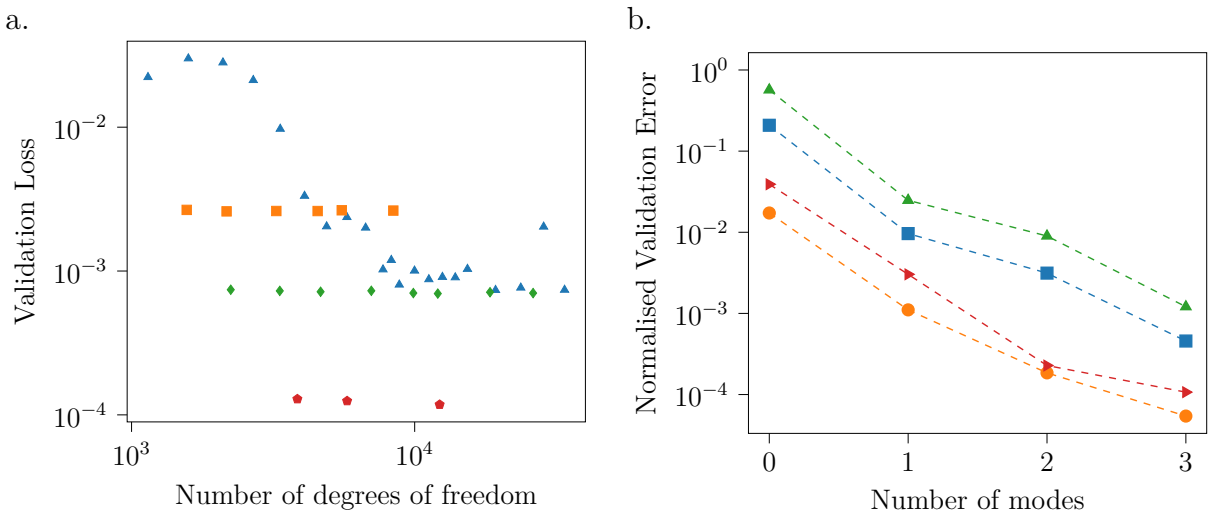


Figure 4.4 With dense data: (a) Comparison of ModalPINN and PINN at given number of degrees of freedom for the same computational time (Classic PINN ▲ , ModalPINN $N = 1$ ■ , 2 ◆ , 3 ⬟); (b) Evolution of normalised validation loss with the number of modes taken into account ($NMSE_u$ ● , $NMSE_v$ ▲ , $NMSE_p$ ▶ , average of the three ■)

4.5.2 Effectiveness of modal and physical equation penalisation

As explained in subsection 4.3.2, two approaches can be used for training a ModalPINN with theoretical knowledge: modal and physical equations. For the incompressible flow over a cylinder, the direct method consisting in penalising mean squared residuals of equations 4.20 - 4.22 is implemented in a concise manner. The disadvantage is that a time sampling is required as well as space sampling. For a classical case, this would mean that to cover the input coordinate space with the same density in every dimension, the amount of points required would increase with the power 3/2 compared to the 2D modal equations.

On the other hand, it is slightly more difficult to implement modal equations 4.23 - 4.25. They usually occupy more place in memory. But in this case, only a spatial sampling is required.

Mode shapes obtained with both equation types are depicted in Figure 4.5. Mode shapes from physical equations in Figure 4.5a are in good agreement compared to those extracted from our reference data and plotted in Figure 4.5c. On the contrary, those obtained with modal equations in Figure 4.5b show some discrepancies starting from mode 2 with the vertical velocity $\hat{v}_2$ being poorly converged in the area downstream. Especially, the third mode did not converge for any of the three fields. Convergence of training loss is compared for both

cases in Figure 4.6a and training with modal equations seems to reach a plateau in fewer iterations than with physical equations. Final values of loss components $\mathcal{L}_m$ and $\mathcal{L}_{eq}$ at the end of the training are summarised in Figure 4.6b and it can be noted that training with physical equation resulted in a reconstruction one order of magnitude more accurate. This leads to the conclusion that computations performed on modal equations might encounter more difficulties to converge properly compared to physical equations. Also as the graph of operation is denser, optimisation is significantly slower as illustrated with the number of iterations performed with each optimiser in the same training time in Figure 4.6b. Thus a larger training time may be required for a similar number of iterations or targeted precision.

4.5.3 Field reconstruction with data from simulated measurements

PINNs have already been proved to work well with dense information, either direct (measurement of velocity and pressure) or indirect (concentration of a passive scalar for instance) as demonstrated by Raissi et al. [72]. Also they were shown to be able to infer hidden variables from equations, such as the pressure field using only velocity measurements [66]. This section aims at evaluating the ability of ModalPINNs to deal with very sparse and asymmetrical data distributed in a simulated experimental framework. The purpose of the following sub-sections is to quantify ModalPINN robustness when confronted with added noise and delay, which are likely to occur in an actual experiment.

As depicted in Figure 4.7, the set-up consists in 4 sections of 10 data points where a time signal of velocity (u, v) is sampled (201 points in time covering approximately 3 periods). Such a set-up simulates an array of pitot or hot-wire measurements found in a typical laboratory experiment. The first section, which is upstream, is located at $x = -3$ starting from the centre of the cylinder. Then, the three sections downstream are respectively at $x = 1$, 2 and 3. Then on the border of the cylinder, 30 points equally distributed on its perimeter provide information about pressure, as would do embedded pressure sensors or taps. The assumption is made that from these measurements, the fundamental frequency can be obtained with a fast Fourier transform. It is therefore fixed in the ModalPINN at the beginning of training.

Results of one training using 3 oscillating modes and physical equations are presented in Figures 4.8, 4.9 and 4.10. A comparison at a given time step of predicted velocity and pressure fields with data from simulations is presented in Figure 4.8a and absolute difference remains small in a large area around the cylinder and in its near wake. The validation loss for different

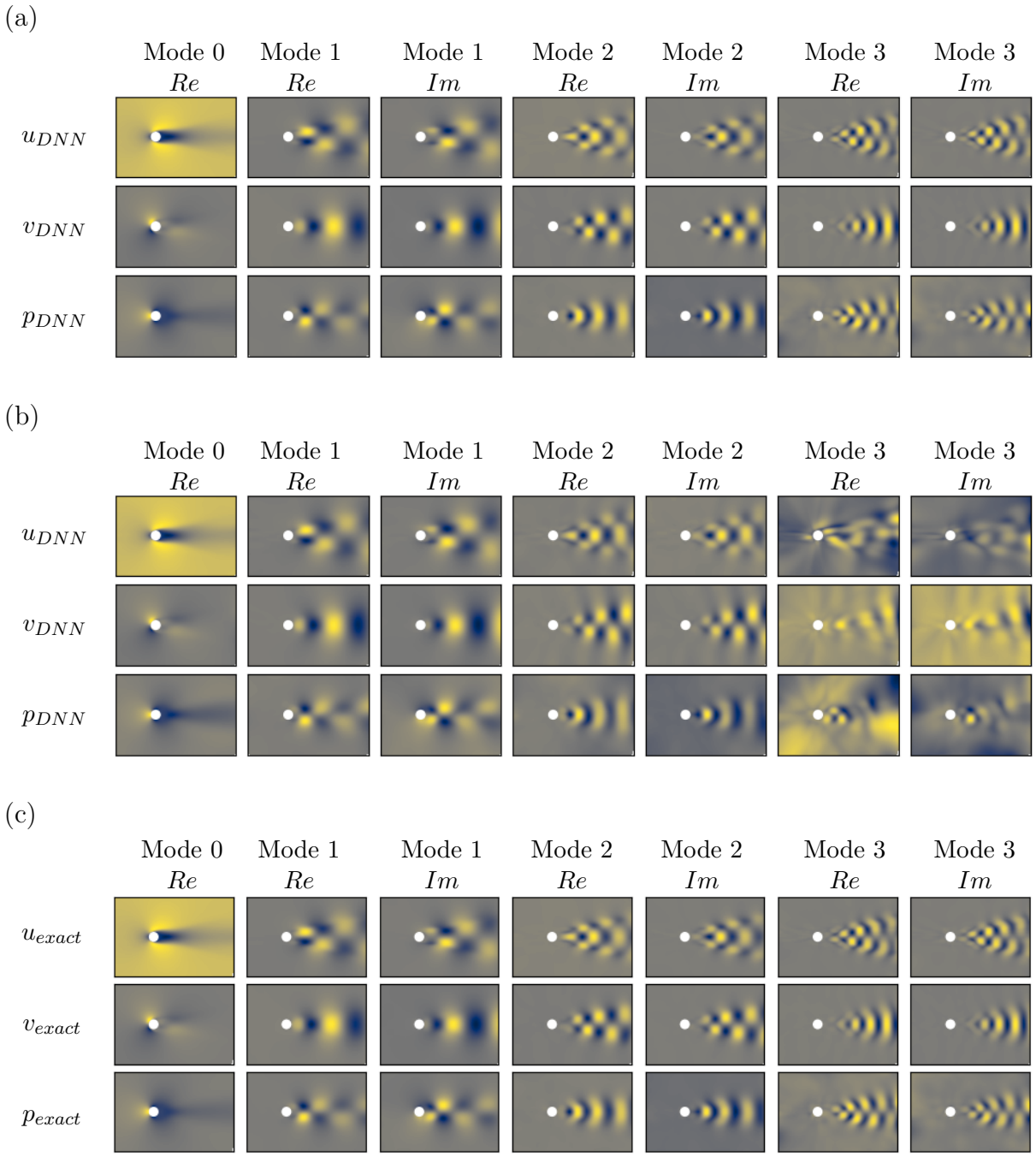
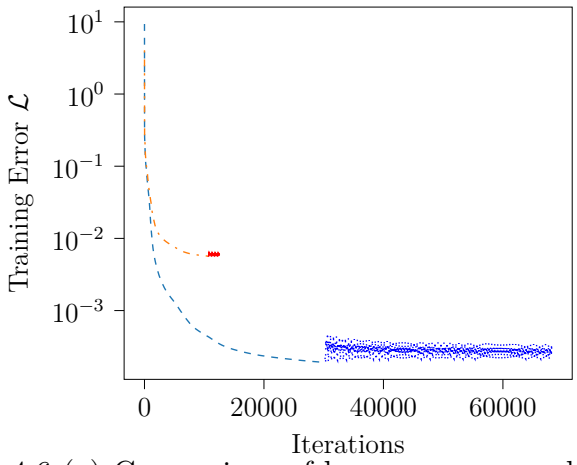


Figure 4.5 Mode shapes computed with: (a) physical equations; (b) modal equations using $N_m = 5 \times 10^3$ dense data with $N = 3$ modes. (c) Comparison with the mode shapes obtained directly from the complete set of reference data.

(a)



(b)

Loss	Physical	Modal
$\mathcal{L}_{eq,p}$	2.2	5.6×10^{-3}
$\mathcal{L}_{eq,m}$	1.4×10^{-4}	1.9×10^{-3}
$\mathcal{L}_m$ (Train.)	1.5×10^{-4}	4.3×10^{-3}
$\mathcal{L}_m$ (Valid.)	1.2×10^{-4}	4.4×10^{-3}
It. L-BFGS-B	30 233	10 655
It. Adam	37 969	1943

Figure 4.6 (a) Comparison of loss convergence during training and (b) final values of losses and numbers of iterations summarised in the Table for both runs using dense measurements and physical or modal equations. For the convergence with physical equations (respectively modal equations), L-BFGS-B optimiser - - - (resp. -·-·-) is used first before iterations with Adam ····· (resp. ——) up to the end of the allowed training duration.

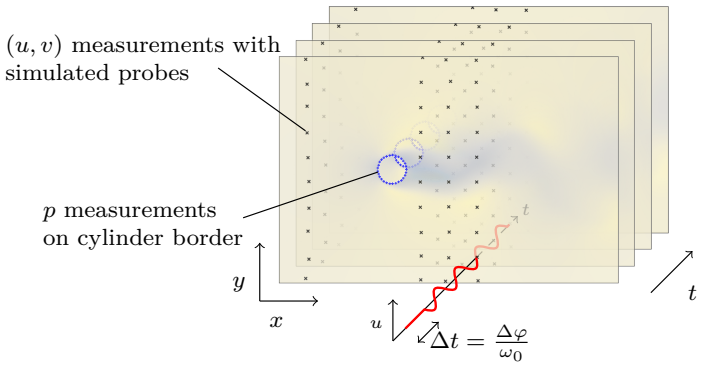


Figure 4.7 Locations of the simulated probes with velocity data points u and v (×) and pressure sensors p (+). The practical problem of setting the time origin while sampling data at several locations is illustrated with a shift in the out-of-plane direction of a time signal (∿).

numbers of modes is shown in Figure 4.8b and may be compared to Figure 4.4b, especially for the convergence of high order modes.

Space averaging of equations residuals is performed for several time steps in Figure 4.9a for each of the three equations 4.20-4.22. Periodicity of these signals is a direct consequence of the enforced periodicity of ModalPINN. The residuals of both momentum equations are of the same order of magnitude, whereas the continuity equation is better satisfied. Nonetheless, these signals stay at values lower than 2×10^{-4}. This empirically happens to be a very acceptable value for equations residuals based on prior qualitative knowledge of cases where the exact and reconstructed flows can not be easily distinguished. The spatial distribution of residuals at a given time is presented in Figure 4.10. Error is mainly located in the wake where most of the flow unsteadiness occurs. Interestingly, the high-gradient region around the cylinder has low residuals. This is a direct consequence of the 2 zones penalisation distribution. In case of a uniform space sampling of V_{in} (not shown), levels of errors are slightly higher near the cylinder border.

Prediction of unsteady forces are plotted alongside simulation data in Figure 4.9b. The horizontal and vertical forces are inferred accurately with normalised root mean square errors of 9.8×10^{-4} and 6.1×10^{-3} respectively.

4.5.4 Noise sensibility

Measurement noise is part of the experimental process. To test the ability of the ModalPINN to deal with random perturbations, a Gaussian noise $\mathcal{N}(\mu, \sigma)$ is considered with an average $\mu = 0$ and a standard deviation σ. The choice of a zero drift μ (which can be related to a kind of systematic error) and the Gaussian distribution may depend on the nature and context of the measure, as discussed by Coleman et al. [118] for instance. The assumption is made that this is a common framework representative of real life. This noise is added to our reference data $p^{cyl}, u^{probe}, v^{probe}$ extracted at probe locations from numerical simulations. One obtains the new training data that are artificially flawed

$$\begin{aligned} p^{cyl}_{noisy} &= p^{cyl} + \epsilon_p, \\ u^{probe}_{noisy} &= u^{probe} + \epsilon_u, \\ v^{probe}_{noisy} &= v^{probe} + \epsilon_v, \end{aligned} \tag{4.27}$$

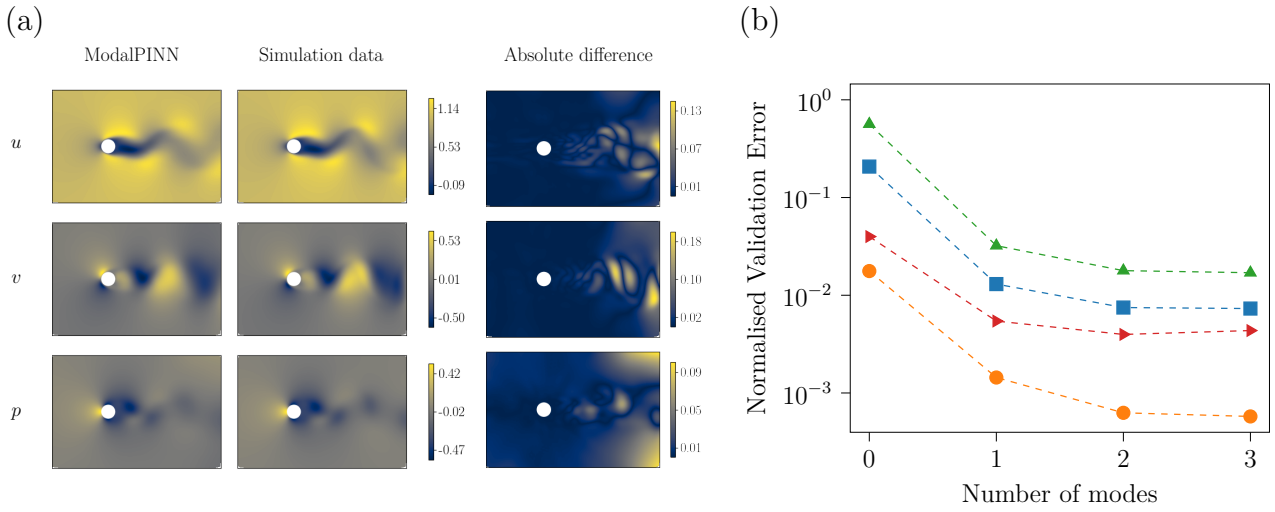


Figure 4.8 Simulated experimental measurements: (a) Comparison of reconstructed fields (with $N = 3$) with simulated data at a given time-step $t = 400$s. (b) Evolution of normalised validation loss with the number of modes ($NMSE_u$ ■ , $NMSE_v$ ▲ , $NMSE_p$ ◀ , average of three ■)

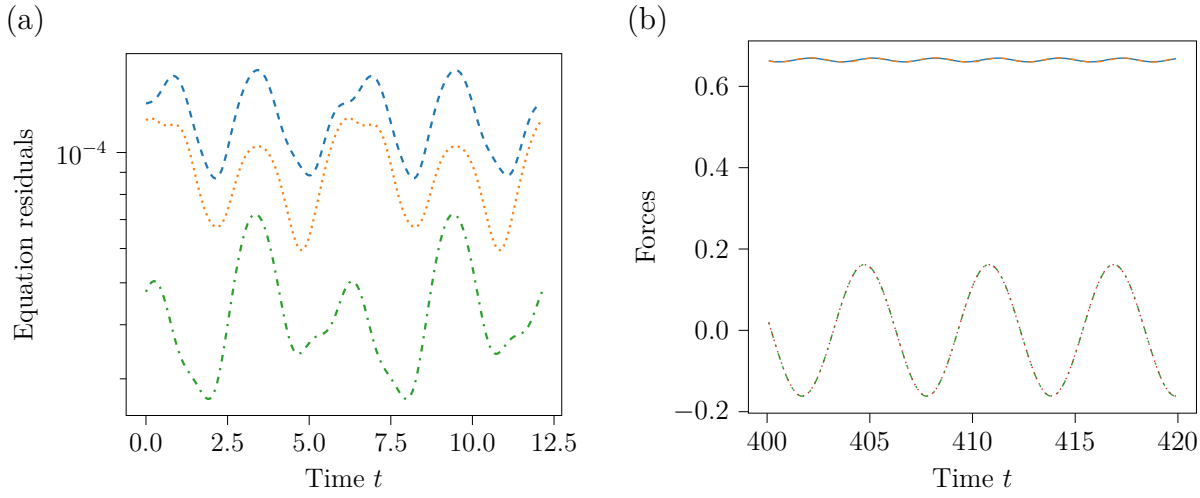


Figure 4.9 (a) Evolution in time of mean squared residuals for equations 4.20 (-·-·-), 4.21 (- - -) and 4.22 (·····). (b) Unsteady forces on cylinders obtained with the ModalPINN (drag F_x - - - and lift F_y ·····) and from simulation data (drag F_x —— and lift F_y -·-·) using simulated experimental measurements. Force curves are indistinguishable.

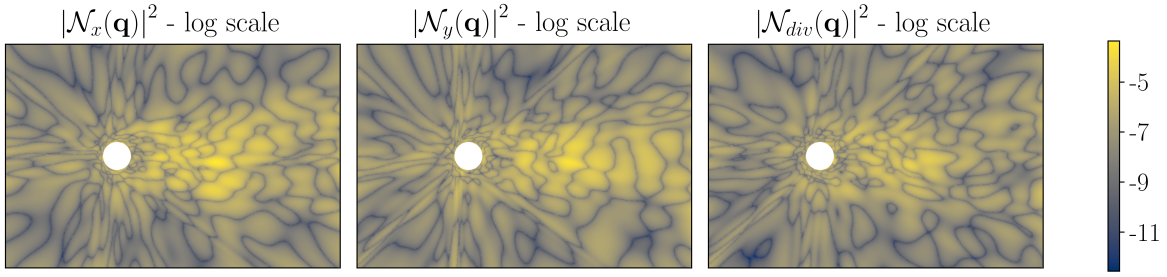


Figure 4.10 Space distribution of residuals on physical equations 4.20, 4.21 and 4.22 at a given time-step using sparse measurements.

where $\epsilon_p, \epsilon_u, \epsilon_v \sim \mathcal{N}(0, \sigma)$ at each time step and are independent from each others. Noiseless simulated probe measurements data are directly replaced by $p^{cyl}_{noisy}, u^{probe}_{noisy}$ and v^{probe}_{noisy} in the fitting part of the training alongside minimisation of physical equations residuals. In addition, the uncertainty on probe coordinates has been neglected in the present study but could be taken into account with a similar formalism.

To test the influence of noise level σ, several runs with $N = 2$ oscillating modes have been carried out on a similar configuration (see section 4.8 and Table 3.1) but with σ taking values between 1×10^{-4} and 1×10^{-1}. The same noise level is added to velocity and pressure since data are physically normalised and, therefore, of order of magnitude 1. Approximately 30 jobs have been executed for each noise level so that statistical quantities that are computed can be representative. For each job, the three parts of the training loss as well as the validation loss at the end of training are presented in Figure 4.11.

Residuals for the fitting of noisy velocity and pressure time signals are depicted in Figures 4.11a and 4.11b. Velocity residuals have low values (under $\sim 5 \times 10^{-4}$) for noise levels smaller than 1×10^{-2} and then grow quickly with a small dispersion. Noisy pressure residuals are slightly more dispersed in the logarithmic scale for $\sigma < 1 \times 10^{-2}$ but still at low levels with an average around 1×10^{-4} and a median between 1×10^{-6} and 1×10^{-5} before increasing with σ as a power law. For both these noisy measurements, there is a threshold from which these fitting errors increase linearly with the square of σ. Taking into account that loss on fitting to data error is a mean square difference, this is equivalent to a linear increase of absolute measurement error with noise level.

Equations loss plotted in Figure 4.11c shows more variability in the distribution of residuals. For nearly each level of noise there are examples of outputs that had an error of order of magnitude one, which is abnormally high and a sign of poorly converged training. This may be due to a wrong direction of optimisation or initialisation and also to the limited allocated time. Nonetheless, more than half of the results are kept at values near 1×10^{-3} which, qualitatively, appears to be a very acceptable value at which usually the differences between the exact and reconstructed flow are difficult to discern for this case. Moreover, no clear increase can be noted with the level of noise for all statistical quantities of equations residuals.

In the end, validation loss is the quantity of interest and reveals the quality of flow reconstruction. As presented in Figure 4.11d, similar conclusions as for the equations loss applied here with a small number of poorly converged results but a median that stays at low values for all noise levels. There is no discernible trend linking noise level to validation loss in the presented statistical quantities.

In the presence of noise, the minimisation of fitting data and the equation residuals become incompatible. Favouring the equation residuals despite the fitting error results in a large value of $\mathcal{L}_m$, and vice versa if the data are prioritised. As both terms $\mathcal{L}_m$ and $\mathcal{L}_{eq}$ have a similar weight in the total loss function $\mathcal{L}$, this choice is not encoded explicitly. From results in Figure 4.11, it seems that only the minimisation of residuals drive the NN learning. Therefore the corruption of data does not affect significantly the validation error. This can be understood as a proof of robustness in the considered range of perturbation.

4.5.5 Data resynchronisation

In the situation where a measurement is performed successively at different locations for a given recording duration with a probe, the initial condition of each measurement point is different. For a periodic phenomenon, this can be treated as an unknown phase shift for every data series, as depicted in Figure 4.7. To account for this phase shift in asynchronous measurements, a variable delay for each time series is determined through optimisation alongside the neural networks coefficients. To test this solution, time series of velocities from a simulated probe have been desynchronised with a random delay following a uniform law $\Delta t \sim U(0, T)$. Consequently, 40 scalar variables, one for each location, are added to the optimisation process. However, pressure measurements on the cylinder border are kept synchronised for two main reasons: the need for a constant initial phase shift to compare with validation data and

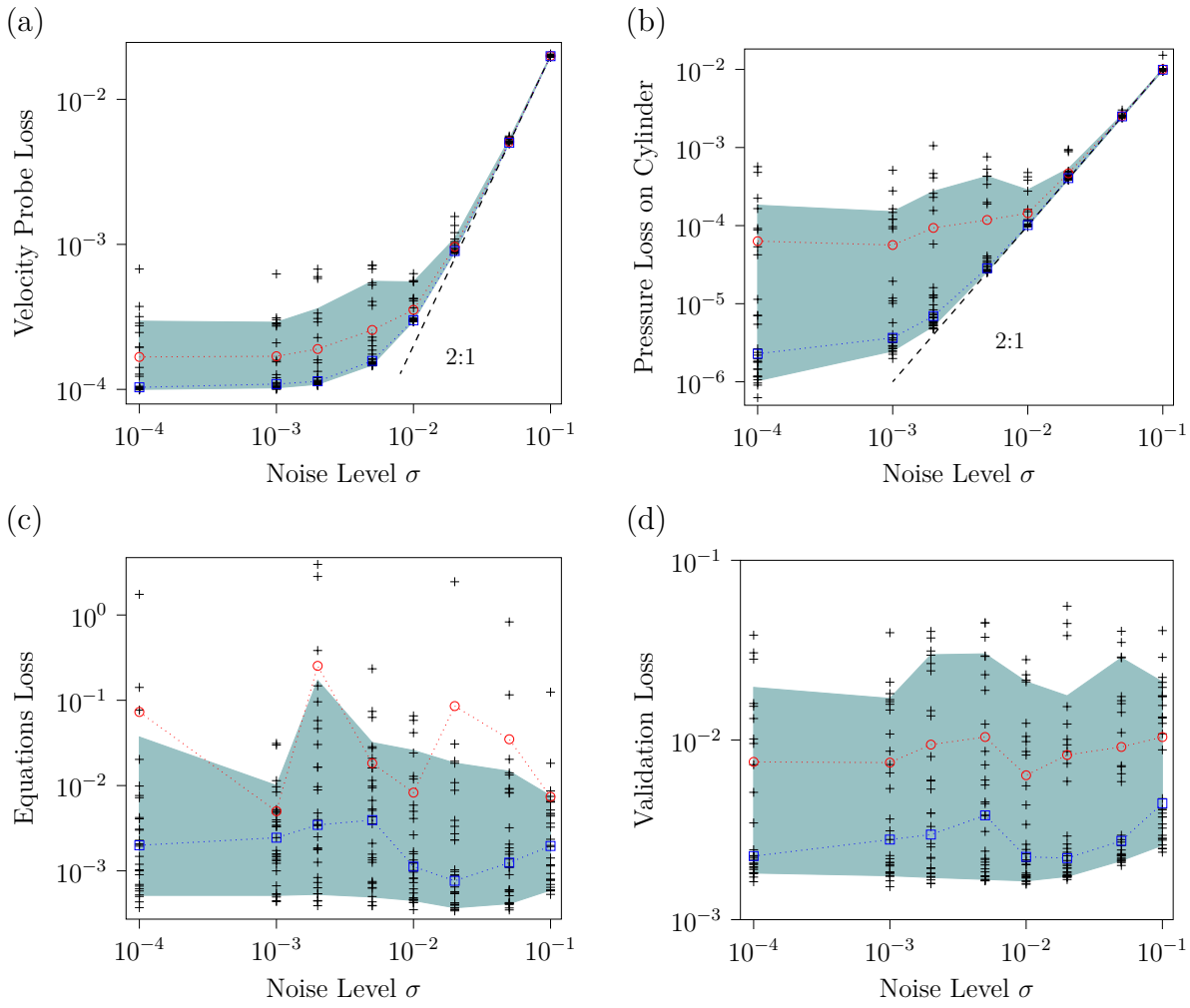


Figure 4.11 Loss residuals dependency with an input noise of standard deviation σ in measurements data. Each job result is depicted with a +, and for each sampling with the same level of noise, the average ··○·· is given as well as the envelope ($10-90\%$ in ■) and the median ··□··. For velocity and pressure fitting error (a and b), the expected 2:1 slope for a square norm is plotted (- - -).

because this could be carried out experimentally by parallel and synchronised pressure probes.

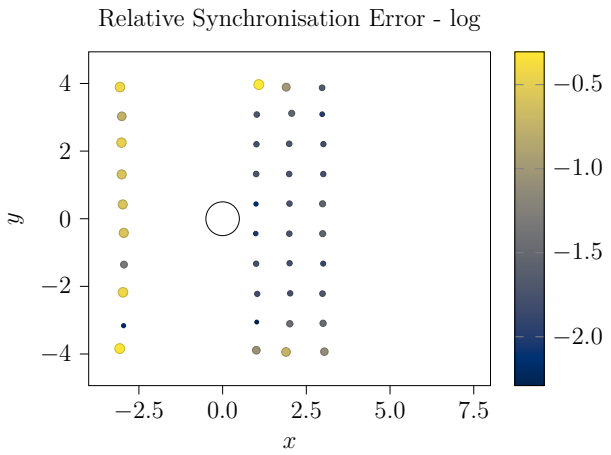
As the array of artificially added delay for every time signal of velocity Δt_{exact} is stored, it can be compared to the delays found through optimisation Δt_{found} at the end of the training. The absolute difference of these two delays, centred in the interval $[-T/2, T/2]$ and then normalised by T quantifies how precisely this time shift is computed. The resynchronisation error is bounded on the interval $[0, 1/2]$ and its value is plotted at each probe location in Figure 4.12a. The error magnitude is shown through the color contours with a logarithmic scale as well as qualitatively represented by each point size.

From Figure 4.12a, two types of outlooks stand for re-synchronisation process. In the wake of the cylinder, the relative error on phase shift converges to approximately 1%. But on the upstream sensors and at downstream probe positions that are the most distant from centre line $y = 0$, residuals remain large. These points are located in areas where oscillatory phenomena are of very low amplitude. This can be seen in unsteady mode shapes. This means that there is a lack of phase information in these ranges which explains why the original phase shift can not be recovered. Fortunately, in the area of interest, phase shift is well retrieved. This can be seen in Figure 4.12b where the obtained mode shapes are compared to reference data. Some differences may be noted with for instance, the second mode that has a slightly lower amplitude than the reference. But overall, there is an acceptable agreement between flow reconstruction and simulation data.

4.6 Discussion

Differences in residual levels and convergence rates of higher frequency modes in the presence of dense data (figure 4.4b) or sparse data from simulated measurement (figure 4.8b) can be explained with two arguments. Extrapolating flow field downstream the last measurement provided for training without any boundary condition on the outlet can be considered as an ill posed problem since there is a lack of information. The neural network simply optimises what works best with the information it has. Whereas in the dense measurement problem, there is information equally distributed in space and time, even if the space between two points can be larger than the length scale of the third mode shape. This transforms the extrapolation problem into an interpolation one, both with physical regularisation. Following these results, it would be interesting to use a PINN for flow extrapolation in the area close to a wall where measurements with hot-wire or PIV are limited [108, 109] using the no-slip

(a)



(b)

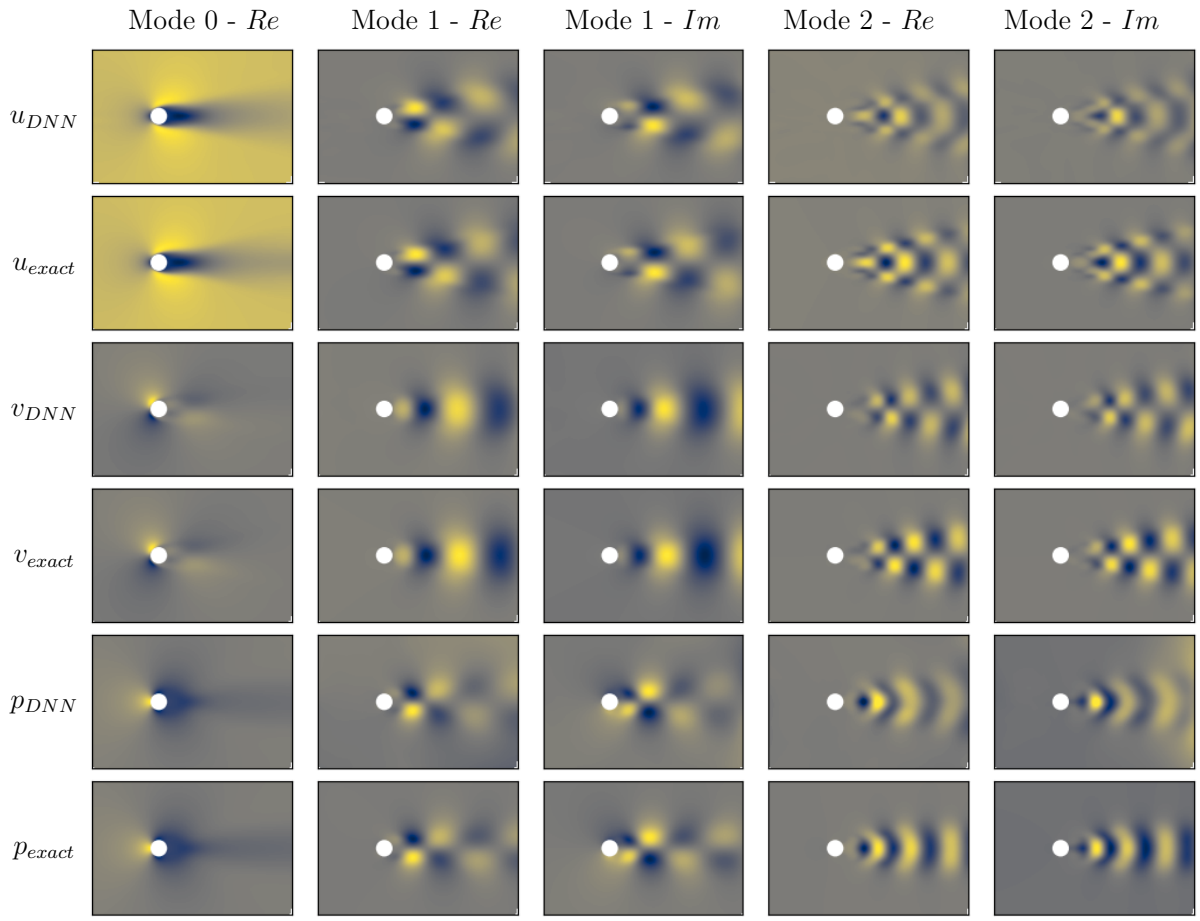


Figure 4.12 (a) Distribution of the residual relative error of synchronisation $\frac{|\Delta t_{\text{found}} - \Delta t_{\text{exact}}|}{T} < 1/2$ (depicted with a log scale). The reconstructed mode shapes are depicted at (b) and compared to the reference data from simulations.

conditions in addition to the physical regularisation. This could help predict local wall shear stress with a better accuracy, which is of interest for drag estimation or for application in bio-medical applications. For instance, Arzani et al. [119] use PINNs to estimate near-wall blood flows and wall shear stress which are linked to cardiovascular diseases.

The differences of convergence of higher frequency mode shapes can also be explained from a computational point of view: higher frequency modes display smaller structures than low frequency modes. In addition to the fact that these mode shapes are more complicated to approximate and thus requires larger neural networks, Navier-Stokes equations should be penalised on points that are distributed with an averaged spacing smaller than the typical wavelength. This leads to significantly increased memory requirements. In our computations this has been a limit due to the availability of RAM on the GPU (16GB in our case). For a $N = 3$ modes computation, the limit in N_{in} before an out of memory (OOM) error was to be found around 10^3 points, which in that case is a small value and may not be fully adequate to thoroughly capture the steep gradients associated with small wavelength mode shapes. Besides, this can not bet fully addressed by batch processing because of the required loading time of training data after a few number of iteration. This could be overcome by using GPU with larger RAM or by splitting computation points of one optimisation iteration between different GPUs, which would require a low-level implementation.

In the test case of laminar vortex shedding using data at different levels of sparsity and quality, the penalisation of modal equations appeared to perform worse than physical equations penalisation, even considering that there is only a 2 dimensional input range to cover instead of 3D time-space coordinates, as illustrated with results in Figures 4.5 and 4.6. This seems to be a consequence of non-linearities in the momentum equations that lead to sums of cross-terms at different frequencies. This makes the convergence of the solution more dependant on the number of modes and their accuracy, whereas physical equations deal with this balancing more directly and seams less affected by the truncation. Nonetheless modal equations could be of interest for linear phenomena where mode shapes might be uncoupled and computed separately, with potential applications in solid mechanics or electromagnetism for instance.

A variation of the ModalPINN structure could be considered, especially for linear phenomena, where $q(\mathbf{x}, t) = \sum_{k=0}^{N} a_k \hat{q}_k(\mathbf{x}) e^{ik\omega_0 t} + c.c.$ where $\hat{q}$ are mode shapes that can be computed previously and normalised $\|\hat{q}_k\|_\Omega = 1$ independently with modal equations. The modal coefficients a_k can be adjusted depending on the excitation. This can ease transfer learning

of once converged mode shapes to different loading configuration as in linear elasticity or electromagnetism. Also, an unknown growth exponent could be added in the presence of developing modes and optimised concurrently.

4.7 Conclusion

In this paper we presented an architecture of PINNs that directly approximates Fourier mode shapes. Space-time output is recovered thanks to a modal sum directly encoded in the neural network graph operation, keeping all the advantages of classical PINN while considerably reducing the required size for a similar target precision. Finally this ModalPINN structure proved to be robust to some data flaws, which makes it an efficient tool for helping academic and industrial researcher with data processing from their experimental work. This formalism can be directly extended outside of the context of fluid mechanics, as a support of digital image correlation or laser displacement measurement in solid mechanics while performing harmonic response for instance.

Some work remains in order to combine this technique with existing advances in PINN in order to increase robustness in the optimisation process as well as its efficiency. It goes alongside with developments in new hardware solution and implementation of libraries that better take advantage from computational resources. Finally, as we introduced this topic in the context of vortex shedding [72], extension to elastic-solid deformation of fluid-structure couplings or stability analysis could be interesting leads for future work.

4.8 Technical details

Training and optimisation were performed on the Graham server from Compute Canada. Each job is carried out with the same computation resources consisting in an allocation of 2 CPUs with 50 GB of RAM and a GPU Nvidia T4 (16 GB of dedicated RAM). Jobs are performed with a training limit in total duration. L-BFGS-B optimiser is used though Scipy's interface and stops when a maximum number of iteration is reached or when the difference between two iterations falls under a threshold. Only one batch of penalisation points is used during this part of the training and validation loss computation is not available. Then Adam optimisation is performed with a learning rate equal to 1×10^{-5} and conducted until time limit is reached for the whole job.

All the scripts are written in Python using Tensorflow 1.14.1 and are available on a Github repository [120], as well as data [114] used for training. Dependencies are listed on the Github repository. Properties of every runs mentioned in the results section are summarised in Table 3.1.

4.9 Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number RDCPJ 507814 - 16], of the Tr-FRANCIS members : Electricité de France, Hydro-Québec, Voith Hydro Inc., Andritz Hydro Canada Inc., InnovÉÉ, GE Énergies Renouvelables Canada Inc. We also acknowledge the support of the Institut de l'Energie Trottier and of the Simulation-Based Engineering Science (SBES) program at Polytechnique Montréal. This research was enabled in part by support provided by Calcul Québec and Compute Canada.

Table 4.1 Summary of run properties which results are presented in section 4.5. Lines 1, 2 and 6 describe a group of runs where the size of the NN (factor W_l) or the noise level (standard deviation σ) have been changed. N denotes the number of modes chosen in ModalPINN.

Run ID	NN size for 1 field (activation function)	Equation type	Data type	N_m	N_{in} (sampling strategy)	Training Time (h)	Figures referenced	Validation Loss
1	$[3, W_l, W_l, W_l, 1]$ $10 \leq W_l \leq 60$ (sin)	Physical	Dense	5×10^3	5×10^4 (uniform)	2	ClassicPINN (fig. 4.4a ▲)	From 7.4×10^{-4} to 3.0×10^{-2}
2	$[2, W_l N^*, W_l N^*, N^*]$ $8 \leq W_l \leq 25$ and $N^* = N + 1$ (tanh)	Physical	Dense	5×10^3	50×10^3 ($N = 1$) 15×10^3 ($N = 2$) 12×10^3 ($N = 3$) (uniform)	2	ModalPINN (fig. 4.4a ■ ◆ ⬟ and 4.4b)	From 1.2×10^{-4} to 1.3×10^{-3}
3	$[2, 80, 80, 4]$ (tanh)	Physical	Dense	5×10^3	10×10^3 (uniform)	10	Fig. 4.5a, 4.6	1.2×10^{-4}
4	$[2, 80, 80, 4]$ (tanh)	Modal	Dense	5×10^3	8×10^3 (uniform)	10	Fig. 4.5b, 4.6	4.4×10^{-3}
5	$[2, 80, 80, 4]$ (tanh)	Physical	Simulated measurements	201 time-steps per location	10×10^3 (2 zones)	6	Fig. 4.8, 4.9, 4.10	1.6×10^{-3}
6	$[2, 60, 60, 3]$ (tanh)	Physical	Noisy simulated measurements	201 time-steps per location	15×10^3 (uniform)	4	Fig. 4.11	From 1.5×10^{-3} to 5.6×10^{-2}
7	$[2, 60, 60, 3]$ (tanh)	Physical	Out of sync. simulated measurements	201 time-steps per location	20×10^3 (2 zones)	10	Fig. 4.12	2.8×10^{-3}

CHAPTER 5 ADAPTING PINN TO FLUID-STRUCTURE INTERACTIONS

5.1 Introduction and motivations

5.1.1 Importance of FSI in nature and engineering

Fluid-Structure interactions is at the crossroads of two fields that surround us everywhere. From air to water and blood and more complex fluids in industrial applications, they interact with their container and obstacles. From the solid perspective, rigidity must be compared with the fluid loading. Plane wings are known to bend under air loading when a certain velocity is reached. Francis turbines blades are quite rigid but they also suffer from vibrations especially at some operating modes [6]. In this chapter we aim at developing a methodology that would suit both small and large movements of the fluid-solid boundary $\partial\Omega_{FS}$ using the mesh free asset of PINNs.

Strong couplings between the solid and fluid dynamics appear when the typical timescales of the flow and of the vibrations of the solid are of the same order of magnitude. In that case, both dynamics can not be uncoupled and computations are usually more expensive [4]. Vortex induced vibrations is a typical example of strong coupling, especially when the frequency of the vortex-shedding locks with the vibrations of the solid as illustrated in Figure 5.1b. A well-studied test-case lies in the in-plane oscillations of a cylinder under a two-dimensional laminar flow as defined in Figure 5.1a. There are three important parameters:

- The Strouhal number which is the normalized frequency of vortex shedding $S_t = \frac{fL}{U_\infty}$.
- The mass number that quantifies the ratio of solid and fluid densities $M = \frac{m}{\pi\rho_f D^2/4}$.
- The reduced velocity $U_r = \frac{2\pi U_\infty}{D}\sqrt{\frac{m}{k}}$ which represents the ratio of the solid timescale over the fluid timescale.

The equation of motion for the degrees of freedom of the cylinder are:

$$\frac{\partial^2 x_c}{\partial t^2} + \left(\frac{2\pi}{U_r}\right) x_c = \frac{4}{\pi M} F_x, \tag{5.1}$$

$$\frac{\partial^2 y_c}{\partial t^2} + \left(\frac{2\pi}{U_r}\right) y_c = \frac{4}{\pi M} F_y, \tag{5.2}$$

where F_x and F_y are the two dimensionless forces computed on $\partial\Omega^t_{FS}$. Besides, for $2 \times 10^5 \gtrsim Re \gtrsim 50$, the Strouhal number is known to be $0.22 \gtrsim St \gtrsim 0.12$ [4, 113]. Values of these parameters are specified in the definition of the test-case in the following sub-section.

Numerical complications occur in that type of cases for several reasons. In classical methods, displacement of the frontier $\partial\Omega^t_{FS}$ have a direct impact on the definition of the fluid solution. As discussed in the introduction, re-meshing, IBM and ALE are methods to deal with the change of fluid domain Ω_F at each iteration. For PINNs, the definition of the flow does not rest upon cells, elements or grids that are spatially located. Nonetheless:

1. The set of equation penalization points V_{in} must follow the fluid domain, and thus depends on the time coordinates. However the time dimension is equivalent to other space dimensions in the PINN so V_{in} must adapt dynamically as a space-time function but also during the optimization process.

2. The spatial localization of some phenomena (like the boundary layer) that is encoded implicitly in the NN can be tricky to move concurrently with the FS frontier. As explored with the inverse extension operator, we could think intuitively that approximating the quantities in a fixed reference frame could be easier for the NN, and especially its optimization that could be quicker and more robust.

5.1.2 Current methods for PINN and their limitations

Raissi et al. [72] proposed a method to deal with vortex-induced vibrations of a rigid two-dimensional cylinder mounted on spring. In their configuration, the movement of the cylinder is described only by its transverse displacement $\eta(t)$ which is approximated by a PINN. Then, dense PINN approximates the three flow fields u, v and p. The particularity lies in the fact that they do not map flow variables with the actual coordinates but instead they use a fixed reference frame, relatively to the cylinder's centre.

This change of reference frame appears in an inertial force in the Navier-Stokes equations

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} &= Re^{-1}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} &= Re^{-1}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial^2 \eta}{\partial t^2}, \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0, \end{aligned} \tag{5.3}$$

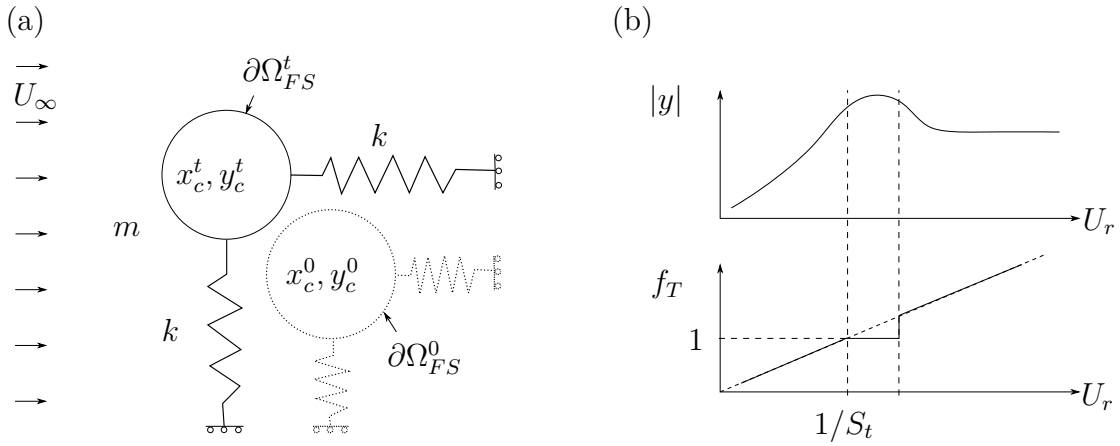


Figure 5.1 (a) Configuration used for the vortex induced vibration test-case. A cylinder (mass m, diameter D) can move in both directions in space and is attached to two identical springs of stiffness k. A uniform flow $(u, v) = (U_\infty, 0)$ far from the cylinder is applied. Figure inspired from Boudina et al. [3]. (b) Illustration of the lock-in effect that occurs when the reduced velocity U_r is near the inverse Strouhal $1/S_t$, an increase in the amplitude of the cylinder's oscillations is observed $|y|$ and the dimensionless frequency of the vortex shedding f_t synchronizes with cylinder's frequency. This Figure is reproduced from De Langre [4].

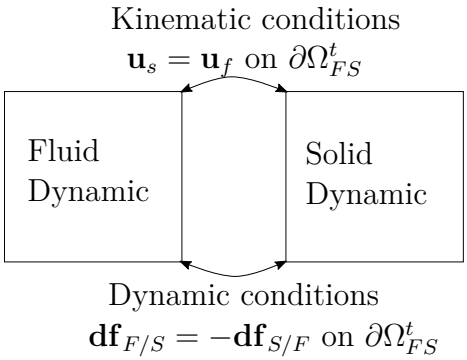


Figure 5.2 Schematizing of the interactions of fluid and solid dynamics through the conditions at the fluid-solid interface $\partial\Omega_{FS}(t)$ that moves with time. Figure inspired from Fig. 3.3 from De Langre [4]

where only the vertical component of the momentum equation is affected. It can be seen that a streamwise displacement could be taken into account with the equivalent of η for the horizontal direction.

In their paper, Raissi et al. are able to infer the forces quite accurately as well as the parameters of a reduced-order model of the oscillations of the spring (damping, stiffness, mass). But it could be complicated to generalize this formalism due to some limitations :

1. The degrees of freedom of the cylinder must be taken into account in the change of reference frame and for the inverse transformation of measurement points from the actual frame to the working frame. This may not be easy in the case of rotations alongside translations, especially in three dimensions.
2. This limits the set of deformations to one rigid body motions only. This formalism can not generalized to elastic reconfigurations for instance, or even for two rigid bodies because the operator that depicts the change of frame must be local.

In the next sections, we present several leads to deal with general elastic deformations of a solid under a flow using extension operators. Test cases are presented in the next subsection. Then the concepts of direct and inverse extension operators, as well as their consequences on the change of reference frame are discussed in section 5.2. Finally we present the encountered limitations with some outlooks for future works.

5.1.3 Presentation of two test cases

During this master thesis we have been using two test cases:

- The reconfiguration of a clamped beam at 90° with the direction of the flow. This is a static case that can be solve for the steady state using symmetric conditions. However no results are presented for the purpose of concision and because the following developments in the second test case are more general.
- Vortex induced vibrations of a cylinder (mass m, diameter D) mounted on spring (stiffness k) in a 2D flow (inlet velocity U_∞). We used data from Boudina et al. [3] that are computed with the Cadyf finite element solver using an Arbitrary Lagrangian Eulerian (ALE) approach. Data used are extracted from a run at $U_r = 5$ where a frequency lock-in is known to occurs. The computational domain and boundary

conditions are identical from those presented in chapter 4 for the vortex shedding, as well as the Reynolds number $Re = 100$. Here the mass number is set to $M = 1$.

5.2 Several approaches around the concept of extension operator

5.2.1 Representation in the actual frame with displacement of training points

Keeping in mind that a PINN is not defined by local elements but by a symbolic mapping of physical coordinates to a quantity, there is no mesh that would need to be moved when the fluid domain is changed (with time and optimization iterations). Nonetheless, there are some consequences for the set of points that penalize residuals of PDE. If this set, that is noted V_{in}, is sampled at the beginning of the training based on the static configuration Ω_f^0, it will not fill completely the fluid domain at each time Ω_f^t: some points will go outside of the fluid domain (for instance inside the solid). Plus, some area in Ω_f^t won't be covered by any penalization points. This situation is illustrated in Figure 5.3 with highlights in red of the initial points that fall into solid domain. Moreover, during the training, before the PINN has converged to the solution, the fluid domain might be deformed even if this is not the final solutions. Still the penalization points should be adapted to this moving fluid domain to enforce the physical regularization.

One way of solving this problem is to move V_{in} points during time and training. This leads to the definition of an extension operator $\boldsymbol{\xi} = (\xi_x, \xi_y)$ which is a function of the space and time coordinates (x^0, y^0, t) in the initial frame Ω_f^0 to the spatial coordinates in the actual frame (x^t, y^t, t):

$$\begin{aligned} x^t &= x^0 + \xi_x \left(x^0, y^0, t\right), \\ y^t &= y^0 + \xi_y \left(x^0, y^0, t\right). \end{aligned} \tag{5.4}$$

In addition, the fluid-solid border is still defined by a symbolic function but moves during time and optimization. With similar notation, $x_{BC}^0(s), y_{BC}^0(s)$ are the coordinates of the initial border as a function of the normalized curvilinear abscissa $s \in [0, 1]$. The deformed border is obtained with a similar transformation using the extension operator $\boldsymbol{\xi}$:

$$\begin{aligned} x_{BC}^t(s,t) &= x_{BC}^0(s) + \xi_x \left(x_{BC}^0(s), y_{BC}^0(s), t\right), \\ y_{BC}^t(s,t) &= y_{BC}^0(s) + \xi_y \left(x_{BC}^0(s), y_{BC}^0(s), t\right). \end{aligned} \tag{5.5}$$

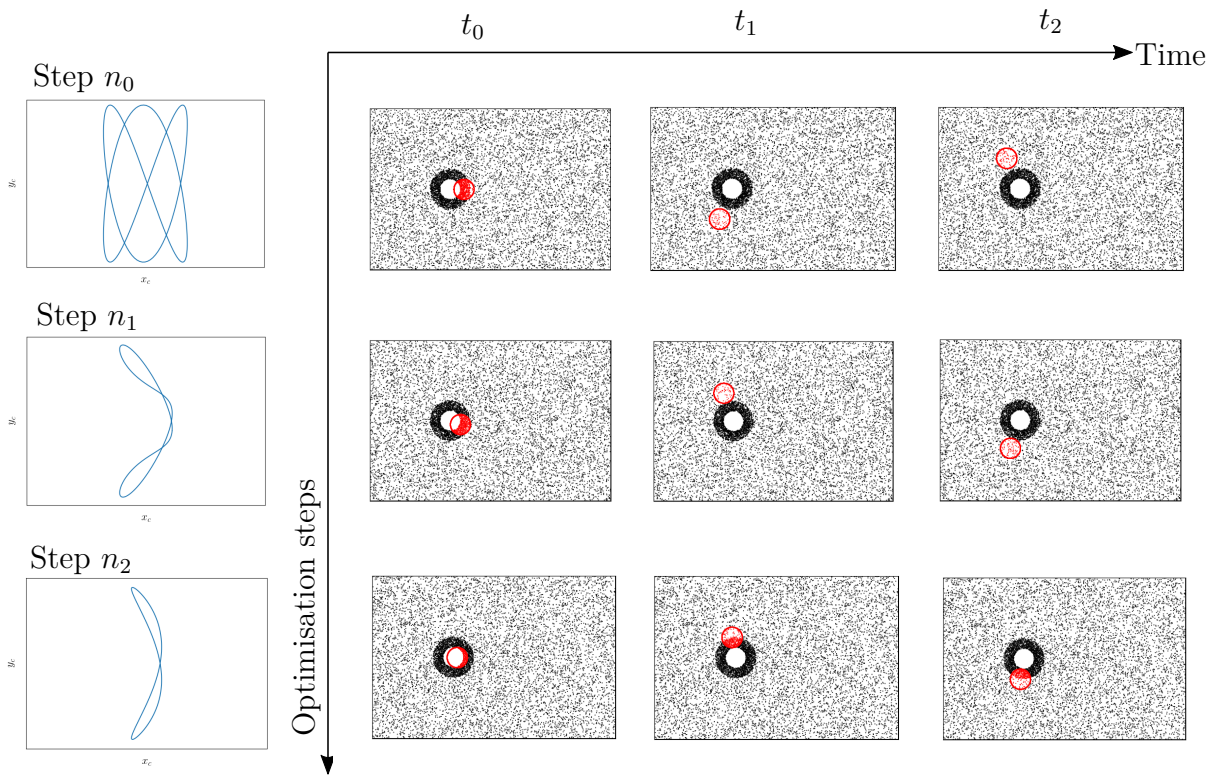


Figure 5.3 Illustration of the problem of static penalization points while the fluid domain moves with time and during the optimization process. Some points are outside Ω_f^t (coloured in red) whereas some empty spaces are created resulting in a flow area that is not penalized. On the left column is illustrated the convergence of PINNs that encode x_c and y_c as functions of time with cylinders coordinates trajectory.

Since every transformation is carried out inside the symbolic graph of operations, derivatives are still available with automatic differentiation. Especially, computation of the instantaneous normal vectors $\mathbf{n}(s,t)$ is straightforward by switching x^0_{BC} by x^t_{BC}:

$$\begin{aligned} n_x &= -\frac{\partial y^t_{BC}}{\partial s}, \\ n_y &= \frac{\partial x^t_{BC}}{\partial s}, \end{aligned} \tag{5.6}$$

as illustrated in Figure 5.4.

In some cases, the displacement of the boundary can be encoded elsewhere. For instance for a moving cylinder, a PINN can encode the solid displacement of the cylinder's center $x_c(t), y_c(t)$. In that case, the boundary is obtained simply by the translation and there is no need to use the extension operator. Nonetheless, it is necessary to make sure that the displacement of the fluid domain obtained via ξ alongside the border and the one of the border via $x_c(t)$ are equal. To achieve that, there are two possibilities:

- Adding a penalization term in the loss that would look like

$$\mathcal{L}_{fr} = \frac{1}{N_s}\sum_{s,t}\left[\xi_x\left(x^0_{BC}(s), y^0_{BC}(s), t\right) - x_c(t)\right]^2 + \left[\xi_y\left(x^0_{BC}(s), y^0_{BC}(s), t\right) - y_c(t)\right]^2. \tag{5.7}$$

- Using a prior dictionary for the extension operator. The current $\boldsymbol{\xi}$ would be modified into $\hat{\boldsymbol{\xi}}$ as follow:

$$\hat{\boldsymbol{\xi}}(x^0, y^0, t) = \underbrace{\boldsymbol{\xi}(x^0, y^0, t)}_{\text{Output of NN}} \times \underbrace{f^0_{bc}(x^0, y^0)}_{=0 \text{ on } \partial\Omega^0_{FS}} + \mathbf{x}_c(t) \tag{5.8}$$

In the case where a PINN is used to define $\boldsymbol{\xi}$, both of the methods are possible and this choice may affect the convergence efficiency. Moreover, like in an ALE method, a regularization is required so that the deformations of $\boldsymbol{\xi}$ in the fluid domain are smooth. Also this prevents $\boldsymbol{\xi}$ from concentrating penalization points in an area with low errors and lowering the concentration of points in the area of importance. To enforce that penalization, a regularization equation can be used, like a solid-elastic (inspired by ALE) or a Poisson's equation which is known to smooth displacements and is also used for mesh generation [121, 122].

On the other hand, with a prior-dictionary it is possible to construct an explicit extension

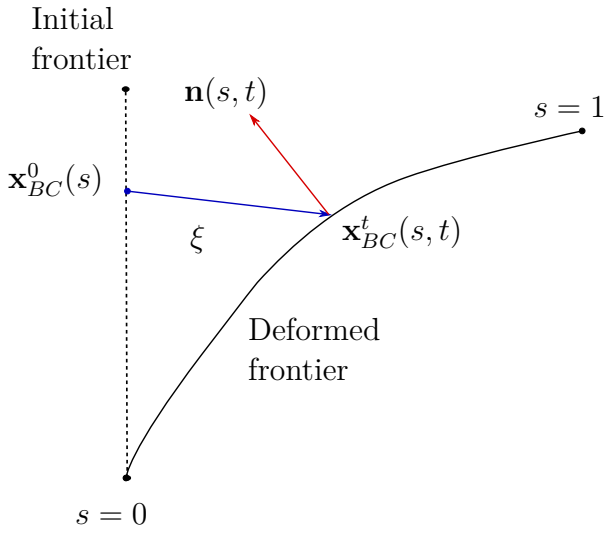


Figure 5.4 Illustration of the deformation of the boundary using the extension operator. The initial frontier (dashed line) is transformed into the deformed frontier (solid line) using the extension operator $\boldsymbol{\xi}$ (blue arrow). Combination of these symbolic operators allows the computation of the normal vector $\mathbf{n}$ (red arrow).

operator that would not require any specific training. For the case of a cylinder in a box, here is an example of an effective explicit extension operator:

$$\begin{aligned}
\xi_x\left(x^0,y^0,t\right) &= \overbrace{\left(1-f^0_{BC}(x^0,y^0)\right)}^{=1 \text{ on } \partial\Omega^0_{FS}} \times x_c(t) \times \overbrace{f_{BDR}(x^0,y^0)}^{=0 \text{ on } \partial\Omega_{\text{ext}}},\\
\xi_y\left(x^0,y^0,t\right) &= \left(1-f^0_{BC}(x^0,y^0)\right) \times y_c(t) \times f_{BDR}(x^0,y^0),\\
f^0_{BC}(x^0,y^0) &= \tanh\left(\gamma\left[\sqrt{(x^0-x^0_c)^2+(y^0-y^0_c)^2}-r_c\right]\right),\\
f_{BDR}(x^0,y^0) &= \tanh\gamma(x^0-L_{x,max}) \times \tanh\gamma(L_{x,min}-x^0)\\
&\quad \times \tanh\gamma(y^0-L_{y,max}) \times \tanh\gamma(L_{y,min}-y^0),
\end{aligned} \tag{5.9}$$

where on the initial fluid-solid border of the cylinder $\partial\Omega^0_{FS}$, $f^0_{BC}=0$ and $f_{BDR}=1$ so that we ensure that $\xi_x = x_c(t)$ and $\xi_y = y_c(t)$ at each time and at each iteration of the training. Moreover, $f_{BDR}=0$ on the exterior border $\partial\Omega_{\text{ext}}$ of the fluid domain so that each penalization point stays in the initial box composed of fluid and solid domains. An example of this explicit extension operator is displayed at Figure 5.5. In this example a slight change was made by choosing

$$f^0_{BC}(x^0,y^0) = \text{ReLu}\left\{\tanh\left(\gamma\left[\sqrt{(x^0-x^0_c)^2+(y^0-y^0_c)^2}-1.2r_c\right]\right)\right\}. \tag{5.10}$$

This slight modification increases the size of the area around the cylinders where $\xi_x \approx x_c$ before this is flattened by the initial f^0_{BC}. The rectified linear unit $\text{ReLu}(x)=\max(x,0)$ is there only to make sure that $f^0_{BC}=0$ in the area where the distance to the initial cylinder's centre is between r_c and $1.2r_c$. In the end, this type of approach can work for simple cases where these adjustments can be shaped by hand. But they may lack of generalization. Instead the general approach of using a PINN for the explicit operator may be chosen since it can adapt to any geometry in two and three dimensions.

The fact that penalization points are moving with time must be taken into account for the governing PDE. Indeed this can create a non-physical change of reference frame and therefore affect the derivatives. Especially, when we compute $\frac{\partial}{\partial t}$ in the graph of operation of one of the variable, for say $u(x^t,y^t,t)$, it takes into account that x^t and y^t are functions of x^0, y^0

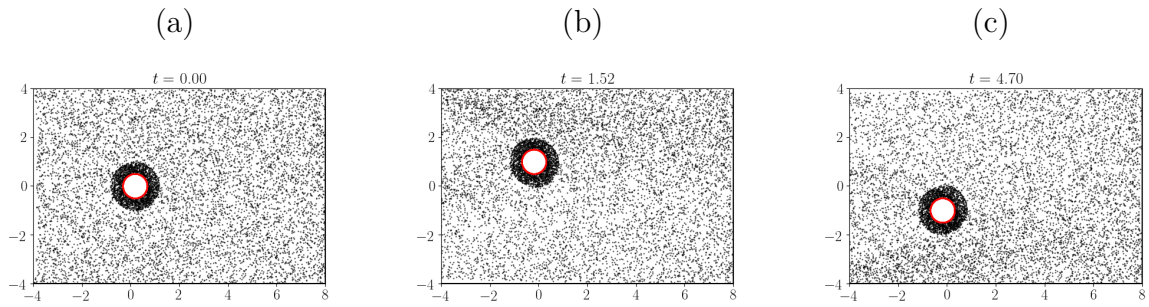


Figure 5.5 Displacement of penalization points using an explicit extension operator with $x_c(t)$ and $y_c(t)$ taken from simulation data. The three snapshots are extracted at the central position (a) and at the extreme transverse displacements (b and c).

and t as defined in equation 5.4. More specifically, we have

$$\underbrace{\frac{\partial u}{\partial t}(x^t, y^t, t)}_{\text{As computed with } \textbf{tf.gradients}} = \left[\frac{\partial u}{\partial t}(x^t, y^t, t)\right]_{x^t, y^t} + \left[\frac{\partial u}{\partial x^t}(x^t, y^t, t)\right]_{y^t, t} \times \left[\frac{\partial x^t}{\partial t}(x^0, y^0, t)\right]_{x^0, y^0} + \left[\frac{\partial u}{\partial y^t}(x^t, y^t, t)\right]_{x^t, t} \times \left[\frac{\partial y^t}{\partial t}(x^0, y^0, t)\right]_{x^0, y^0}, \tag{5.11}$$

and to obtain the $\frac{\partial u}{\partial t}$ in the NS equations (that we should write $\left[\frac{\partial u}{\partial t}(x^t, y^t, t)\right]_{x^t, y^t}$ to emphasize that x^t and y^t are kept constant while derivating with respect to t), we can use the first term in the right hand side of equation 5.11. All the other terms can be obtained with `tf.gradients` function from automatic differentiation in Tensorflow.

It is possible to bypass this correction of derivatives with a trick specific to Tensorflow. When computing `tf.gradients(u,t)`, Tensorflow looks into the graph of symbolic operations to link $u\left(x^t(x^0, y^0, t), y^t(x^0, y^0, t), t\right)$ to every occurrence of t which are to be found in x^t and y^t in addition to the third input of u. But if we compute u as $u\left(x^t(x^0, y^0, t'), y^t(x^0, y^0, t'), t\right)$ where t' is a variable independent from t in the graph of operations, then

$$\left[\frac{\partial u}{\partial t}\left(x^t(x^0, y^0, t'), y^t(x^0, y^0, t'), t\right)\right]_{t'=t} = \left[\frac{\partial u}{\partial t}(x^t, y^t, t)\right]_{x^t, y^t \text{ fixed}}, \tag{5.12}$$

under the condition that we assign the same values for t' and t.

Another possibility that has not been explored consist of sampling penalization points uniformly in the domain and use only those in the current fluid domain. To do so, there need to

be an explicit condition that is able to discriminate whether points are in or outside Ω_f^t at time t and for the current optimization step. One problem that may be encountered is that it could create a bump during the optimization and therefore trouble or make the gradient descent unstable. Indeed, when a group of points change of status, the loss is directly affected and even if it is dynamically averaged by the activated points, the loss is not uniformly sampled in space.

5.2.2 Representation in a fixed frame with displacement of measurement points

Encoding the small fluid patterns of the boundary layer and the vortices in the wake is already a challenge. Moreover, it is more complicated for the PINN to encode these pattern if they are moving spatially. So one idea is to constrain the representation of the flow in a fixed frame, for instance the initial configuration where the fluid-solid frontier is defined by $\partial\Omega_{FS}^0$. Consequently, the PINNs that approximate flow variables are functions of the fixed frame coordinates x^0, y^0 as well as time t.

Although the set of penalization points does not have to be moved since flow variables are defined in the initial domain, a similar problem appears for the fitting term of the loss. From a practical point of view, measurements data are known in the actual frame and are formatted as a list of $x_{mes}^t, y_{mes}^t, t_{mes}, u_{mes}, v_{mes}, p_{mes}$. But to compare it to the output of PINNs, the corresponding location in the initial frame is required. Thus, we define an inverse extension operator $\boldsymbol{\eta} = (\eta_x, \eta_y)$ that maps the actual locations with the fixed frame where PINNs are defined :

$$\begin{aligned} x^0(x^t, y^t, t) &= x^t - \eta_x\left(x^t, y^t, t\right), \\ y^0(x^t, y^t, t) &= y^t - \eta_y\left(x^t, y^t, t\right), \end{aligned} \tag{5.13}$$

as a mirror of the direct extension operator $\boldsymbol{\xi}$.

Therefore, a similar problem appears for the derivatives of the flow quantities with this change of frame. To express the derivatives of the actual coordinates in terms of x^0, y^0 and t of a

flow variable $q\left(x^0, y^0, t\right)$, we develop the chain-rule:

$$\begin{aligned}\left[\frac{\partial q}{\partial x^t}(x^0, y^0, t)\right]_{y^t,t} = &\left[\frac{\partial q}{\partial x^0}(x^0, y^0, t)\right]_{y^0,t} \times \left[\frac{\partial x^0}{\partial x^t}(x^t, y^t, t)\right]_{y^t,t} \\ &+ \left[\frac{\partial q}{\partial y^0}(x^0, y^0, t)\right]_{x^0,t} \times \left[\frac{\partial y^0}{\partial x^t}(x^t, y^t, t)\right]_{y^t,t}.\end{aligned} \tag{5.14}$$

The result is similar for $\frac{\partial}{\partial y^t}$. For the derivation with respect to time, there is an additional term:

$$\begin{aligned}\left[\frac{\partial q}{\partial t}(x^0, y^0, t)\right]_{x^t,y^t} = &\left[\frac{\partial q}{\partial t}(x^0, y^0, t)\right]_{x^0,y^0} + \left[\frac{\partial q}{\partial x^0}(x^0, y^0, t)\right]_{y^0,t} \times \left[\frac{\partial x^0}{\partial t}(x^t, y^t, t)\right]_{x^t,y^t} \\ &+ \left[\frac{\partial q}{\partial y^0}(x^0, y^0, t)\right]_{x^0,t} \times \left[\frac{\partial y^0}{\partial t}(x^t, y^t, t)\right]_{x^0,y^0}.\end{aligned} \tag{5.15}$$

The problem is that, to compute the gradient of coordinates in the fixed frame with respect to the actual coordinates using the inverse extension operator

$$\frac{\partial x^0}{\partial x^t} = \frac{\partial}{\partial x^t}\left[x^t - \eta_x(x^t, y^t, t)\right] = 1 - \frac{\partial \eta_x}{\partial x^t}(x^t, y^t, t), \tag{5.16}$$

we can notice that x^t, y^t must also be provided. This means practically that both $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ needs to be used. In the case $\boldsymbol{\xi}$ is the inverse operator of $\boldsymbol{\eta}$, the previous quantity can be computed using both operators:

$$\frac{\partial x^0}{\partial x^t} = 1 - \frac{\partial \eta_x}{\partial x^t}\left(x^0 + \xi_x(x^0, y^0, t), y^0 + \xi_y(x^0, y^0, t), t\right). \tag{5.17}$$

And to make sure this happens, the penalization of the inversion of operators is taken into account with an additional term in the loss function:

$$\begin{aligned}\mathcal{L}_{inverse} = \frac{1}{N_{in}} \sum_{x^0,y^0,t \in V_{in}^0} &\left[x^0 - \overbrace{\underbrace{\left(x^0 + \xi_x(x^0, y^0, t)\right)}_{x^t} + \eta_x(\underbrace{x^0 + \xi_x(x^0, y^0, y)}_{x^t}, \underbrace{y^0 + \xi_y(x^0, y^0, t)}_{y^t}, t)}^{\sim x \text{ component of } \eta \circ \xi}\right]^2 \\ &+ \left[y^0 - \overbrace{\underbrace{\left(y^0 + \xi_y(x^0, y^0, t)\right)}_{y^t} + \eta_y(\underbrace{x^0 + \xi_x(x^0, y^0, y)}_{x^t}, \underbrace{y^0 + \xi_y(x^0, y^0, t)}_{y^t}, t)}^{\sim y \text{ component of } \eta \circ \xi}\right]^2\end{aligned} \tag{5.18}$$

This penalizes the mean square error between the initial coordinates and the one that is found when applying $\boldsymbol{\xi}$ then $\boldsymbol{\eta}$ for all the points in the domain. It is possible to do the symmetric penalization of $\xi \circ \eta$ instead of $\eta \circ \xi$ with a similar formalism.

However, this exact inversion may not be required and it could be possible to only use a direct operator $\boldsymbol{\xi}$ and an inverse operator $\boldsymbol{\eta}$ that are not the exact inverse but that still perform a smooth mapping between the fixed and the actual frame. In the end we would have $\boldsymbol{\eta}$ for the fitting part of the loss and for plotting the solution. For training points we would generate points in Ω_f^0 that would be transformed in Ω_f^t using $\boldsymbol{\xi}$ and then sent back to Ω_f^0 using $\boldsymbol{\eta}$. In that case we just need to assure that this double transformation allows a smooth distribution of points in Ω_f^0 so that penalization of equations are carried out correctly.

5.3 On the use of Prior-Dictionnary and Modal Analysis

5.3.1 Prior Dictionaries for Vortex Induced Vibrations in PINN

Using Prior-Dictionary is an asset since it make it possible to enforce kinematic boundary conditions directly without a penalization term that may not converge before the other terms of the loss. Some prior-dictionary have been presented for the extension operators in the case of the VIV test-case. Besides, it is still possible to use prior-dictionary for velocity field with slight adjustments:

- Instead of $u = NN(\cdot;\theta) \times f_{bc}$ where the $f_{bc} = 0$ on the frontier, we should add the velocity of the frontier defined by $\frac{\partial x_c}{\partial t}(t) : u = NN(\cdot;\theta) \times f_{bc} + \frac{\partial x_c}{\partial t}$ so that u equals the velocity of the solid instead of 0.
- Depending on the approach chosen for flow mapping (in the actual frame or in a fixed frame), the area where $f_{bc} = 0$ changes. Moreover f_{bc} is either a function of x^0, y^0 when u is defined on Ω_f^0, or a function of x^t, y^t, t if u is defined on Ω_f^t.

However, this formalism may lack of generalization since most of the explicit shapes are designed by hand and this become more complex when the geometry becomes less trivial.

5.3.2 Modal Analysis

Modal decomposition in a moving domain may uncover interesting phenomena. First, we did not find in the scientific literature any appropriate definition of mode shapes in the cases of large deformations of the fluid domain. This can be problematic for the first approach where the fluid fields are computed in the actual reference frame because there are area that are not in Ω_f^t at every moment of the oscillation period and are thus, undefined. We could neglect this and plot mode shapes with the same formalism as in chapter 4 on all the domain (fluid an solid) keeping in mind that the obtained solution must be cropped depending on the phase position.

For the second approach where the flow fields are defined in a fixed frame of coordinates, it makes more sense to consider mode shapes since there is a flow quantity defined at every point and every time instant. Besides, there would also be mode shapes for the extension operators. However, there is a difficulty that is still not clarified: the combination of relative speed between that of the domain and those of flow quantities might lead into a Doppler effect which could affect the distribution of frequencies. At this stage, it is not clear if the Fourier decomposition consisting in an average in time mode $\hat{u}_0$ and a sequence of mode shapes $\hat{u}_k$ at harmonic frequencies $k \times f_0$ would still hold. It is possible that we would need to consider several fundamental frequencies alongside some of their combined harmonics, which has not been explored yet.

5.4 Results, limitations and outlooks

Here are presented the results of a typical job where we tried to reconstruct the flow fields u, v, p and cylinder's displacements x_c, y_c using classical PINN (without the modal approach). The presented results are obtained after a run with properties recalled in Table 5.1 and with quantitative errors at the end of the training listed in Table 5.2. In this run the approach used is the fixed frame of coordinates to define flow fields. Direct and inverse extension operators are used as described earlier. They are defined by an analytical shape close to equation 5.9 (with the modification of $1.2r_c$ discussed in the paragraph next to equation 5.9) and for $\boldsymbol{\eta}$:

$$
\begin{aligned}
\eta_x\left(x^t,y^t,t\right) =& -\overbrace{\left(1-f^t_{BC}(x^t,y^t,t)\right)}^{=1 \text{ on } \partial\Omega^t_{FS}} \times x_c(t) \times \overbrace{f_{BDR}(x^t,y^t)}^{=0 \text{ on } \partial\Omega_{\text{ext}}},\\
\eta_y\left(x^t,y^t,t\right) =& -\left(1-f^t_{BC}(x^t,y^t,t)\right) \times y_c(t) \times f_{BDR}(x^t,y^t),\\
f^t_{BC}(x^t,y^t,t) =& \text{ReLu}\left\{\tanh\left(\gamma\left[\sqrt{(x^t-x_c(t))^2+(y^t-y_c(t))^2}-1.2r_c\right]\right)\right\},\\
f_{BDR}(x^t,y^t) =& \tanh\gamma(x^t-L_{x,max}) \times \tanh\gamma(L_{x,min}-x^t)\\
&\times \tanh\gamma(y^t-L_{y,max}) \times \tanh\gamma(L_{y,min}-y^t).
\end{aligned}
\tag{5.19}
$$

Dense data for the three flow fields as well as for solid displacements are provided in loss terms:

$$
\mathcal{L}_{mes,f} = \frac{1}{N_m}\sum\left[(u_{mes}-u)^2+(v_{mes}-v)^2+(p_{mes}-p)^2\right]_{\text{Computed at } x^0_{mes},y^0_{mes},t}, \tag{5.20}
$$

$$
\mathcal{L}_{mes,s} = \frac{1}{N_{m,s}}\sum_{t_{mes}}\left(x_{c,mes}-x_c(t_{mes})\right)^2+\left(y_{c,mes}-y_c(t_{mes})\right)^2. \tag{5.21}
$$

Besides, penalization of the NS equations takes into account the change of reference in the term $\mathcal{L}_{eqs,f}$. Dynamic boundary conditions as recalled in equations 5.1 and 5.2 are penalized for a sampling of time instants and averaged in the loss term $\mathcal{L}_{eqs,s}$. Finally a last term is used in the loss function and quantifies the mean square error on kinematic conditions:

$$
\mathcal{L}_{BC} = \frac{1}{N_{BC}}\sum_{s,t}\left[u(x^t_{BC}(s,t),y^t_{BC}(s,t),t)-\frac{\partial x_c}{\partial t}(t)\right]^2+\left[v(x^t_{BC}(s,t),y^t_{BC}(s,t),t)-\frac{\partial y_c}{\partial t}(t)\right]^2. \tag{5.22}
$$

Then all the loss terms are concatenated for the training:

$$
\mathcal{L} = \mathcal{L}_{eqs,f}+\mathcal{L}_{eqs,s}+\mathcal{L}_{mes,f}+\mathcal{L}_{mes,s}+\mathcal{L}_{BC} \tag{5.23}
$$

The value of each part of the loss after the training is summarized in Table 5.2. These errors are 1-2 order of magnitude higher than those obtained in the previous chapter for the vortex shedding around a fixed cylinder. Qualitative results are depicted in Figure 5.6 for snapshots of flow fields and Figure 5.7 for forces and displacement of the solid border. It can be noted from the snapshots that the area around the cylinder has been displaced slightly downstream and in the positive y. Besides the general shape of the wake is quite different than the one from exact data. Regarding the displacements, the longitudinal components x_c

display an average value coherent with data but misses the variable part (which are of small amplitudes, about 10%, compared to the average), and y_c is accurately recovered for both the average (which is zero) and the oscillating part, but this is not that difficult since a lot of measurements of x_c and y_c have been fed to the training. However important differences both in average value and oscillations of the forces can be noted. These results show indeed that the fields of velocities and pressure have not converged properly.

Several other runs were carried out using the possible combinations of solutions proposed. But at this day, no jobs have clearly converged in an acceptable way. Moreover it has to be noted that these training have been performed using dense and numerous data of the fluid fields as well as data of solid displacement. This training configuration is much easier than more realistic ones that are targeted for these tools, as discussed in the end of chapter 4.

Other training configurations consists in:

- Computing the flow fields in the actual frame or in a fixed reference frame
- Using Explicit direct and inverse extension operators. Or use a PINN with or without prior-dictionaries to shape $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$.
- Pre-train $x_c(t)$ and $y_c(t)$ with measurements of displacement before working on flow quantities so that explicit extension operators can provide a good mapping to measurement data.
- Using the classical derivation technique with the trick of mirroring the time variable in the graph of operations in Tensorflow (see equation 5.12).
- Training without the penalization of equations.
- Using (or not) a ModalPINN with a sequence of frequencies (harmonic or any type of list so that multiple fundamental frequencies may be took into consideration)

Eventually, among the tested combination of configurations and optimization parameters, we have not yet obtained any result that shows that, even when feeding the complete solution to the PINNs, the VIV can be reconstructed using these proposed approaches. Nonetheless, the step forward compared to the static case does not seem to be that big, and it is also possible that the absence of convincing results may be due to mischosen neural networks, optimization process or errors in the code.

Table 5.1 Parameters set in a run for the reconstruction of VIV, which results are summarized in Table 5.2.

Parameter	Value
FSI Approach	Fixed reference frame using $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$
Data	Dense (u, v, p, x_c, y_c)
Time	7h
N_{in}	4×10^4
N_m	1.5×10^5
N_{BC}	1×10^5
modalPINN	No
Prior-Dictionary	No
Extension operator	Explicit for $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$
L-BFGS-B	108 iterations
Adam	53 200 iterations

Table 5.2 Summary of the errors at the end of the run described in Table 5.1 for the flow reconstruction of VIV.

Symbol	Error	Value
$\mathcal{L}$	Training (end)	1.7×10^{-1}
$\mathcal{L}_{eqs,f}$	Fluid equations (end of train.)	3.6×10^{-2}
$\mathcal{L}_{eqs,s}$	Solid equations/Dynamic BC (end of train.)	3.3×10^{-2}
$\mathcal{L}_{mes,f}$	Velocity and pressure mes. (end of train.)	8×10^{-2}
$\mathcal{L}_{mes,f}$	Velocity and pressure mes. (validation)	8×10^{-2}
$\mathcal{L}_{mes,s}$	Solid displacement mes. (end of train.)	6.7×10^{-3}
$\mathcal{L}_{BC}$	Kinematic BC u, v (end of train.)	1.9×10^{-2}

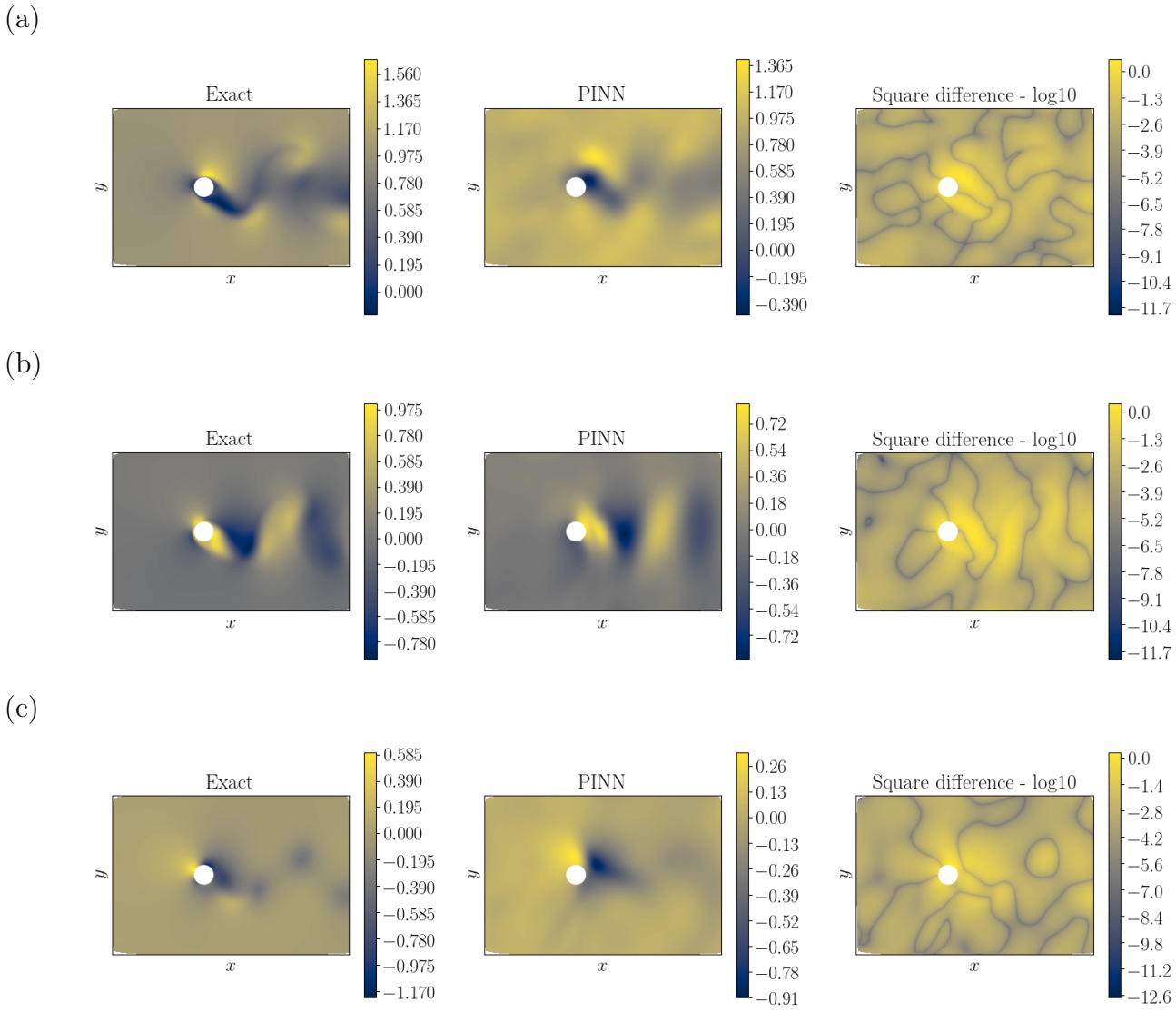


Figure 5.6 Snapshot of reconstructed flow fields u, v and p (resp. a, b and c) with exact data from numerical simulations plotted in the fixed frame. For each quantity, square difference is plotted in log scale.

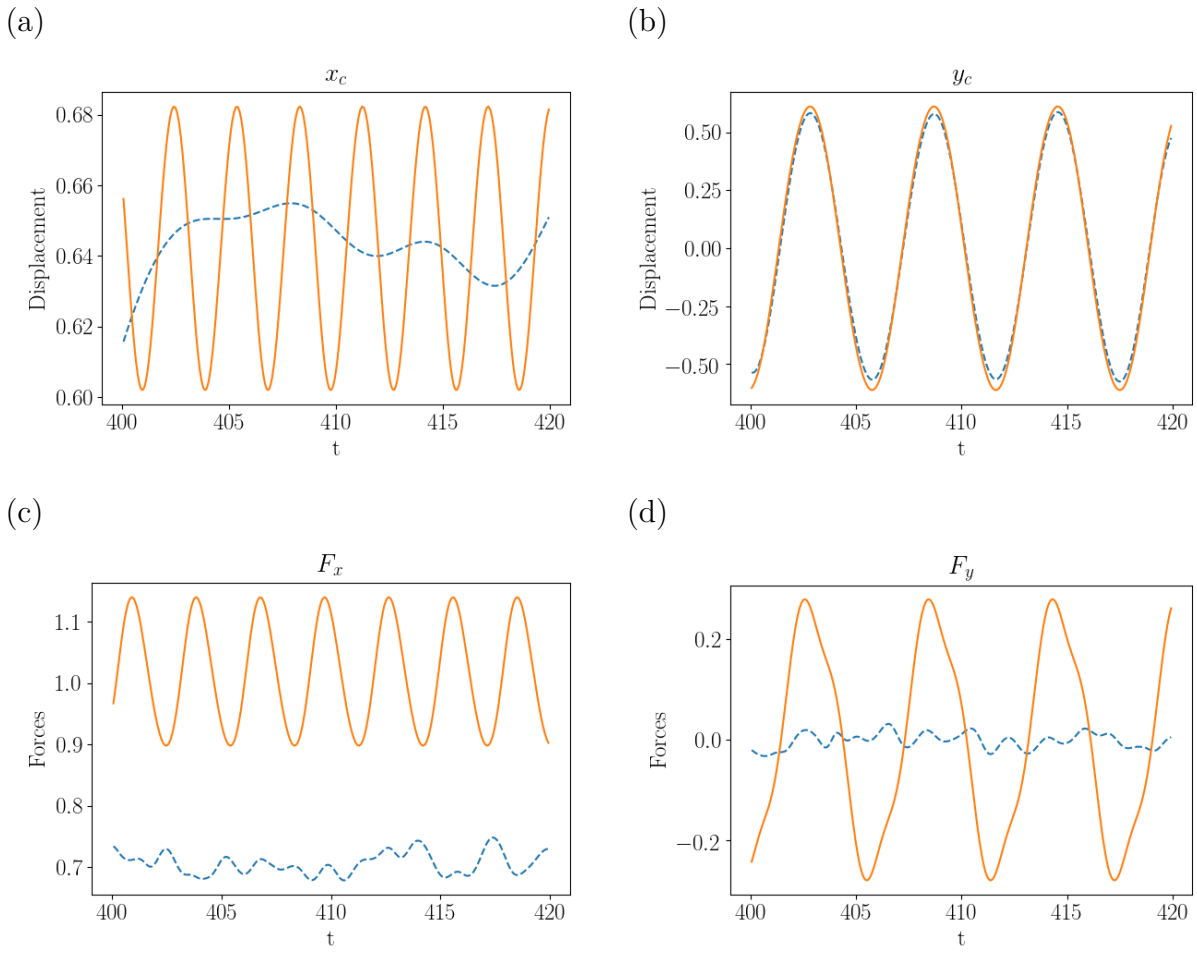


Figure 5.7 Results of displacements x_c, y_c and forces F_x, F_y in subplot (a), (b), (c) and (d) respectively for the VIV test-case. Exact data from numerical simulations in orange solid line are compared to PINN reconstruction in blue dashed lines.

In the case an appropriate configuration manages to converge to the fed solution, it could be interesting to try to recover the same solution but using partial information. Especially here are some suggested scenarios:

- Using only fluid data, without measurements of the cylinder's position.
- Using only displacement data and boundary conditions for the fluid.
- Using only fluid data but in an area constrained to the wake of the cylinder.

Being able to perform such reconstruction with accurate estimation of forces and displacements could be of interest for experimental research and industrial applications.

CHAPTER 6 GENERAL DISCUSSION

One of the reason that may explain the time between the description of the concept of PINNs by Dissanayake et al. in 1994 [63] and the interest and spreading of this technique after the work of Raissi et al. in 2018-2019 [66], apart from the differences of computing resources (hardware and software), lie in the the small data regimes. As illustrated in Figure 6.1, without nearly any knowledge of the physical laws, experimental fluid mechanics can provide data that are directly used to study, design, optimize and certify various engineering solutions. The drawbacks are that scaled models must be used to fit in laboratory facilities for large systems and that information can not be measured everywhere. On the contrary, numerical simulations, and especially DNS where no hypothesis is used to model fluid flows, require no data on the solution (which is useful for completely new test cases) and provide large amount of information for the engineering process without constrains on the size of the system. However the drawback is that the precision is limited by the computing resources (especially for turbulent flows) and by the uncertainty on the physical parameters and boundary conditions that are inputted in the computation. In the end, PINNs stand as a bridge between these two sides. As observed especially in chapters 3, 4 and in appendix A.2, and showed in the literature [119], PINN let extend the experimental data range outside of its area of measurement. As with the example of the noised PIV with holes in appendix A.2.2, it opens a path to correct experimental data when they are corrupted by noise for instance. But PINNs also allow improving the performance of classical numerical simulations by allowing a more precise identification of the physical parameters as discussed in chapter 3 or by easing the creation of models as discussed for turbulent flows [123].

This in-between objective find practical tools in machine learning and especially PINNs, largely because the computing resources are designed to perform fast optimization with large data sets. Nonetheless technical limitations of PINNs as the convergence duration, the lack of robustness and the difficulty of PINNs to be generalized to several configuration (often referred as transfer learning) encourage to follow other techniques that target similar objectives. Among these, adjoint optimization [124] or automatic differentiation implemented in more classical codes like finite elements [125, 126], and identification of non linear dynamics (SINDY) applied to low dimensional representations are example of promising techniques [53]. These could also serve as inspiration for upgrading the approach of PINNs. As an example, hp-VPINN defined by Kharazmi et al. [127] has many similarities in its formulation with FEM. Furthermore the ModalPINN presented in chapter 4 is comparable to the use of

SINDY [52] applied to Fourier mode shapes. Especially ModalPINN offers leads to increase the generalizability of PINN because the mode shapes may be combined in a different way with different time functions, amplitude or phase shift. This also proved to be more efficient that the classical PINN approaches, which then may help the convergence of PINNs in applications closer to realistic configurations, which is discussed in appendix A.2. Finally, the understanding that PINN is primarily a way of **discretisation** (or approximation of the continuous fields with symbolic functions) leads to its combination with various way of **representation** (or model reduction). As outlined in appendix A.1 and described in Figure A.1, the continuity of ModalPINN is to use PINN approximation of simpler functions, then combined in modal representation that can be enriched the one after the other. This possibility also finds similarity with the Proper Generalized Decomposition (PGD) [128] in terms of representation. Finally, a gap in the literature lie in the use of modal representation for fluid-structure vibrations with the problem of moving domain as discussed in chapter 5. As vibrations are often associated with frequencies and mode shapes, there is some motivation to overcome the question of the fluid domain definition to be able to use the previously developed tools of modal representation to address FSI.

Independently of the system to which PINN is applied, this thesis has focused on challenging PINNs in various directions as schematized in Figure 6.2. The horizontal axis stands for the quantity of data fed to the training and highlights a similar in-between as in Figure 6.1. When data is abundant for all the continuous fields to be reconstructed, without holes in space or time and without imperfections, the reconstruction is relatively easy to perform (see the discussion in chapter 3). On the other hand, there is little interest to perform so because a good reconstruction could be obtained with a direct fitting using low order polynomial at a relatively low error (that could be around a few percents, 1-5% to give a quantitative value). With the same quantity of data, a slightly harder problem appears when some fields are missing in the training data because they can only be obtained using the PDE couplings between all the fields. This is the case for pressure fields in incompressible flows, that are not experimentally measured with PIV for instance. When data is still dense but when the amount (and in fine, the nearly uniform concentration) of it decreases, the reconstruction of the field using only polynomial fitting becomes less precise and can leads to errors of the order of magnitude of the solution itself. The physical regularization of PINNs brings there a practical solution to correct these interpolations with the PDE. Then when holes in the data appears (as illustrated with the array of sensors in subsection 4.5.3 or with the 2 PIV areas around the array of cylinders in appendix A.2.2), the direct interpolation leads to errors greater than the solution itself and physical regularization can not be ignored. Finally, it is

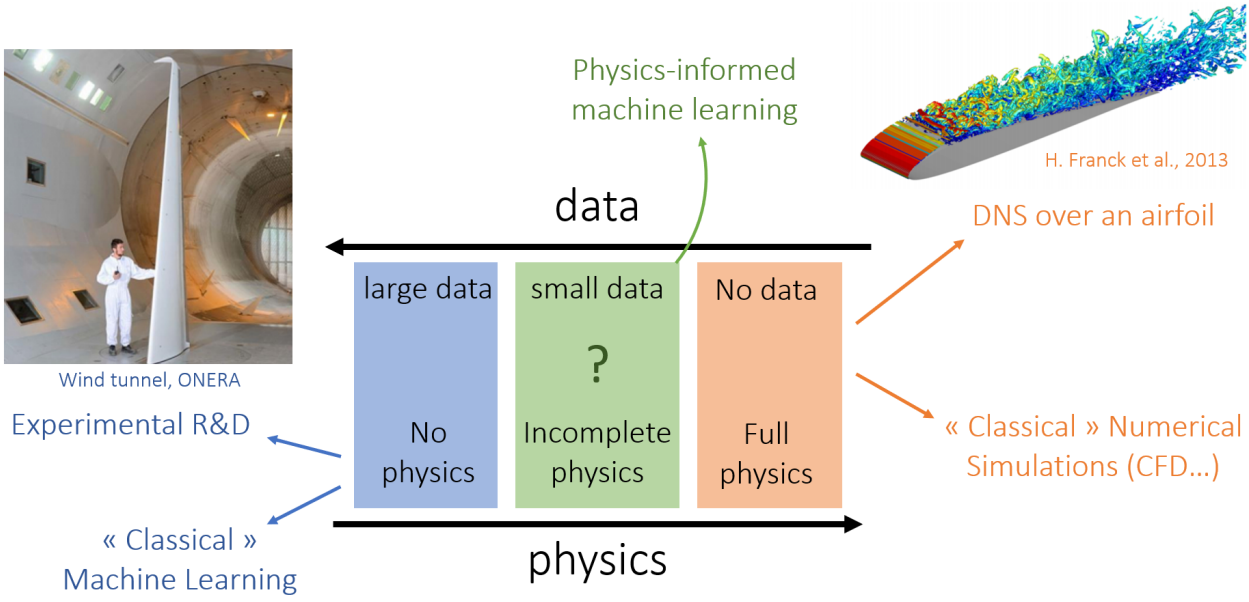


Figure 6.1 Illustration of the area of interest for future developments of PINNs.

possible to consider that a direct problem with Dirichlet boundary conditions (traditionally solved with CFD) is an extreme extension of data sparsity . This is why PINNs are theoretically still suited to solve direct problems. However in these typical cases, more classical methods like FEM show a greater efficiency and robustness than PINNs.

The vertical axis in Figure 6.2 stands for the decrease of data quality. In this thesis, this has been illustrated through the adding of Gaussian noise or delays in time signals (that add uncertainty in a different way). This is an area where uncertainty quantification techniques could be applied to quantify more generally the effect of data imperfections on the precision of predictions. These approaches could also try to understand how incompatibilities in the multi objective optimization performed by PINNs (data and PDE which are incompatible in case of imperfections and overconstrains) are handled.

Finally, the out of plane axis in Figure 6.2 stands for difficulties that are brought by the model itself. This can be the case when multi-physics phenomena like FSI developed in chapter 5 leads to coupling between fields that have difficulties to converge concurrently. Thus one field can not use the second one to help its own convergence (and conversely) through the couplings in PDE. In a simpler version, parameters identification leads to a similar increase in difficulty when, for instance, a scalar value in the PDE is missing, the PDE changes during

optimization and thus the target for the continuous fields while they are converging. In this last case, PINNs seems to perform well as illustrated by the wide literature on parameter identification with PINNs [67–69, 72, 74, 77, 129, 130] which is optimistic for the special case of FSI.

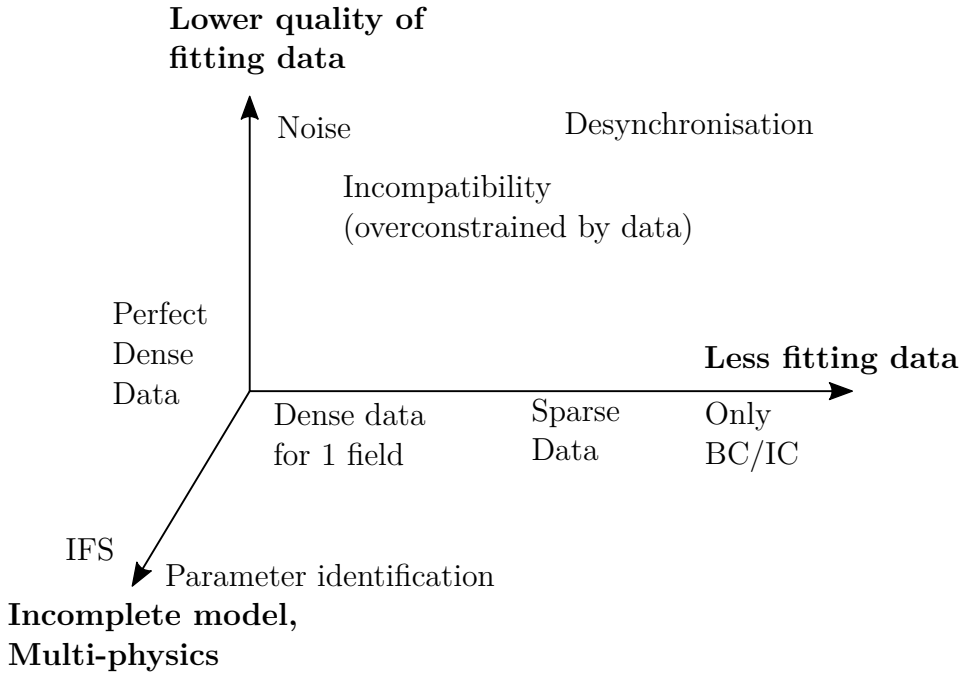


Figure 6.2 Schematics of the three directions explored with PINNs. The quantity of data (horizontal axis), its quality (vertical axis) and the coupling between physics or on the contrary unknown in the physical models (out of plane axis) are illustrated with several test-cases in the previous chapters.

CHAPTER 7 CONCLUSION AND RECOMMENDATIONS

7.1 Summary of Works and Limitations

In this thesis we recalled and developed the formalism of physics-informed neural networks for fluid mechanics applications. After introducing the mathematical foundations of PINNs and the variety of techniques to perform the training, we provided practical methods to generate loss functions containing both fitting data and partial knowledge of physical laws. This formalism was implemented in Python using Tensorflow for automatic differentiation and training. Several runs were performed on a wide variety of hardware set-up (laptops, professional desktops on CPU and GPU, world-class computation cluster with Graham server on Compute Canada). The influence of PINN's parameters and several training strategies were discussed over basic test-cases and practical guidelines were provided in order to chose appropriate NN and convergence criteria.

Flow reconstruction using modal decomposition proved to be of interest for periodic flows to increase the efficiency and the precision at a given neural network size. Moreover, the test performed regarding the handling of unknowns in the data like spatial sparsity, noise and out of synchronization, tend to prove that PINNs are able to correct some imperfections and enhance the quantity of the provided data. We also discussed other possible modal decomposition as well as adaptations to more complex flow configurations so that this work may be expanded toward more physically-rich and applied applications.

This thesis also investigates techniques to address fluid-structure interactions within the framework of PINNs. Several hypothesis have been made to developed two approaches using the moving domain and a fixed artificial frame of reference. Extension operators that encode the deformations of the domains modify the partial derivatives used in the physical equations. So far, none of these solutions proved to converge properly using a test case consisting of vortex induced vibrations.

7.2 Future Research

In the short-term there are interesting challenges to address using PINNs with the optimization of new auxiliary variables. For instance it could be interesting to try to recover boundary

conditions from experimental data, since it is known to be a critical point for correlation between numerical and experimental work [131].

In terms of architecture of PINNs, modalPINN or the use of extension operators in FSI brought some changes in the way the solutions are defined and optimized in the end. New approaches might still be explored to find more efficient and robust designs for PINNs.

Moreover, we strongly encourage future developments on PINNs to join one of the open-sources communities like `DeepXDE` (a more exhaustive list is provided by Karniadakis et al. at the end of the paper [81]) for several reasons: even if a basic PINN code can be written in Python and Tensorflow in less than 500 lines (which can be a good practical example for understanding how a PINN works), several stabilizing techniques and improvements for avoiding convergence issues may be implemented in these codes, especially since some hardware manufacturers are involved in these projects (like NVIDIA in `SimNet`) and might better understand the difficulties for distributed training. Besides, default training properties might be better tuned in general, which is often a trouble for beginners in ML. Finally, these codes might implement compatibility with CAD files that will be helpful to deal with more complex geometries and boundary definitions.

Finally, with applications like digital twins in mind, it could be of interest to investigate parametric PINNs where scalar values specific to the configuration could be input in the PINN (like the Reynolds number, Strouhal number, geometric value or any other scale of interest). It is likely to increase a lot the size of the NN required for such a problem, but on the other hand it could make the parametric PINN less dependant of the configuration.

REFERENCES

[1] "Graham — Compute Canada Documentation." [Online]. Available: https://docs.computecanada.ca/wiki/Graham

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, mit press ed., 2016. [Online]. Available: http://www.deeplearningbook.org

[3] M. Boudina, F. P. Gosselin, and S. Étienne, "Vortex-induced vibrations: a soft coral feeding strategy?" *arXiv:2009.11764 [physics]*, Sep. 2020, arXiv: 2009.11764. [Online]. Available: http://arxiv.org/abs/2009.11764

[4] E. d. Langre, *Fluides et solides*. Editions Ecole Polytechnique, 2001, google-Books-ID: IFnLwQ4eTIwC.

[5] C. E. R. Government of Canada, "Canada's Energy Future 2019 - Energy Supply and Demand Projections to 2040," Canada Energy Regulator, Calgary, Alberta, Tech. Rep. NE2-12E-PDF, 2019, iSSN 2292-1710. [Online]. Available: https://www.cer-rec.gc.ca/en/data-analysis/canada-energy-future/2019/2019nrgftr-eng.pdf

[6] A. Presas, Y. Luo, Z. Wang, and B. Guo, "Fatigue life estimation of Francis turbines based on experimental strain measurements: Review of the actual data and future trends," *Renewable and Sustainable Energy Reviews*, vol. 102, pp. 96–110, Mar. 2019.

[7] S. Brunton, B. Noack, and P. Koumoutsakos, "Machine Learning for Fluid Mechanics," *arXiv:1905.11075 [physics, stat]*, May 2019, arXiv: 1905.11075. [Online]. Available: http://arxiv.org/abs/1905.11075

[8] M. P. Brenner, J. D. Eldredge, and J. B. Freund, "Perspective on machine learning for advancing fluid mechanics," *Phys. Rev. Fluids*, vol. 4, no. 10, p. 100501, Oct. 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevFluids.4.100501

[9] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," *Rev. Mod. Phys.*, vol. 91, no. 4, p. 045002, Dec. 2019, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/RevModPhys.91.045002

[10] J. F. Wendt, *Computational Fluid Dynamics: An Introduction*. Springer Science & Business Media, Oct. 2008, google-Books-ID: P_wrvhjiFWIC.

[11] C. L. Fefferman, "EXISTENCE AND SMOOTHNESS OF THE NAVIER–STOKES EQUATION," p. 6.

[12] P. Argoul, "Overview of Inverse Problems," p. 17.

[13] M. Tanaka and G. S. Dulikravich, *Inverse Problems in Engineering Mechanics.* Burlington: Elsevier, 1998, oCLC: 476079017. [Online]. Available: http://www.123library.org/book_details/?id=39224

[14] A. Jameson, "Optimum aerodynamic design using CFD and control theory," in *12th Computational Fluid Dynamics Conference.* San Diego,CA,U.S.A.: American Institute of Aeronautics and Astronautics, Jun. 1995. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.1995-1729

[15] R. M. Errico, "What is an adjoint model?" *Bulletin of the American Meteorological Society*, vol. 78, no. 11, pp. 2577–2592, 1997, publisher: American Meteorological Society.

[16] J. L. Callaham, K. Maeda, and S. L. Brunton, "Robust flow reconstruction from limited measurements via sparse representation," *Phys. Rev. Fluids*, vol. 4, no. 10, p. 103907, Oct. 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevFluids.4.103907

[17] N. B. Erichson, L. Mathelin, Z. Yao, S. L. Brunton, M. W. Mahoney, and J. N. Kutz, "Shallow neural networks for fluid flow reconstruction with limited sensors," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 476, no. 2238, p. 20200097, Jun. 2020, publisher: Royal Society. [Online]. Available: https://royalsocietypublishing.org/doi/full/10.1098/rspa.2020.0097

[18] S. Symon, D. Sipp, P. J. Schmid, and B. J. McKeon, "Mean and Unsteady Flow Reconstruction Using Data-Assimilation and Resolvent Analysis," *AIAA Journal*, vol. 58, no. 2, pp. 575–588, 2020, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.J057889. [Online]. Available: https://doi.org/10.2514/1.J057889

[19] M. Bocquet, "Introduction to the principles and methods of data assimilation in the geosciences," p. 89.

[20] R. Swinbank, "Numerical Weather Prediction," in *Data Assimilation: Making Sense of Observations*, W. Lahoz, B. Khattatov, and R. Menard, Eds.

Berlin, Heidelberg: Springer, 2010, pp. 381–406. [Online]. Available: https://doi.org/10.1007/978-3-540-74703-1_15

[21] M. Bergmann and L. Cordier, "Optimal control of the cylinder wake in the laminar regime by trust-region methods and POD reduced-order models," *Journal of Computational Physics*, vol. 227, no. 16, pp. 7813–7840, Aug. 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999108002659

[22] G. Allaire, *Analyse numérique et optimisation: une introduction à la modélisation mathématique et à la simulation numérique*. Editions Ecole Polytechnique, 2005, google-Books-ID: vReEuE4margC.

[23] S. S. Rao, *The Finite Element Method in Engineering*. Butterworth-Heinemann, Oct. 2017, google-Books-ID: XN0kDwAAQBAJ.

[24] F. Moukalled, L. Mangani, and M. Darwish, *The finite volume method in computational fluid dynamics*. Springer, 2016, vol. 113.

[25] R. W. Clough, "The finite element method in plane stress analysis," in *Proceedings of 2nd ASCE Conference on Electronic Computation, Pittsburgh Pa., Sept. 8 and 9, 1960*, 1960.

[26] A. W. RIZZI and M. INOUYE, "Time-Split Finite-Volume Method for Three-Dimensional Blunt-Body Flow," *AIAA Journal*, vol. 11, no. 11, pp. 1478–1485, 1973, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/3.50614. [Online]. Available: https://doi.org/10.2514/3.50614

[27] C. Mimeau and I. Mortazavi, "A Review of Vortex Methods and Their Applications: From Creation to Recent Advances," *Fluids*, vol. 6, no. 2, p. 68, Feb. 2021, number: 2 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2311-5521/6/2/68

[28] J. Monaghan, "Smoothed Particle Hydrodynamics and Its Diverse Applications," *Annual Review of Fluid Mechanics*, vol. 44, no. 1, pp. 323–346, 2012, _eprint: https://doi.org/10.1146/annurev-fluid-120710-101220. [Online]. Available: https://doi.org/10.1146/annurev-fluid-120710-101220

[29] S. Chen and G. D. Doolen, "Lattice Boltzmann Method for Fluid Flows," *Annual Review of Fluid Mechanics*, vol. 30, no. 1, pp. 329–364, 1998. [Online]. Available: https://doi.org/10.1146/annurev.fluid.30.1.329

[30] X. He and L.-S. Luo, "Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation," *Phys. Rev. E*, vol. 56, no. 6, pp. 6811–6817, Dec. 1997, publisher: American Physical Society. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.56.6811

[31] M. Y. Hussaini and T. A. Zang, "Spectral Methods in Fluid Dynamics," *Annual Review of Fluid Mechanics*, vol. 19, no. 1, pp. 339–367, 1987, _eprint: https://doi.org/10.1146/annurev.fl.19.010187.002011. [Online]. Available: https://doi.org/10.1146/annurev.fl.19.010187.002011

[32] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, Jr, *Spectral Methods in Fluid Dynamics*. Springer Science & Business Media, Dec. 2012, google-Books-ID: c2XmCAAAQBAJ.

[33] J. Hovnanian, "Immersed boundary method for the fluid mecanics applied to fish-like swimming," Theses, Université Sciences et Technologies - Bordeaux I, Dec. 2012. [Online]. Available: https://tel.archives-ouvertes.fr/tel-00835013

[34] G. Hou, J. Wang, and A. Layton, "Numerical Methods for Fluid-Structure Interaction — A Review," *Commun. comput. phys.*, vol. 12, no. 2, pp. 337–377, Aug. 2012. [Online]. Available: https://www.cambridge.org/core/product/identifier/S1815240600003029/type/journal_article

[35] J.-L. Pfister, O. Marquet, and M. Carini, "Linear stability analysis of strongly coupled fluid-structure problems with the Arbitrary-Lagrangian-Eulerian method," *Computer Methods in Applied Mechanics and Engineering*, vol. 355, pp. 663–689, Jun. 2019.

[36] H. Wang and B. Raj, "On the Origin of Deep Learning," *arXiv:1702.07800 [cs, stat]*, Mar. 2017, arXiv: 1702.07800. [Online]. Available: http://arxiv.org/abs/1702.07800

[37] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv:1603.04467 [cs]*, Mar. 2016, arXiv: 1603.04467. [Online]. Available: http://arxiv.org/abs/1603.04467

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, 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," *arXiv:1912.01703 [cs, stat]*, Dec. 2019, arXiv: 1912.01703. [Online]. Available: http://arxiv.org/abs/1912.01703

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[40] D. H. Ballard, G. E. Hinton, and T. J. Sejnowski, "Parallel visual computation," *Nature*, vol. 306, no. 5938, pp. 21–26, Nov. 1983, number: 5938 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/306021a0

[41] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, number: 7540 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/nature14236

[43] A. Beck and M. Kurz, "A Perspective on Machine Learning Methods in Turbulence Modelling," *arXiv:2010.12226 [cs]*, Oct. 2020, arXiv: 2010.12226. [Online]. Available: http://arxiv.org/abs/2010.12226

[44] J. N. Kutz, "Deep learning in fluid dynamics," *Journal of Fluid Mechanics*, vol. 814, pp. 1–4, Mar. 2017. [Online]. Available: https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/deep-learning-in-fluid-dynamics/F2EDDAB89563DE5157FC4B8342AD9C70

[45] K. Taira, S. L. Brunton, S. T. M. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley, "Modal Analysis of Fluid Flows: An Overview," *AIAA Journal*, vol. 55, no. 12, pp. 4013–4041, 2017, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.J056060. [Online]. Available: https://doi.org/10.2514/1.J056060

[46] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, no. 1, pp. 53–58, Jan. 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900142

[47] M. Milano and P. Koumoutsakos, "Neural Network Modeling for Near Wall Turbulent Flow," *Journal of Computational Physics*, vol. 182, no. 1, pp. 1–26, Oct. 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999102971469

[48] P. Dörfler, M. Sick, and A. Coutu, *Flow-Induced Pulsation and Vibration in Hydroelectric Machinery: Engineer's Guidebook for Planning, Design and Troubleshooting*. London: Springer-Verlag, 2013. [Online]. Available: https://www.springer.com/gp/book/9781447142515

[49] R. D. Blevins, *Flow-induced Vibration*. Van Nostrand Reinhold, 1990, google-Books-ID: yJ9RAAAAMAAJ.

[50] D. Stefan and P. Rudolf, "Proper Orthogonal Decomposition of Pressure Fields in a Draft Tube Cone of the Francis (Tokke) Turbine Model," *J. Phys.: Conf. Ser.*, vol. 579, p. 012002, Jan. 2015. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/579/1/012002

[51] P. Rudolf and D. Štefan, "Decomposition of the swirling flow field downstream of Francis turbine runner," *IOP Conf. Ser.: Earth Environ. Sci.*, vol. 15, no. 6, p. 062008, Nov. 2012. [Online]. Available: https://iopscience.iop.org/article/10.1088/1755-1315/15/6/062008

[52] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *PNAS*, vol. 113, no. 15, pp. 3932–3937, Apr. 2016. [Online]. Available: https://www.pnas.org/content/113/15/3932

[53] K. Fukami, T. Murata, and K. Fukagata, "Sparse identification of nonlinear dynamics with low-dimensionalized flow representations," *arXiv:2010.12177 [physics]*, Oct. 2020, arXiv: 2010.12177. [Online]. Available: http://arxiv.org/abs/2010.12177

[54] S. B. Pope, "Turbulent Flows," Aug. 2000. [Online]. Available: /core/books/turbulent-flows/C58EFF59AF9B81AE6CFAC9ED16486B3A

[55] C. D. Argyropoulos and N. C. Markatos, "Recent advances on the numerical modelling of turbulent flows," *Applied Mathematical Modelling*, vol. 39, no. 2, pp. 693–732,

Jan. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0307904X14003448

[56] K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence Modeling in the Age of Data," *Annual Review of Fluid Mechanics*, vol. 51, no. 1, pp. 357–377, 2019, _eprint: https://doi.org/10.1146/annurev-fluid-010518-040547. [Online]. Available: https://doi.org/10.1146/annurev-fluid-010518-040547

[57] B. Font, G. D. Weymouth, V.-T. Nguyen, and O. R. Tutty, "Deep learning of the spanwise-averaged Navier–Stokes equations," *Journal of Computational Physics*, vol. 434, p. 110199, Jun. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999121000942

[58] X. Guo, W. Li, and F. Iorio, "Convolutional Neural Networks for Steady Flow Approximation," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 481–490, event-place: San Francisco, California, USA. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939738

[59] J. Rabault, M. Kuchta, A. Jensen, U. Reglade, and N. Cerardi, "Artificial Neural Networks trained through Deep Reinforcement Learning discover control strategies for active flow control," *arXiv:1808.07664 [physics]*, Dec. 2018, arXiv: 1808.07664. [Online]. Available: http://arxiv.org/abs/1808.07664

[60] S. Verma, G. Novati, and P. Koumoutsakos, "Efficient collective swimming by harnessing vortices through deep reinforcement learning," *arXiv:1802.02674 [physics]*, Feb. 2018, arXiv: 1802.02674. [Online]. Available: http://arxiv.org/abs/1802.02674

[61] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, and E. Hachem, "A review on deep reinforcement learning for fluid mechanics," *Computers & Fluids*, vol. 225, p. 104973, Jul. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045793021001407

[62] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, Jan. 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0893608089900208

[63] M. W. M. G. Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," *Communications in*

Numerical Methods in Engineering, vol. 10, no. 3, pp. 195–201, 1994, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cnm.1640100303. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1640100303

[64] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, Sep. 1998, conference Name: IEEE Transactions on Neural Networks.

[65] M. Raissi, "Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations," *arXiv:1801.06637 [cs, math, stat]*, Jan. 2018, arXiv: 1801.06637. [Online]. Available: http://arxiv.org/abs/1801.06637

[66] 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*, vol. 378, pp. 686–707, Feb. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999118307125

[67] B. Reyes, A. A. Howard, P. Perdikaris, and A. M. Tartakovsky, "Learning Unknown Physics of non-Newtonian Fluids," *arXiv:2009.01658 [physics]*, Aug. 2020, arXiv: 2009.01658. [Online]. Available: http://arxiv.org/abs/2009.01658

[68] P. P. Mehta, G. Pang, F. Song, and G. E. Karniadakis, "Discovering a universal variable-order fractional model for turbulent Couette flow using a physics-informed neural network," *Fractional Calculus and Applied Analysis*, vol. 22, no. 6, pp. 1675–1688, Dec. 2019. [Online]. Available: http://www.degruyter.com/view/j/fca.2019.22.issue-6/fca-2019-0086/fca-2019-0086.xml

[69] A. M. Tartakovsky, C. O. Marrero, P. Perdikaris, G. D. Tartakovsky, and D. Barajas-Solano, "Physics-Informed Deep Neural Networks for Learning Parameters and Constitutive Relationships in Subsurface Flow Problems," *Water Resources Research*, vol. 56, no. 5, p. e2019WR026731, 2020. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019WR026731

[70] R. Tipireddy, P. Perdikaris, P. Stinis, and A. Tartakovsky, "A comparative study of physics-informed neural network models for learning unknown dynamics and constitutive relations," *arXiv:1904.04058 [physics]*, Apr. 2019, arXiv: 1904.04058. [Online]. Available: http://arxiv.org/abs/1904.04058

[71] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, Mar. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045782519306814

[72] M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, "Deep learning of vortex-induced vibrations," *Journal of Fluid Mechanics*, vol. 861, pp. 119–137, Feb. 2019. [Online]. Available: https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/deep-learning-of-vortexinduced-vibrations/B7D9B152C42B7F1E895C1661F5A85881

[73] C. Rao, H. Sun, and Y. Liu, "Physics informed deep learning for computational elastodynamics without labeled data," Jun. 2020. [Online]. Available: https://arxiv.org/abs/2006.08472v1

[74] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, "A deep learning framework for solution and discovery in solid mechanics: linear elasticity," *arXiv:2003.02751 [cs, stat]*, Feb. 2020, arXiv: 2003.02751. [Online]. Available: http://arxiv.org/abs/2003.02751

[75] S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk, "Transfer learning enhanced physics informed neural network for phase-field modeling of fracture," *Theoretical and Applied Fracture Mechanics*, vol. 106, p. 102447, Apr. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016784421930357X

[76] Y. Xu, J. Li, and X. Chen, "Physics informed neural networks for velocity inversion." Society of Exploration Geophysicists, Sep. 2019. [Online]. Available: https://www.onepetro.org/conference-paper/SEG-2019-3216823

[77] K. Shukla, P. C. Di Leoni, J. Blackshire, D. Sparkman, and G. E. Karniadakis, "Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks," *arXiv:2005.03596 [cs, stat]*, May 2020, arXiv: 2005.03596. [Online]. Available: http://arxiv.org/abs/2005.03596

[78] X. Hu and N. E. Buris, "A Deep Learning Framework for Solving Rectangular Waveguide Problems," p. 3.

[79] Y. Chen, L. Lu, G. E. Karniadakis, and L. D. Negro, "Physics-informed neural networks for inverse problems in nano-optics and metamaterials," *arXiv:1912.01085 [physics]*, Mar. 2020, arXiv: 1912.01085. [Online]. Available: http://arxiv.org/abs/1912.01085

[80] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," *arXiv:2105.09506 [physics]*, May 2021, arXiv: 2105.09506. [Online]. Available: http://arxiv.org/abs/2105.09506

[81] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, pp. 1–19, May 2021, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s42254-021-00314-5

[82] S. Wang, X. Yu, and P. Perdikaris, "When and why PINNs fail to train: A neural tangent kernel perspective," *arXiv:2007.14527 [cs, math, stat]*, Jul. 2020, arXiv: 2007.14527. [Online]. Available: http://arxiv.org/abs/2007.14527

[83] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient pathologies in physics-informed neural networks," *arXiv:2001.04536 [cs, math, stat]*, Jan. 2020, arXiv: 2001.04536. [Online]. Available: http://arxiv.org/abs/2001.04536

[84] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis, "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks," *Journal of Computational Physics*, p. 109136, Nov. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999119308411

[85] S. Mishra and R. Molinaro, "Estimates on the generalization error of Physics Informed Neural Networks (PINNs) for approximating PDEs," *arXiv:2006.16144 [cs, math]*, Jun. 2020, arXiv: 2006.16144. [Online]. Available: http://arxiv.org/abs/2006.16144

[86] Y. Shin, J. Darbon, and G. E. Karniadakis, "On the Convergence and generalization of Physics Informed Neural Networks," *arXiv:2004.01806 [cs, math]*, Apr. 2020, arXiv: 2004.01806. [Online]. Available: http://arxiv.org/abs/2004.01806

[87] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[88] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, Jan. 1964. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0041555364901375

[89] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," p. 39.

[90] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning," *Coursera, video lectures*, vol. 264, no. 1, pp. 2146–2153, 2012.

[91] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017, arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980

[92] S. Cai, Z. Wang, F. Fuest, Y.-J. Jeon, C. Gray, and G. E. Karniadakis, "Flow over an espresso cup: Inferring 3D velocity and pressure fields from tomographic background oriented schlieren videos via physics-informed neural networks," *arXiv:2103.02807 [physics]*, Mar. 2021, arXiv: 2103.02807. [Online]. Available: http://arxiv.org/abs/2103.02807

[93] J. M. Pérez, S. Le Clainche, and J. M. Vega, "Reconstruction of three-dimensional flow fields from two-dimensional data," *Journal of Computational Physics*, p. 109239, Jan. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0021999120300139

[94] S. Weinzierl, "Introduction to Monte Carlo methods," *arXiv:hep-ph/0006269*, Jun. 2000, arXiv: hep-ph/0006269. [Online]. Available: http://arxiv.org/abs/hep-ph/0006269

[95] G. Peter Lepage, "A new algorithm for adaptive multidimensional integration," *Journal of Computational Physics*, vol. 27, no. 2, pp. 192–203, May 1978. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0021999178900049

[96] K. Sigman, "Acceptance-rejection method," in *Columbia University*, 2007. [Online]. Available: http://www.columbia.edu/~ks20/4703-Sigman/4703-07-Notes-ARM.pdf

[97] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations," *arXiv:2003.06496 [physics]*, Mar. 2020, arXiv: 2003.06496. [Online]. Available: http://arxiv.org/abs/2003.06496

[98] "Quasi-Newton Methods," in *Numerical Optimization*, ser. Springer Series in Operations Research and Financial Engineering, J. Nocedal and S. J. Wright, Eds. New York, NY: Springer, 2006, pp. 135–163. [Online]. Available: https://doi.org/10.1007/978-0-387-40065-5_6

[99] "minimize(method='L-BFGS-B') — SciPy v1.7.0 Manual." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/optimize.minimize-lbfgsb.html

[100] "AdamOptimizer - TensorFlow Core v2.5.0." [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/compat/v1/train/AdamOptimizer

[101] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine Learning for Fluid Mechanics," *Annual Review of Fluid Mechanics*, vol. 52, no. 1, pp. 477–508, 2020, _eprint: https://doi.org/10.1146/annurev-fluid-010719-060214. [Online]. Available: https://doi.org/10.1146/annurev-fluid-010719-060214

[102] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi, "Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control," *Journal of Fluid Mechanics*, vol. 865, pp. 281–302, Apr. 2019, publisher: Cambridge University Press. [Online]. Available: https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/artificial-neural-networks-trained-through-deep-reinforcement-learning-discover-control-strategies-fo D5B80D809DFFD73760989A07F5E11039

[103] M. Corbetta, "Application of sparse identification of nonlinear dynamics for physics-informed learning," in *2020 IEEE Aerospace Conference*, Mar. 2020, pp. 1–8, iSSN: 1095-323X.

[104] W. Peng, W. Zhou, J. Zhang, and W. Yao, "Accelerating Physics-Informed Neural Network Training with Prior Dictionaries," *arXiv:2004.08151 [cs, stat]*, Apr. 2020, arXiv: 2004.08151. [Online]. Available: http://arxiv.org/abs/2004.08151

[105] T. Zhang, B. Dey, P. Kakkar, A. Dasgupta, and A. Chakraborty, "Frequency-compensated PINNs for Fluid-dynamic Design Problems," *arXiv:2011.01456 [cs]*, Nov. 2020, arXiv: 2011.01456. [Online]. Available: http://arxiv.org/abs/2011.01456

[106] D. Zhang, L. Guo, and G. E. Karniadakis, "Learning in Modal Space: Solving Time-Dependent Stochastic PDEs Using Physics-Informed Neural Networks," *arXiv:1905.01205 [physics, stat]*, May 2019, arXiv: 1905.01205. [Online]. Available: http://arxiv.org/abs/1905.01205

[107] C. Bailly and G. Comte-Bellot, "Experimental Methods," in *Turbulence*. Cham: Springer International Publishing, 2015, pp. 269–316.

[108] C. J. Kähler, S. Scharnowski, and C. Cierpka, "On the uncertainty of digital PIV and PTV near walls," *Exp Fluids*, vol. 52, no. 6, pp. 1641–1656, Jun. 2012. [Online]. Available: https://doi.org/10.1007/s00348-012-1307-3

[109] F. Durst and E. Zanoun, "Experimental investigation of near-wall effects on hot-wire measurements," *Exp Fluids*, vol. 33, no. 1, pp. 210–218, Jul. 2002. [Online]. Available: https://doi.org/10.1007/s00348-002-0472-1

[110] V. Aeschlimann, S. Beaulieu, S. Houde, G. D. Ciocan, and C. Deschênes, "Inter-blade flow analysis of a propeller turbine runner using stereoscopic PIV," *European Journal of Mechanics - B/Fluids*, vol. 42, pp. 121–128, Nov. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0997754613000642

[111] M. R. Cholemari, "Modeling and correction of peak-locking in digital PIV," *Exp Fluids*, vol. 42, no. 6, pp. 913–922, Jun. 2007. [Online]. Available: https://doi.org/10.1007/s00348-007-0300-8

[112] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM J. Sci. Comput.*, vol. 16, no. 5, pp. 1190–1208, Sep. 1995, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/0916069

[113] U. Fey, M. König, and H. Eckelmann, "A new Strouhal–Reynolds-number relationship for the circular cylinder in the range 47," *Physics of Fluids*, vol. 10, no. 7, pp. 1547–1549, Jun. 1998, publisher: American Institute of Physics. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.869675

[114] M. Boudina, "Numerical simulation data of a two-dimensional flow around a fixed circular cylinder," Jun. 2021, type: dataset. [Online]. Available: https://zenodo.org/record/5039610

[115] S. Étienne, A. Garon, and D. Pelletier, "Perspective on the geometric conservation law and finite element methods for ALE simulations of incompressible flow," *Journal of Computational Physics*, vol. 228, no. 7, pp. 2313–2333, Apr. 2009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0021999108006165

[116] A. Hay, S. Etienne, D. Pelletier, and A. Garon, "hp-Adaptive time integration based on the BDF for viscous flows," *Journal of Computational Physics*, vol. 291, pp. 151–176, Jun. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999115001692

[117] M. Rosenfeld, "Utilization of Fourier decomposition for analyzing time-periodic flows," *Computers & Fluids*, vol. 24, no. 4, pp. 349–368, May 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/004579309400043X

[118] H. W. Coleman, W. G. Steele Jr, and W. G. Steele, *Experimentation and uncertainty analysis for engineers*. John Wiley & Sons, 1999.

[119] A. Arzani, J.-X. Wang, and R. M. D'Souza, "Uncovering near-wall blood flow from sparse data with physics-informed neural networks," *arXiv:2104.08249 [physics]*, Apr. 2021, arXiv: 2104.08249. [Online]. Available: http://arxiv.org/abs/2104.08249

[120] G. Raynaud, "ModalPINN Python Code," 2021. [Online]. Available: https://gitfront.io/r/user-5103109/7bfe50d1abac8e606dc47386e4fb82670e90cfe3/ModalPINN-Python-code/

[121] R. W. Noack and D. A. Anderson, "Solution-adaptive grid generation using parabolic partial differential equations," *AIAA Journal*, vol. 28, no. 6, pp. 1016–1023, 1990, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/3.25159. [Online]. Available: https://doi.org/10.2514/3.25159

[122] Y. C. Liou and Y. N. Jeng, "Parabolic Equation Method of Grid Generation for Enclosed Regions," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 29, no. 3, pp. 289–303, Apr. 1996, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10407799608914983. [Online]. Available: https://doi.org/10.1080/10407799608914983

[123] J.-L. Wu, H. Xiao, and E. Paterson, "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework," *Phys. Rev. Fluids*, vol. 3, no. 7, p. 074602, Jul. 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevFluids.3.074602

[124] G. J. Chandler, M. P. Juniper, J. W. Nichols, and P. J. Schmid, "Adjoint algorithms for the Navier–Stokes equations in the low Mach number limit," *Journal of Computational Physics*, vol. 231, no. 4, pp. 1900–1916, Feb. 2012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0021999111006681

[125] L. F. R. Espath, R. V. Linn, and A. M. Awruch, "Shape optimization of shell structures based on NURBS description using automatic differentiation," *International Journal for Numerical Methods in Engineering*, vol. 88, no. 7, pp. 613–636,

2011, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.3183. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.3183

[126] L. Chen and H. M. H. Shen, "Topology Optimization through Differentiable Finite Element Solver," *arXiv:2009.10072 [cs]*, Sep. 2020, arXiv: 2009.10072. [Online]. Available: http://arxiv.org/abs/2009.10072

[127] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, "hp-VPINNs: Variational Physics-Informed Neural Networks With Domain Decomposition," *arXiv:2003.05385 [cs, math]*, Mar. 2020, arXiv: 2003.05385. [Online]. Available: http://arxiv.org/abs/2003.05385

[128] F. Chinesta, P. Ladeveze, and E. Cueto, "A Short Review on Model Order Reduction Based on Proper Generalized Decomposition," *Arch Computat Methods Eng*, vol. 18, no. 4, pp. 395–404, Nov. 2011, company: Springer Distributor: Springer Institution: Springer Label: Springer Number: 4 Publisher: Springer Netherlands. [Online]. Available: https://link.springer.com/article/10.1007/s11831-011-9064-7

[129] B. M. de Silva, D. M. Higdon, S. L. Brunton, and J. N. Kutz, "Discovery of Physics From Data: Universal Laws and Discrepancies," *Front. Artif. Intell.*, vol. 3, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frai.2020.00025/full

[130] E. Zhang, M. Yin, and G. E. Karniadakis, "Physics-Informed Neural Networks for Nonhomogeneous Material Identification in Elasticity Imaging," *arXiv:2009.04525 [cond-mat]*, Sep. 2020, arXiv: 2009.04525. [Online]. Available: http://arxiv.org/abs/2009.04525

[131] P. Véras, G. Balarac, O. Métais, D. Georges, A. Bombenger, and C. Ségoufin, "Influence of turbulent inlet conditions on the flow inside a bulb turbine draft tube using Large-Eddy Simulations," *IOP Conf. Ser.: Earth Environ. Sci.*, vol. 774, no. 1, p. 012014, Jun. 2021. [Online]. Available: https://iopscience.iop.org/article/10.1088/1755-1315/774/1/012014

[132] C. Frey, G. Ashcroft, H.-P. Kersken, and C. Voigt, "A Harmonic Balance Technique for Multistage Turbomachinery Applications," in *Volume 2B: Turbomachinery*. Düsseldorf, Germany: American Society of Mechanical Engineers, Jun. 2014, p. V02BT39A005. [Online]. Available: https://asmedigitalcollection.asme.org/GT/proceedings/GT2014/45615/D%C3%BCsseldorf,%20Germany/250350

[133] K. Taira, M. S. Hemati, S. L. Brunton, Y. Sun, K. Duraisamy, S. Bagheri, S. T. M. Dawson, and C.-A. Yeh, "Modal Analysis of Fluid Flows: Applications and Outlook,"

AIAA Journal, vol. 58, no. 3, pp. 998–1022, 2020, publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.J058462. [Online]. Available: https://doi.org/10.2514/1.J058462

[134] B. G. Dehkordi, H. S. Moghaddam, and H. H. Jafari, "Numerical Simulation of Flow Over Two Circular Cylinders in Tandem Arrangement," *J Hydrodyn*, vol. 23, no. 1, pp. 114–126, Feb. 2011. [Online]. Available: http://link.springer.com/10.1016/S1001-6058(10)60095-9

[135] M. Païdoussis, "A review of flow-induced vibrations in reactors and reactor components," *Nuclear Engineering and Design*, vol. 74, no. 1, pp. 31–60, Jan. 1983. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0029549383901383

APPENDIX A SUPPLEMENTARY MATERIAL ON MODALPINN

A.1 A summary on spectral methods and modal analysis

Especially in CFD where computations can be carried out using millions of degrees of freedoms (and often more with the number of nodes that skyrockets in meshes designed for turbulence), it can be practical to reduce the complexity of a flow to a lower order space of modes that we can eventually truncate. This can help reducing a dimension or better understanding some global patterns. From a mathematical point of view, it consists in decomposing a field as a weighted sum of basis functions defined globally over the domain. This can be practical to solve directly some PDE. Especially when PDEs are linear and when the basis functions have good mathematical properties (we know their derivatives, or the functions are orthogonal to one another...). For instance some PDE can be analytically solved using Fourier transform like wave equations. In a certain way, the FE method can also be expressed as a spectral method since it decomposes the solution over a basis of continuous functions that are non zero only on a sub part of the domain called an element.

Data driven approaches use the same kind of decomposition. However the objective is not directly to solve a PDE bu rather finding patterns in data already collected. Moreover, once these patterns are found, it can act as a filter to remove less interesting or noisy parts in the data and finally compressing the information for storage. A review of modal analysis techniques for fluid flows by Taira et al. [45] emphasizes on these point of interests while recalling some of these techniques.

Among these techniques, Harmonic Balance Method (HBM) [132] uses a finite basis of $e^{i\omega_k t}$ so that the time dimension is reduced to a finite number of Fourier coefficients (that depends of space). The proposed ModalPINN is a direct application of this, excepted that the modal coefficients are mode shapes, i.e. continuous functions of space defined by a neural networks. These techniques are well suited when the boundary conditions (in time or space) are periodic since we use a periodic set of basis functions.

When data are already available, Dynamic Mode Decomposition (DMD) [133] provides a list of spatial mode shapes associated with one frequency which is very similar to the Fourier decomposition (except growth rate are usually taken into account in DMD whereas we did

not in the presented version of ModalPINN). In DMD, a given snapshot of a field x_k at time step $t_k = t_0 + k\Delta t$ is concatenated in a matrix and then a linear operator A is fitted so that the residuals of

$$\begin{bmatrix} x_{k+1} \\ \vdots \\ x_{N+k+1} \end{bmatrix} = A \begin{bmatrix} x_k \\ \vdots \\ x_{N+k} \end{bmatrix} \tag{A.1}$$

are as low as possible. Then the eigenvectors of A are DMD modes and the complex eigenvalues provides a frequency and a growth rate.

Proper Orthogonal Decomposition (POD) [133] uses a more general decomposition of an average-free function of space and time $u(x,t)$:

$$u(x,t) = \sum_k a_k(t)\phi_k(x) \tag{A.2}$$

where both a_k and ϕ_k are unknown but the space functions are set to be orthogonal the one between each others: $\int_\Omega \phi_k(x)\phi_j(x)dx = 0$ if $k \neq j$. The objective is to minimize the quadratic difference between the truncated sum and the complete data u. Several methods exists for solving this problem : spatial POD, snapshot POD, a Singular Value Decomposition (SVD) and even a linear auto-encoder using a one hidden Neural Network with linear activation functions trained with gradient descent [101]. Once the mode shapes $\phi_j(x)$ are known, time coefficients can be obtained by a direct projection $a_k(t) = \int_\Omega u(x,t)\phi_k(x)dx$. It can be noted that as POD minimizes the quadratic error, the decomposition is optimal in terms of kinetic energy so that the number of modes required for a certain phenomena and a given error threshold is minimal in L^2 sense.

The choice of Fourier approach for the ModalPINN presented in chapter 4 have been motivated by the simplicity of the decomposition. Only the space remains unknown and is easier to approximate using an NN. But it would be quite direct to generalize ModalPINN to other type of decompositions. For instance, growth rate can be taken into account using an additional complex variable in the list of frequencies. Multiple fundamental frequencies can also be used so that we can address non periodic phenomena. Even POD could be implemented in a PINN. As summarized in figure A.1, a second NN could approximate the functions of time $a_k(t)$, the modal sum would be implemented in the graph of operation in the exact same way as with the ModalPINN. There would be an additional penalization term $\mathcal{L}_{ort}$ in the loss for making sure that the obtained mode shapes are orthogonal.

A.2 Performance of ModalPINN on more complex flow patterns

It is a matter of interest to generalize the framework presented in this chapter to more realistic flows. To explore this direction, some tests with flows over one body have been performed with a more complex shape using a turbine blade at a high angle of attack. We have also used another type of geometry with an array of several cylinders. The goal is to observe if the presented framework generalizes well.

A.2.1 Periodic flow over a 2D turbine blade

Flow reconstruction over a non-canonical geometry have been carried out using a blade profile at a high angle of attack. The geometry comes from the CAD file of a guide vane from a Francis turbine which is depicted at figure A.2a. A numerical simulation using `ANSYS CFX` is carried out on the domain presented at figure A.2b which consists of a rectangle with free-slip boundaries on top and bottom, a uniform inlet velocity $(u, v) = (U_\infty, 0)$ and an average zero pressure at the outlet. The Reynolds number based on the inlet velocity U_∞, the approximated chord-length L_0 and water properties is $Re = \frac{U_\infty L_0}{\nu} = 168$ and vortex shedding occurs at a dimensionless frequency $\omega_0 = 2\pi f_0 = 1.89\frac{L_0}{U_\infty}$. After 6000 iterations using a large time-step $\Delta t_1 = 7.5 \times 10^{-2}\frac{L_0}{U_\infty}$ corresponding to 22.5 convection time over the entire computational domain to initialize the simulation, a second run of 6000 iterations is carried out using a lower time step $\Delta t_2 = 1.5 \times 10^{-2}\frac{L_0}{U_\infty}$ corresponding to 4.5 convection times over the whole domain. Then, 100 snapshots over 3 vortex shedding periods are extracted from the second run, made dimensionless using L_0, U_∞ and ρ_{water}. Then these data are cropped to a domain for ModalPINN reconstruction consisting of a rectangular box of 9×5 centred on the blade.

One of the challenge is to adapt the ModalPINN to a non canonical body shape, especially for boundary penalization or for the Prior-Dictionary approach. If we chose to use the penalization of the no-slip condition, a sampling of points directly obtained from a CAD file (that encode the frontier as a list of points) can be used. However, this is not enough in order to get the forces on the boundary. Indeed a value for the normal vector is required. In the previous example, the normal vector was obtained using a symbolic parametric definition of the boundary $s \in [0, 1] \to x_{bc}(s), y_{bc}(s)$ and by differentiating it according to equation

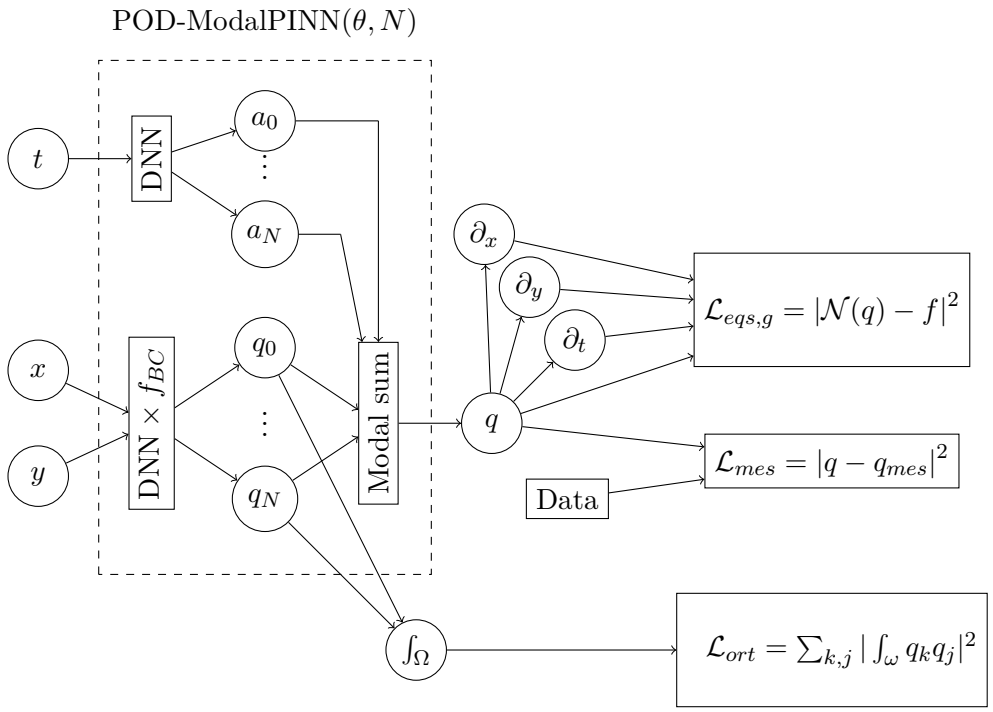


Figure A.1 Some ideas to adapt a ModalPINN for POD

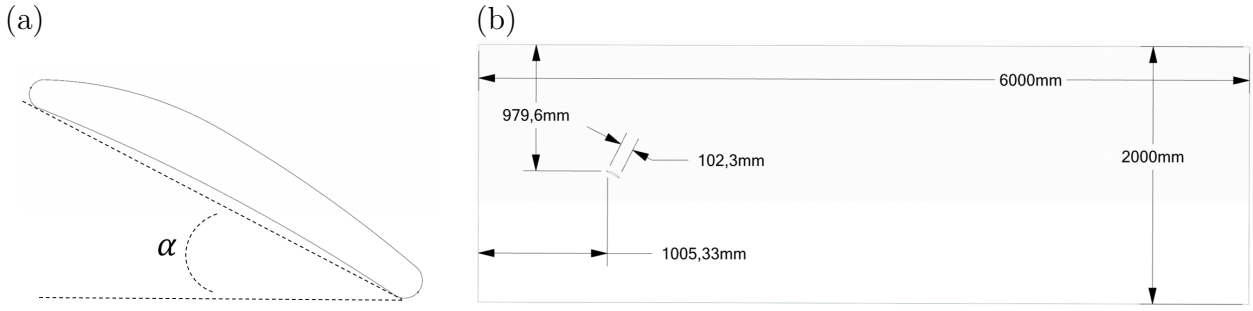


Figure A.2 (a) Geometry of the turbine blade at an angle $\alpha \approx 25°$. (b) Computational domain with dimensions

3.25. One way of solving this is to use an auxiliary neural network NN_{bc} which is trained before the actual flow reconstruction (and kept untouched once it is converged). There are two possibilities for this approximation:

- Using an NN that approximates the two coordinates directly $s \to x_{bc}(s), y_{bc}(s)$. In that case we can enforce the periodicity either by penalizing $x_{bc}(1) = x_{bc}(0)$ or by using a periodic input for the neural network : $NN : \sin(2\pi s), \cos(2\pi s) \to x_{bc}(s), y_{bc}(s)$.
- Using an NN that approximates the radius of the frontier r_c as a function of the angle θ as depicted in figure A.3. In that case, one needs a geometry which is convex and a centre needs to be defined arbitrary. One of the problem of this method is that the function $r_c(\theta)$ can have sharp variation especially if the profile has a large aspect ratio. Then the coordinates are obtained by $x_{bc}(s) = x_c + r_c(2\pi s)\cos(2\pi s)$ and $y_{bc}(s) = y_c + r_c(2\pi s)\sin(2\pi s)$.

In both cases, the obtained function is continuous and derivable in the graph of operation thanks to back propagation.

In the case of Prior-Dictionary, we are looking for a function of space f_{bc} that is symbolic, almost equal to 1 and flat far from the frontier and equal to 0 on its border. A direct transposition of the function used in the case of the cylinder would be to use the previously learned radius $r_c(\theta)$ as a function of an angle. That would give

$$\begin{aligned} f_{bc}(x,y) &= \tanh\left[\gamma(r - r_c(\theta))\right], \\ r &= \sqrt{(x-x_c)^2 + (y-y_c)^2}, \\ \theta &= \text{angle}(x - x_c, y - y_c). \end{aligned} \tag{A.3}$$

with the same choice of factor γ as discussed in the article. The result of this approach is depicted at figure A.4. This function depicts strong gradients especially when $\left\| \frac{\partial r_c}{\partial \theta} \right\| \gg 1$, which occurs near the edges of the blade. It is highly possible that these sharp gradients impact the initial PINN and therefore affects the optimization process.

Another solution that has been used is to create a map of points in the fluid domain and compute their distance to the nearest point of the frontier $\partial\Omega$. Then a NN can be trained directly to satisfy $f_{bc}(x,y) = NN(x,y) = \text{distance}\left[(x,y), \partial\Omega\right]$. Result of this approach is plotted at figure A.5 and the iso-contour of $f_{bc} = 0$ that is nearly indistinguishable from

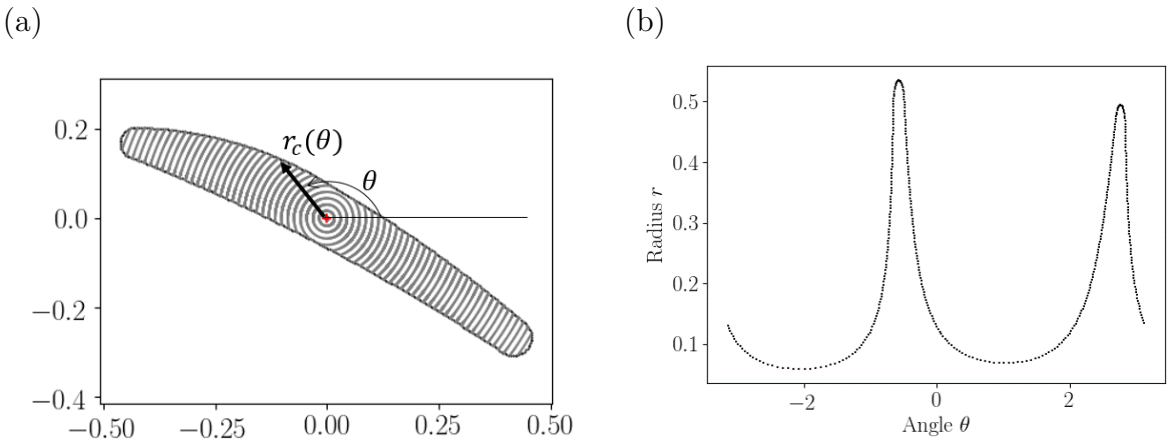


Figure A.3 Approximation of the boundary using a changing radius : position of the centre (a) and the obtained radius (b) that is to be learned by the auxiliary NN.

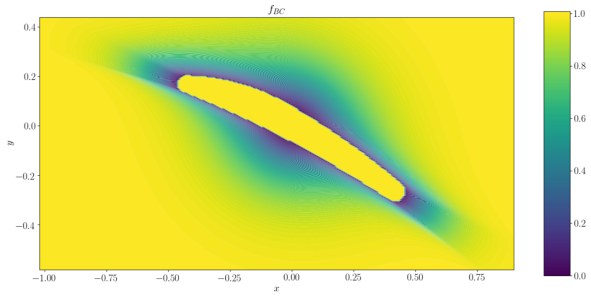


Figure A.4 Illustration of the prior-dictionary obtained using the moving radius approach

CAD data proves that this approach may work.

Mode shapes obtained from a flow reconstruction using dense data are pictured at figure A.8. Properties of this run are summarized at table A.1. Besides, a prior dictionary based on the distance mapping have been used and the frontier is also encoded with the $r_c(\theta)$ approach. These mode shapes are qualitatively consistent with those extracted from CFD snapshots using a direct Fourier transform. Moreover the error on the no-slip boundary is at a similar order of magnitude (between 10^{-4} and 10^{-3}) compared to other studies where the boundary condition is imposed by penalization. This is an encouraging sign for generalizing PINNs to non canonical shapes.

A.2.2 Periodic flow over an array of cylinders

The studied geometry consists in 3 identical cylinders placed in a uniform flow as depicted in figure A.6. The computational domain is chosen large in the transverse direction with free-slip boundary conditions on the top and on the bottom so that their effects on the flow around the cylinders might not be too important. The inlet condition is a uniform flow speed $(u, v) = (U_\infty, 0)$ and the outlet is obtained by imposing a zero averaged pressure. The simulation is conducted in `ANSYS CFX` which uses a finite-volume method and a second order Backward Euler scheme for time integration. After the first iterations with an adaptive time-step to reach the periodic regime, time integration was performed with a constant time-step $\Delta t = 1 \times 10^{-2} d/U_\infty$ and extraction of velocity and pressure data were performed every 10 time-steps. Finally, every quantity is made dimensionless using the inlet velocity U_∞, fluid viscosity ν, its density ρ and the diameter of the cylinder d. The obtained Reynolds number with respect to one cylinder is $Re = \frac{U_\infty d}{\nu} = 112$ so that the flow remains laminar.

Similarly to the vortex shedding past one cylinder, flow-field reconstruction have been car-

Table A.1 Summary of the flow reconstruction run properties for the turbine blade with dense measurements which mode shapes are presented at figure A.8

Property	Value	Error	Value
N_m	2	No-Slip Boundary	4.9×10^{-4}
Train. time	10 h	Training Error	1.4×10^{-3}
N_{mes}	3×10^4	Equations (training)	9.6×10^{-4}
N_{int}	2×10^4	Meas. (valid.)	4.6×10^{-4}

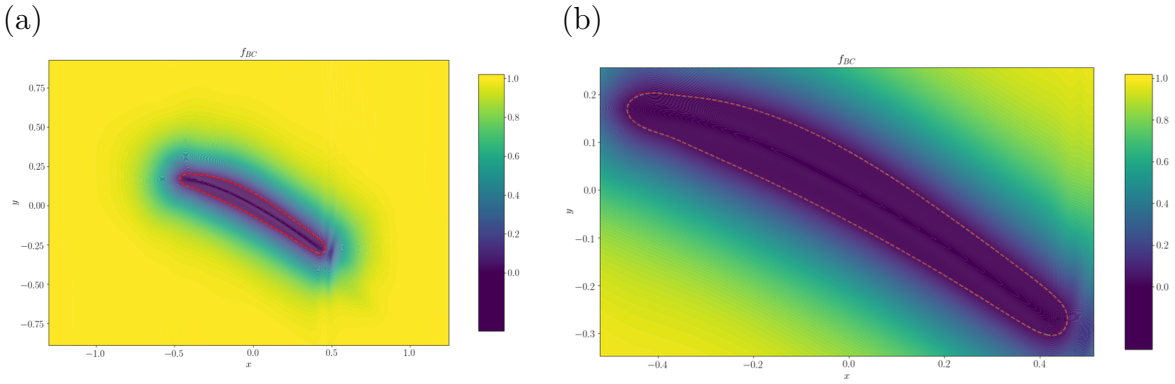


Figure A.5 (a) Learned f_{bc} from a map of distance to the frontier. The close-up (b) compare the position of the frontier from the CAD data (red points) and the iso-contour of $f_{bc} = 0$ (grey line).

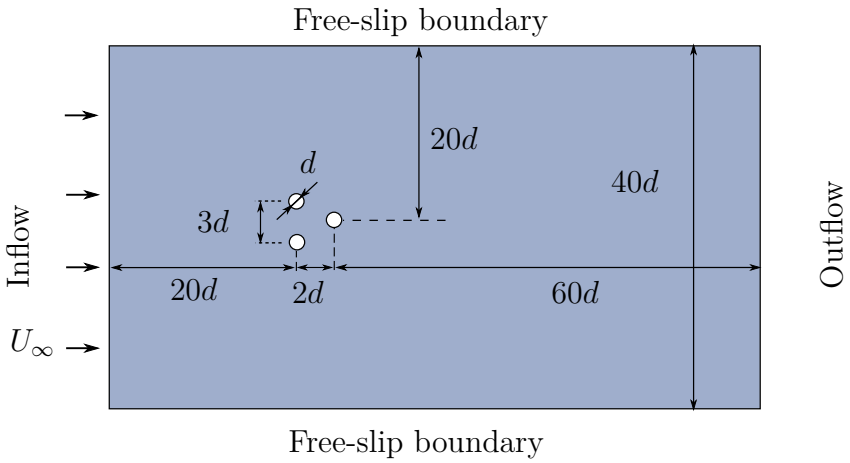


Figure A.6 Configuration of the computational domain in **ANSYS CFX** consisting of an array of 3 cylinders in a uniform flow.

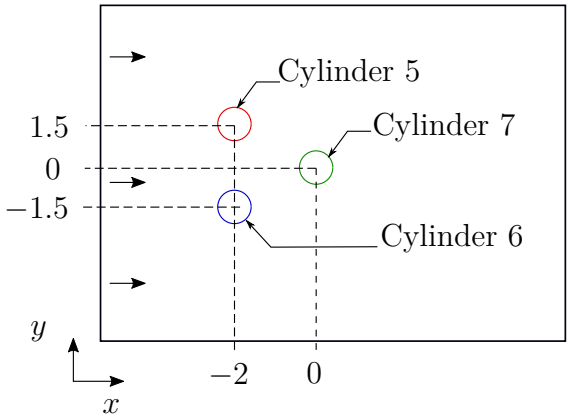


Figure A.7 Configuration of the reduced domain for flow reconstruction using the ModalPINN. Numbers 1-4 corresponds to external boundaries and 5-7 to the three cylinders.

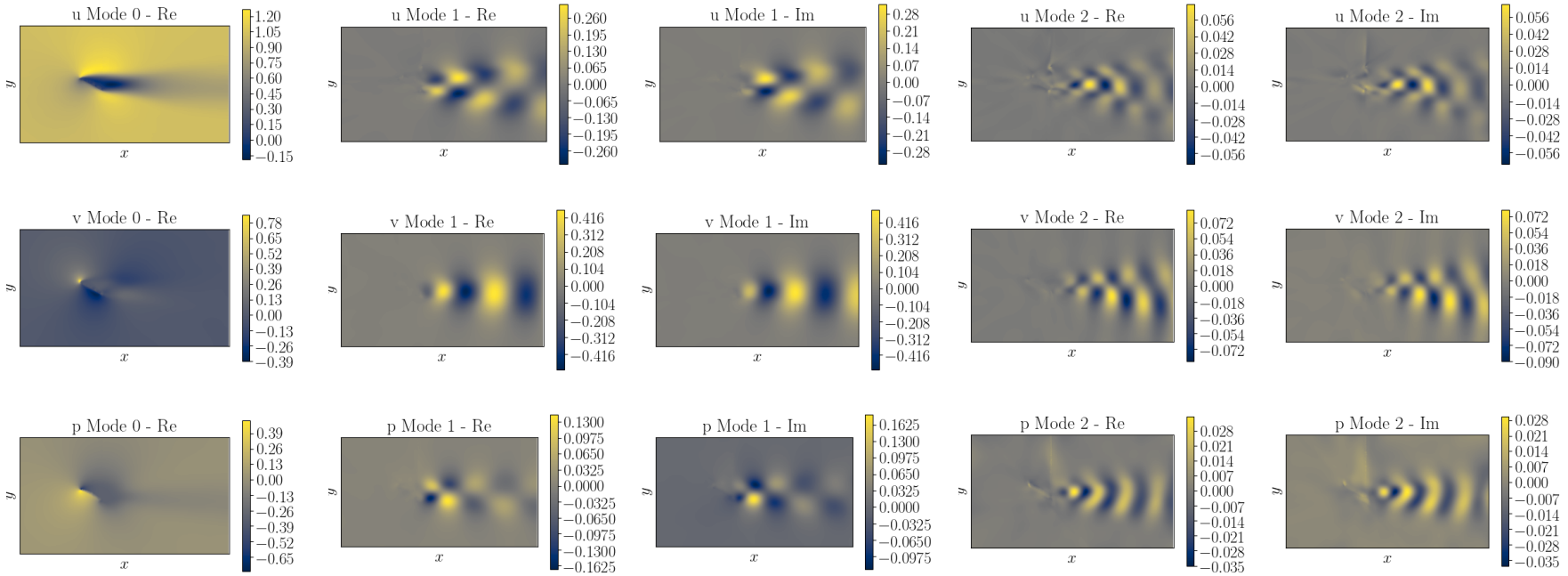


Figure A.8 Mode shapes obtained with ModalPINN using dense measurements for turbine blade flow reconstruction

ried out on a reduced domain consisting of $x, y \in [-7, 8] \times [-6, 6]$, with the origin placed at the centre of the middle-right cylinder (cylinder 7). Snapshots of velocity and pressure fields obtained from `CFX` are plotted at a given time-step at figure A.9b, c and d. A periodic wake is observed downstream the cylinders but with a more complex pattern than with only one cylinder. Indeed there is an interesting interference between the wake of each individual cylinders downstream and also in between the array. This kind of effects are expected in flows around arrays of cylinders. A review of flow patterns around 2 cylinders by Dehkordi et al. [134] depicts coupling of different nature and this has lead research towards fluid-solid vibrations with a particular interest for nuclear reactor as reviewed by Païdoussis et al. [135].

To enforce the no-slip boundary conditions on the frontier of each cylinders, an extension of the Prior-Dictionary is used. It consists in multiplying the output of the velocity PINNs by f_{bc} defined by

$$\begin{aligned} f_{bc}(x, y) &= 1 + \sum_{k=5}^{7} f_k(x, y), \\ f_k(x, y) &= \tanh\left[\gamma \Delta r_k(x, y)\right] - 1, \\ \Delta r_k(x, y) &= \sqrt{(x - x_{c,k})^2 + (y - y_{c,k})^2} - r_c, \end{aligned} \tag{A.4}$$

where $x_{c,k}$ and $y_{c,k}$ denote the coordinates of the centre of each of the 3 cylinders which are recalled in figure A.7. Each f_k is approximately equal to 0 and flat far from the k^{th} cylinder and equal -1 on its frontier. The constant γ that quantifies the area of influence of f_k has been set to $\gamma = 5$ for the same reasons as for the flow around one cylinder. An overview of f_{bc} is depicted in figure A.9a.

The formalism of ModalPINN as presented in the main section of this chapter is used for reconstructing the flow around these 3 cylinders. We focused on four test cases:

1. Reconstruction of the flow using data everywhere in the domain. From a practical point of view, we randomly picked out some measurements in space and time from numerical simulations.

2. Reconstruction of the flow using no information in the area of the cylinders. This simulates some difficulties that we could have of gathering experimental measurements inside an array of cylinders due to reflections of laser beam while using optical methods, or just because of the confinement and the impossibility to put a probe inside the array of cylinders. Data from upstream and downstream have been used to train the

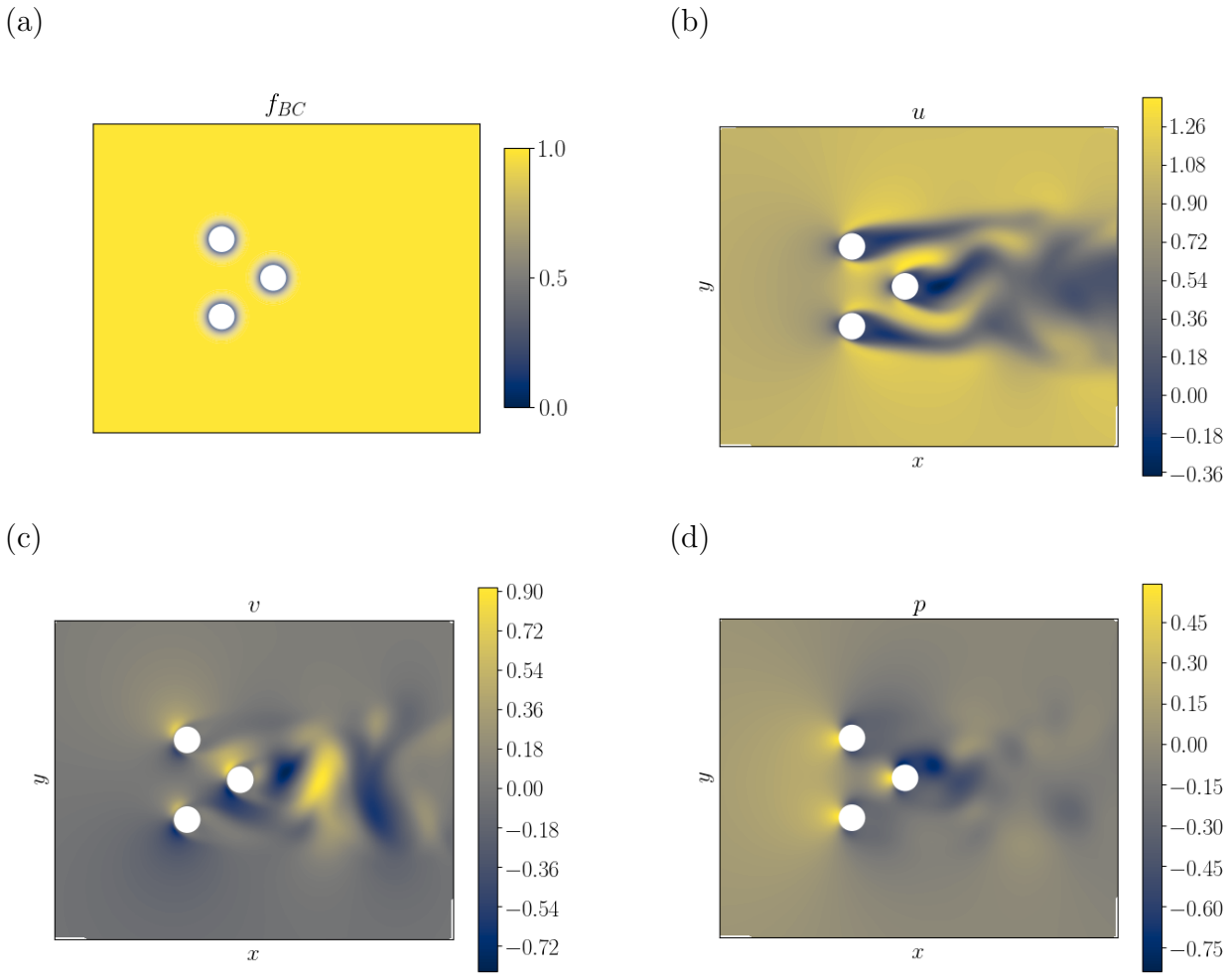


Figure A.9 Adaptation of the prior-dictionary for several bodies (a) and snapshots of velocities u (b) v (c) and pressure (d) fields from numerical simulations at a given time-step.

ModalPINN by randomly selecting measurements points that verify this criteria

$$-4.5 \leq x \leq -3 \text{ or } 1 \leq x \leq 4 \tag{A.5}$$

For comparison, the first row of cylinders is at $x = -2$ and the second is at $x = 0$. An overview of this distribution of points is given at figure A.10a. We will refer to this as a PIV-like sampling of measurement points.

3. Reconstruction of the flow using PIV-like sampling of velocity measurements combined with pressure data around the cylinders to help correcting convergence problems.

4. Reconstruction of the flow as in 3. but with an artificially added noise in PIV-like data.

For each of these approaches, some trainings were run with or without pressure data. When pressure data was not available, a penalization of $p = 0$ on the outlet was used to fix the constant. Moreover, sampling of penalization points for the equations have been tested with slightly different approaches: uniformly or with two zones as in the main section of this chapter. Also a 3 zones approaches was tested to get more accuracy in the area between the cylinders as well as very near to each of the cylinders. An example of such a sampling is plotted at figure A.10b with 2×10^4 points and it is expected to bring a better accuracy in the estimation of forces on cylinders.

Result of a 2 oscillating modes reconstruction with a Neural Network composed of 2 hidden layers of 75 neurons per layer for each of the 3 variables u, v and p have been trained for 9 h on a GPU V100 card (with 32Go of RAM) with penalization points using a 3 zones layout. No pressure data have been provided excepted $p = 0$ on $x = 8$. For each run described previously, a summary of the run properties, details on the loss and errors on force predictions are provided as well as comparison of the force signals and a snapshot at a given instant. The figures are referenced in table A.2. Mode shapes obtained in ModalPINN trained with dense measurements are depicted at figure A.13.

From these results, the reconstruction of pressure field from dense measurements of velocity only seems to give appropriate results but can suffer from a shift in the constant value between the outlet to the rest of the flow area as pictured in figure A.12c. The error on the prediction of the forces stays under 10 % for the horizontal component and is even lower for the two front cylinders (5 and 6). If we look more closely in figure A.11a, the average value is in good agreement but there are important errors in the oscillating parts. Trans-

Table A.2 Summary of the jobs performed for the flow over 3 cylinders.

Test-case	Data type	Properties	Forces	Snapshots	Mode shapes
1	Dense data	Table A.3	Figure A.11	Figure A.12	Figure A.13
2	PIV-like	Table A.4	Figure A.14	Figure A.15	-
3	PIV-like + pressure probe	Table A.5	Figure A.16	Figure A.17	-
4	noised PIV-like + pressure probe	Table A.6	Figure A.18	Figure A.19	-

verse forces in figure A.11b are nonetheless a bit less retrieved, especially for cylinder 7 with an important error on the amplitude that leads to this relative difference of 68 %. Overall the reconstruction of forces is less satisfying than for the one-cylinder case. Perhaps more penalization points for the equations should be placed closer to the frontier during training to better solve the near-wall flows.

For the PIV-like reconstruction, the wake around cylinders 5 and 6 appears to be qualitatively well rendered. However, there is an area upstream the stagnation point of cylinder 7 where u has a nearly zero value which may not be very physical. This is an issue that we already encountered with the flow around one cylinder without finding an appropriate explanation for that phenomena. This error has some consequences on the reconstruction of pressure at that location which therefore affects the reconstruction of forces especially on cylinder 7 (green lines in figure A.14). Excepted for cylinder 7, the errors on the predictions of the forces are not much larger than it was for the reconstruction using dense measurements.

An array of 30 pressure probes distributed regularly around each cylinders was simulated in the same way as for the case around one cylinder in chapter 4. This pressure data at each time-steps was added to the training loss along the PIV-like measurement of velocities u and v. Comparing the losses at the end of training in tables A.4 and A.5, the added information from the pressure probes allows a gain of one order of magnitude for pressure reconstruction and a loss for u and v divided by two compared with only PIV-like measurements. Regarding the predictions of forces, the increase of precision is about one order of magnitude for each force component. It can be noted that this increase in precision is not only due to the pressure information since the relative L^2 average defined by

$$\sqrt{\frac{\int_0^T \left(C_t(t)\right)^2 dt}{\int_0^T \left(C_t(t)+C_n\right)^2 dt}} \sim 0.25, \tag{A.6}$$

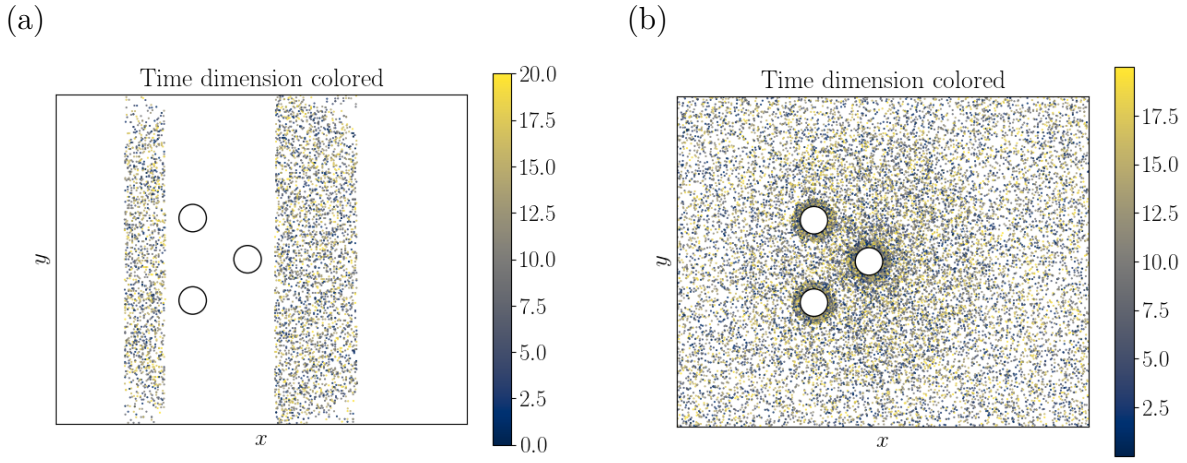


Figure A.10 Sampling of training points for measurements ((a) 5×10^3 points in PIV-like configuration) and for penalization of equations ((b) 2×10^4 points in a 3 zones distribution). For both sampling, the time dimension is colored.

Table A.3 Summary of a run on dense measurements for the flow reconstruction around 3 cylinders

Parameter	Value
Data	Dense
Pressure Mes.	No
Time	9h
N_{int}	1×10^5
N_{mes}	5×10^4
NN size	$[2, 75, 75, 3]$

Error	Value	Error on Forces	Relative	Absolute
BC	7.1×10^{-12}	C_{x5}	6.5×10^{-2}	8.4×10^{-2}
Total (train.)	4.2×10^{-3}	C_{x6}	7.0×10^{-2}	9.1×10^{-2}
Equations (train.)	2.5×10^{-3}	C_{x7}	1.0×10^{-1}	1.3×10^{-1}
Mes. (train.)	1.6×10^{-3}	C_{y5}	1.3×10^{-1}	2.0×10^{-2}
Mes u, v (Valid.)	1.7×10^{-3}	C_{y6}	1.7×10^{-1}	2.5×10^{-2}
Mes p (valid.)	9.5×10^{-4}	C_{y7}	6.3×10^{-1}	2.6×10^{-1}

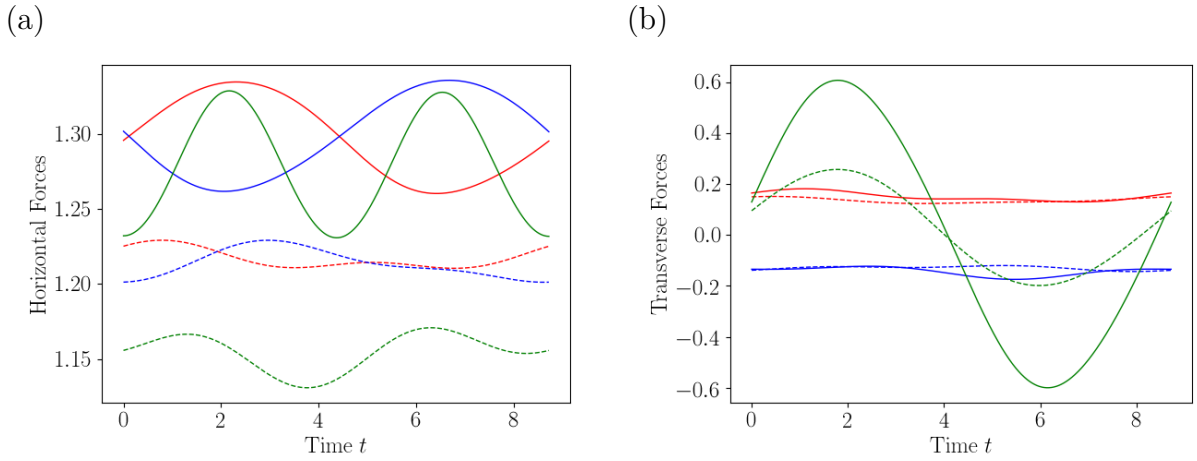


Figure A.11 Estimation of the unsteady drag C_x (a) and lift C_y (b) coefficients with the ModalPINN (dashed lines) using dense measurements and comparison with numerical simulations (solid lines). Cylinder 5 (top left) is in red, cylinder 6 (bottom left) in blue and cylinder 7 (middle right) in green.

Table A.4 Summary of a run with PIV-like measurements for the flow reconstruction around 3 cylinders

Parameter	Value
Data	PIV-like
Pressure Mes.	No
Time	9h
N_{int}	1×10^5
N_{mes}	5×10^4
NN size	$[2, 75, 75, 3]$

Error	Value	Error on Forces	Relative	Absolute
BC	6.2×10^{-12}	C_{x5}	1.1×10^{-1}	1.5×10^{-1}
Total (train.)	5.5×10^{-3}	C_{x6}	1.1×10^{-1}	1.4×10^{-1}
Equations (train.)	3.4×10^{-3}	C_{x7}	7.9×10^{-1}	1.0
Mes. (train.)	2.1×10^{-3}	C_{y5}	1.6×10^{-1}	2.4×10^{-2}
Mes u, v (Valid.)	1.1×10^{-2}	C_{y6}	1.1×10^{-1}	1.6×10^{-2}
Mes p (valid.)	7.2×10^{-3}	C_{y7}	6.8×10^{-1}	2.9×10^{-1}

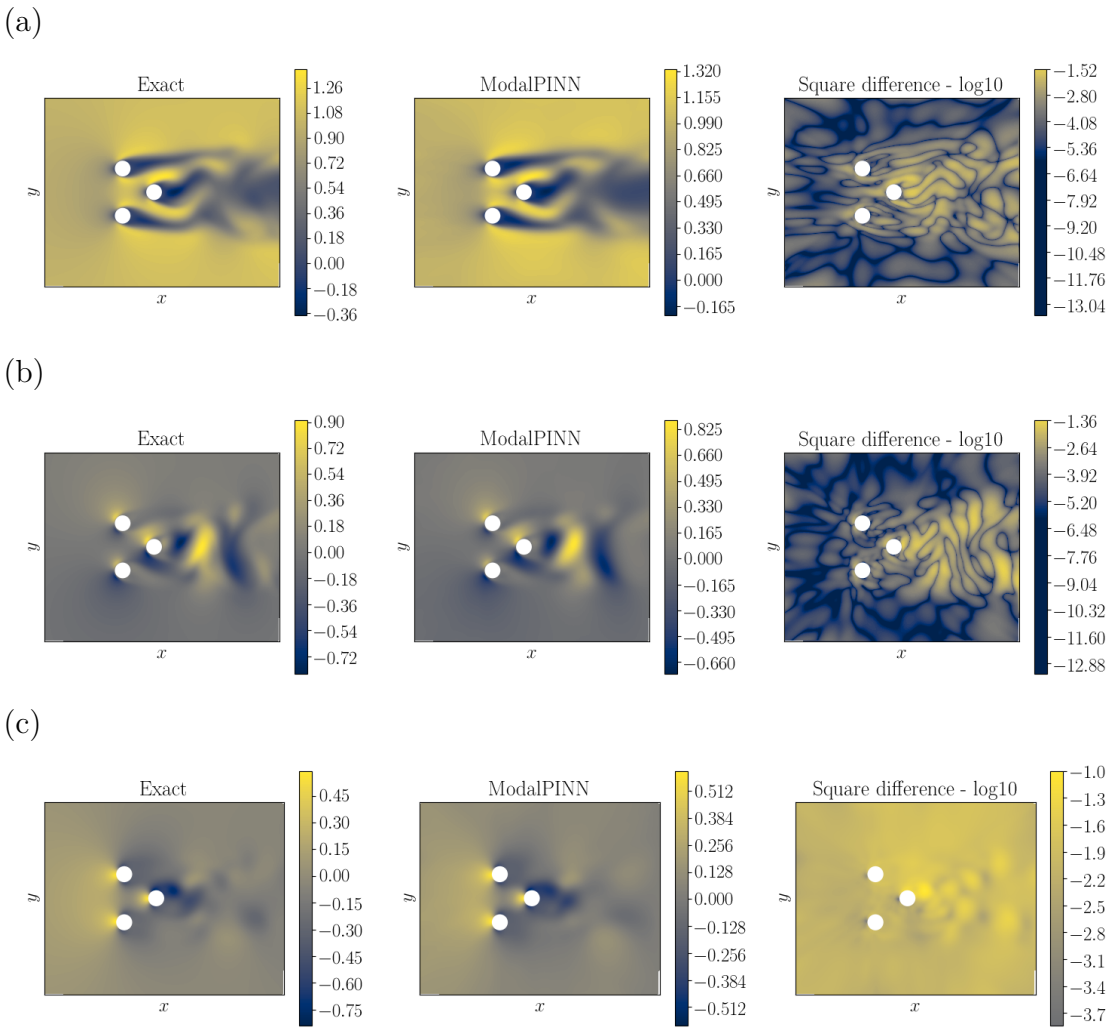


Figure A.12 Comparison at a given time of the velocity and pressure fields u, v and p (resp. a, b and c) with data from numerical simulations using dense measurements for training. The square difference is plotted in logarithmic scale.

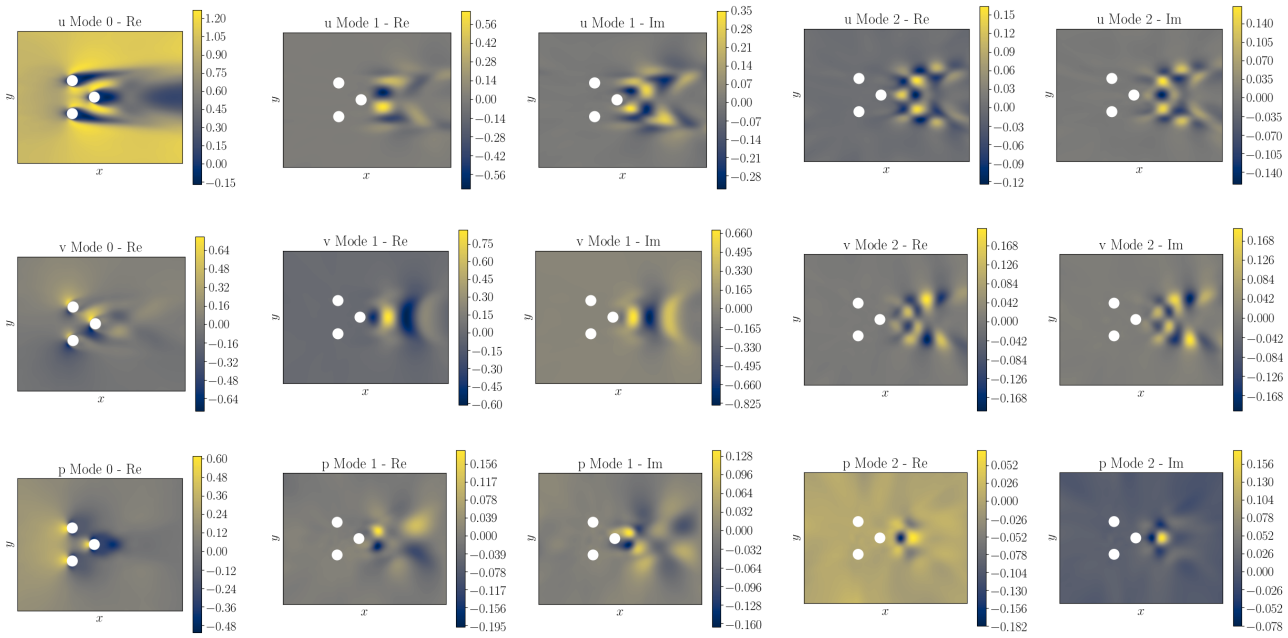


Figure A.13 Mode shapes retrieved by ModalPINN during the training on dense measurements as recalled at table A.3

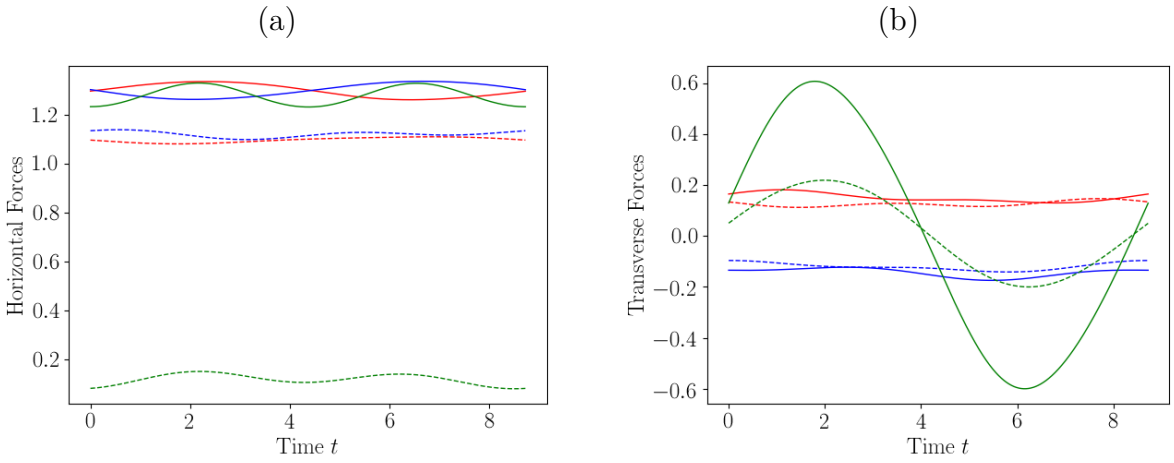


Figure A.14 Estimation of the unsteady drag C_x (a) and lift C_y (b) coefficients with the ModalPINN (dashed lines) using PIV-like measurements and comparison with numerical simulations (solid lines). Cylinder 5 (top left) is in red, cylinder 6 (bottom left) in blue and cylinder 7 (middle right) in green.

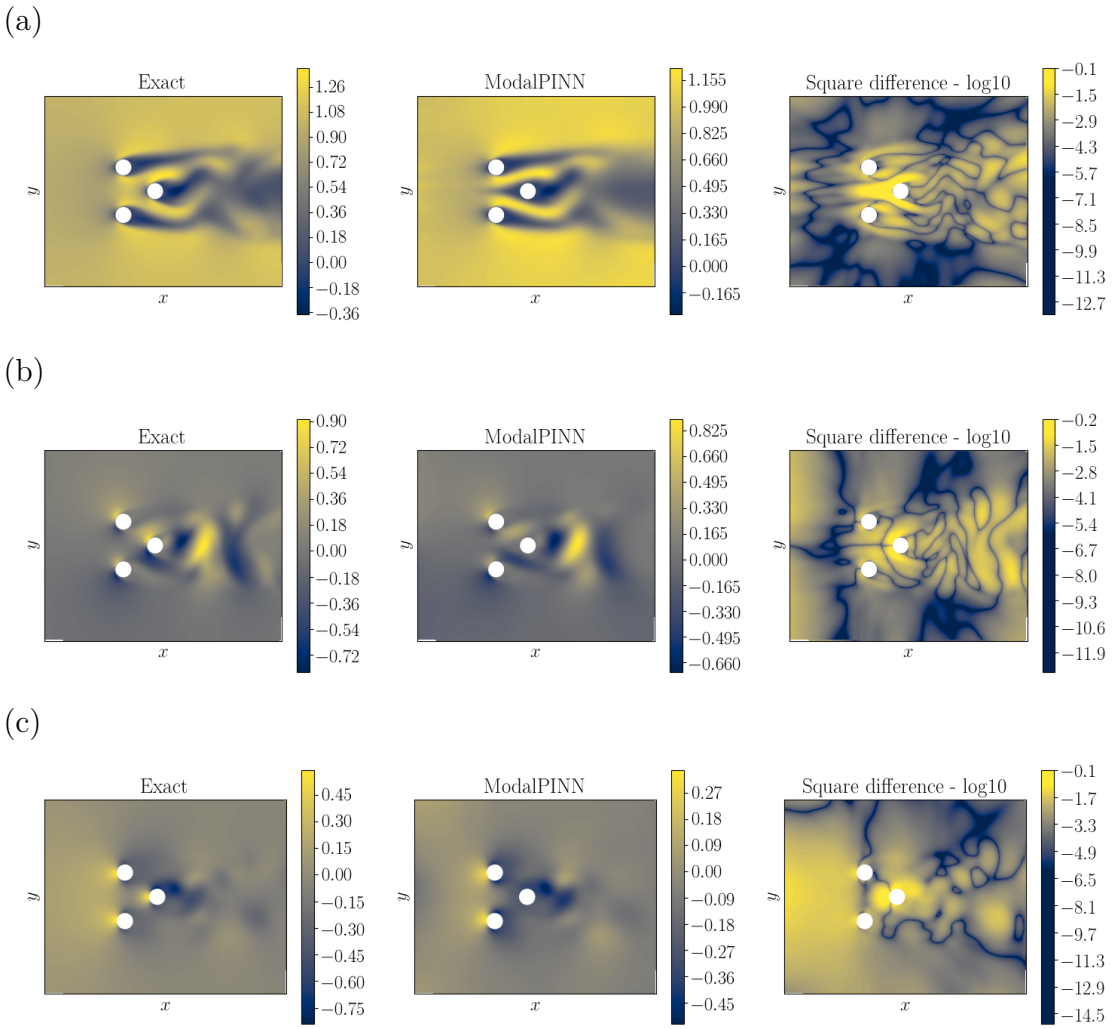


Figure A.15 Comparison at a given time of the velocity and pressure fields with data from numerical simulations using PIV-like measurements for training. The square difference is plotted in logarithmic scale.

Table A.5 Summary of a run with PIV-like measurements and pressure probe on cylinders for the flow reconstruction around 3 cylinders

Parameter	Value
Data	PIV-like u, v + Pressure probe
Pressure Mes.	Only on cylinders borders
Time	9h
N_{int}	1×10^5
N_{mes}	5×10^4
NN size	$[2, 75, 75, 3]$

Error	Value	Error on Forces	Relative	Absolute
BC	6.1×10^{-12}	C_{x5}	1.0×10^{-2}	1.3×10^{-2}
Total (train.)	6.2×10^{-3}	C_{x6}	1.6×10^{-2}	2.1×10^{-2}
Equations (train.)	3.7×10^{-3}	C_{x7}	2.3×10^{-2}	2.9×10^{-2}
Mes. (train.)	2.8×10^{-3}	C_{y5}	3.6×10^{-2}	5.5×10^{-3}
Mes u, v (Valid.)	4.1×10^{-3}	C_{y6}	3.5×10^{-2}	5.1×10^{-3}
Mes p (valid.)	6.6×10^{-4}	C_{y7}	1.7×10^{-1}	7.1×10^{-2}

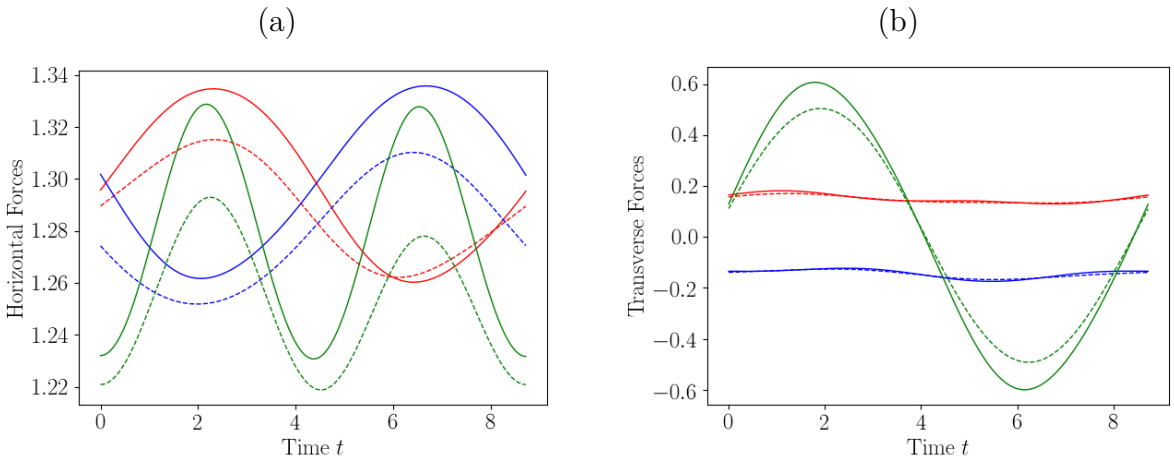


Figure A.16 Estimation of the unsteady drag C_x (a) and lift C_y (b) coefficients with the ModalPINN (dashed lines) using PIV-like measurements and pressure probe on cylinders compared with numerical simulations (solid lines). Cylinder 5 (top left) is in red, cylinder 6 (bottom left) in blue and cylinder 7 (middle right) in green.

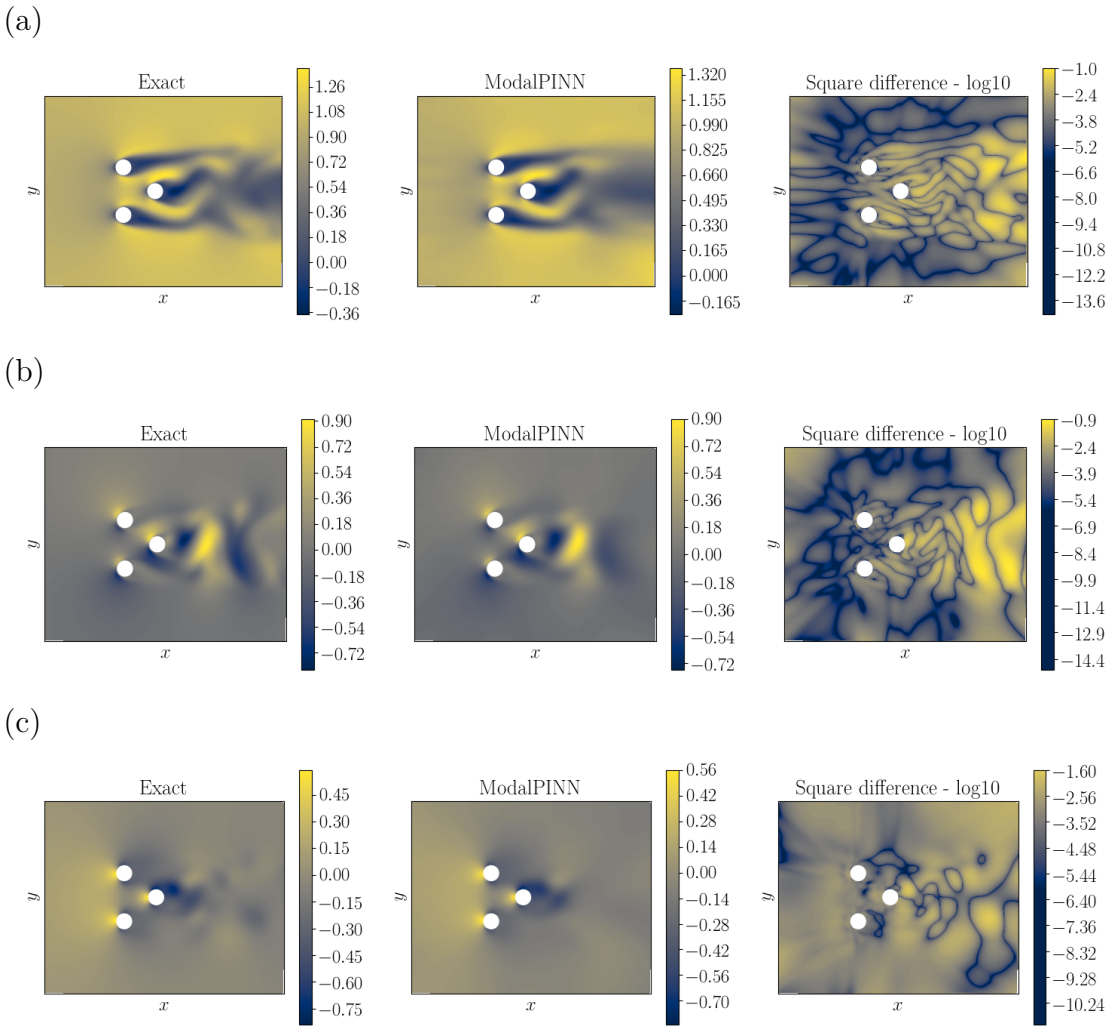


Figure A.17 Comparison at a given time of the velocity and pressure fields with data from numerical simulations using PIV-like measurements and pressure probe on cylinders for training. The square difference is plotted in logarithmic scale.

for each component described in table A.5 except for $C_{y,7}$ where this ratio is about 0.14. Consequently, without the information on velocity gradients, the relative difference between predicted forces and exact data would be stuck at values around 2×10^{-1}. The absence of the non-physical $u \approx 0$ area upstream of cylinder 7 is also noticeable when comparing figures A.17a and A.15a.

A direct test for robustness consists in feeding the neural network with artificially noised data, as discussed in sub-section 4.5.4. Here we use the same set-up as in the previous paragraph with velocity data in the PIV-like distribution and pressure data around cylinder borders. Velocities are artificially noised using a Gaussian distribution with a zero average and a standard deviation $\sigma = 0.2$. In order to get an idea of the level of imperfection, the complete set of u and v at a given time step from simulations data with $\sigma = 0.2$ is plotted in figure A.20. All the other parameters are kept the same (see table A.6) and one result is presented in figures A.18 and A.19. Between the training with noise and the one without (tables A.6 and A.5 respectively), we can observe that the precision of predicted forces is very similar and that the validation loss on u, v and p are of the same order of magnitude. However the training loss is significantly higher at the end of the training in the case of noise. This can be explained by the loss on fitting data that is of order 1 whereas the loss on equations is nearly two orders of magnitude lower. We can here conclude in a similar way than in sub-section 4.5.4: when $\mathcal{L}_{eq}$ and $\mathcal{L}_m$ are not compatible due to the presence of noise, we observe that only the PDE residuals drive the NN learning.

Although a more detailed study on this problem would be of interest, it seems that overall ModalPINNs show some signs of robustness in the context of imperfect data. Some questions still arise for other types of noise or for higher noise level. A possibility could be to adapt the function used to compute the distance: in our case an L^2 squared norm is used in $\mathcal{L}_m$ when performing the average between fitting and predicted data $(q_{DNN} - q_m)^2$. However other solutions that take into account an estimation of the amplitude of the noise could help reduce $\mathcal{L}_m$ when the predicted value q_{DNN} is within a given (confidence) interval around the noisy fitting value q_m. An illustration of this idea is proposed in figure A.21.

Table A.6 Summary of a run with noised PIV-like measurements and noiseless pressure probe on cylinders for the flow reconstruction around 3 cylinders

Parameter	Value
Data	Noisy PIV-like u, v + Noiseless Pressure probe
Noise level σ	0.2
Pressure Mes.	Only on cylinders borders
Time	9h
N_{int}	1×10^5
N_{mes}	5×10^4
NN size	$[2, 75, 75, 3]$

Error	Value	Error on Forces	Relative	Absolute
BC	7.8×10^{-12}	C_{x5}	1.3×10^{-2}	1.6×10^{-2}
Total (train.)	1.0×10^{-1}	C_{x6}	9.8×10^{-3}	1.3×10^{-2}
Equations (train.)	2.4×10^{-3}	C_{x7}	2.9×10^{-2}	3.7×10^{-2}
Mes. (train.)	9.9×10^{-2}	C_{y5}	4.4×10^{-2}	6.6×10^{-3}
Mes u, v (Valid.)	6.1×10^{-3}	C_{y6}	3.9×10^{-2}	5.6×10^{-3}
Mes p (valid.)	1.1×10^{-3}	C_{y7}	1.7×10^{-1}	7.0×10^{-2}

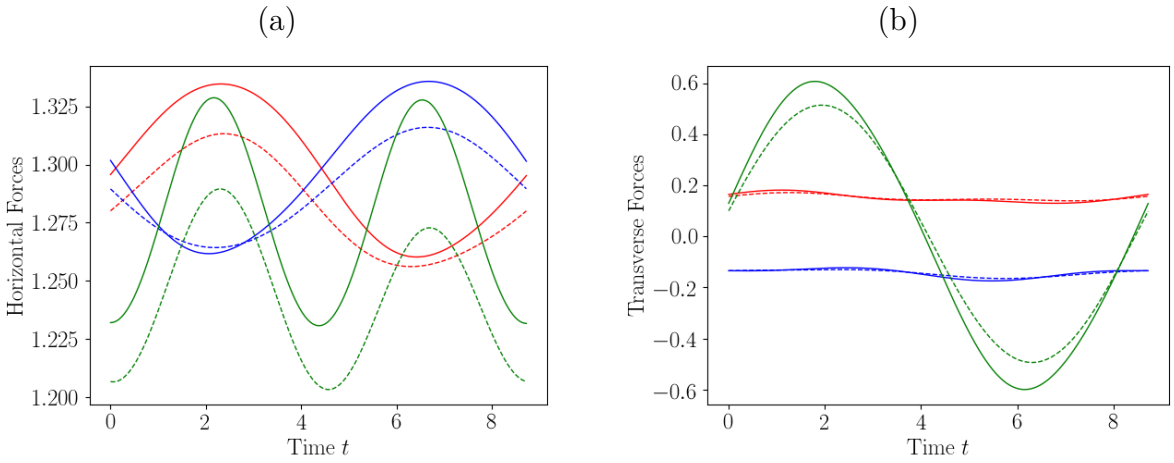


Figure A.18 Estimation of the unsteady drag C_x (a) and lift C_y (b) coefficients with the ModalPINN (dashed lines) using noised PIV-like measurements and noiseless pressure probe on cylinders compared with numerical simulations (solid lines). Cylinder 5 (top left) is in red, cylinder 6 (bottom left) in blue and cylinder 7 (middle right) in green.

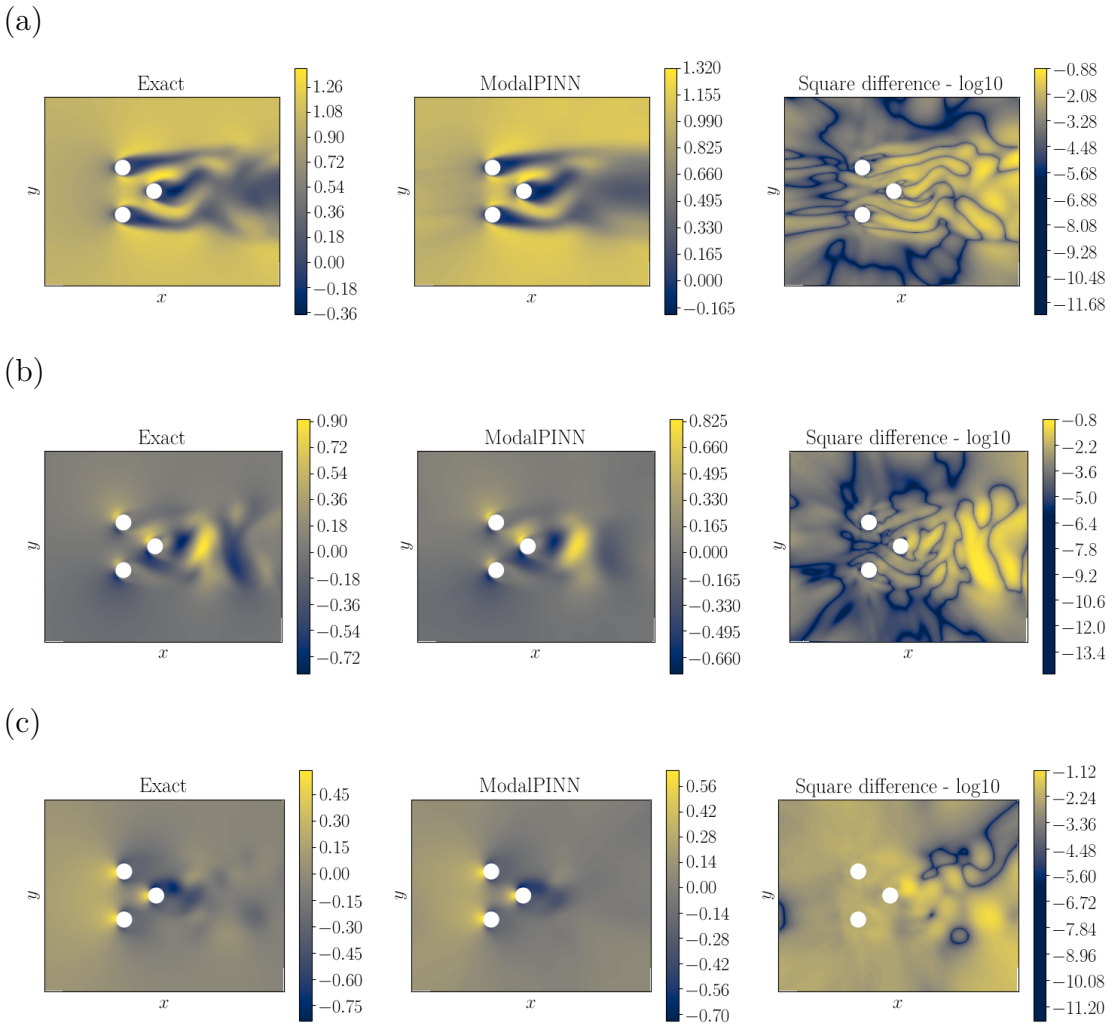


Figure A.19 Comparison at a given time of the velocity and pressure fields with data from numerical simulations using noised PIV-like measurements and noiseless pressure probe on cylinders for training. The square difference is plotted in logarithmic scale.

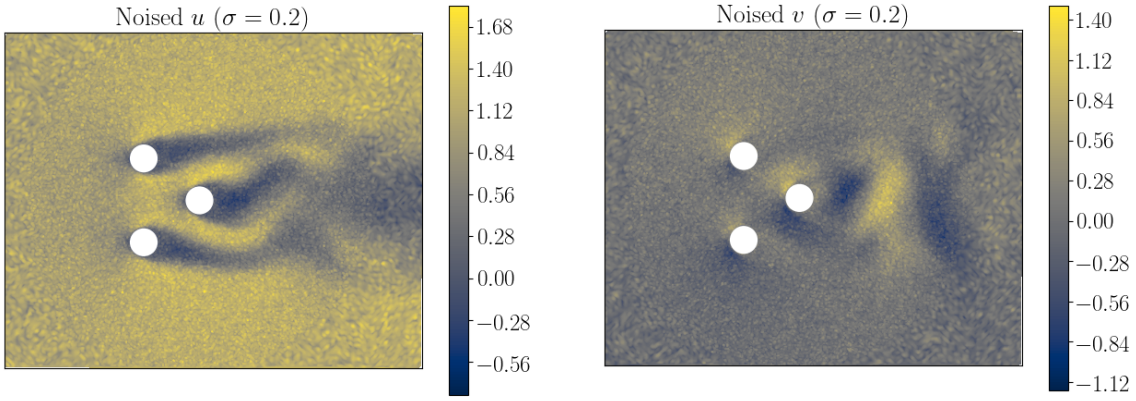


Figure A.20 Illustration of the noise level $\sigma = 0.2$ with velocity fields before cutting only the area for the PIV-like training.

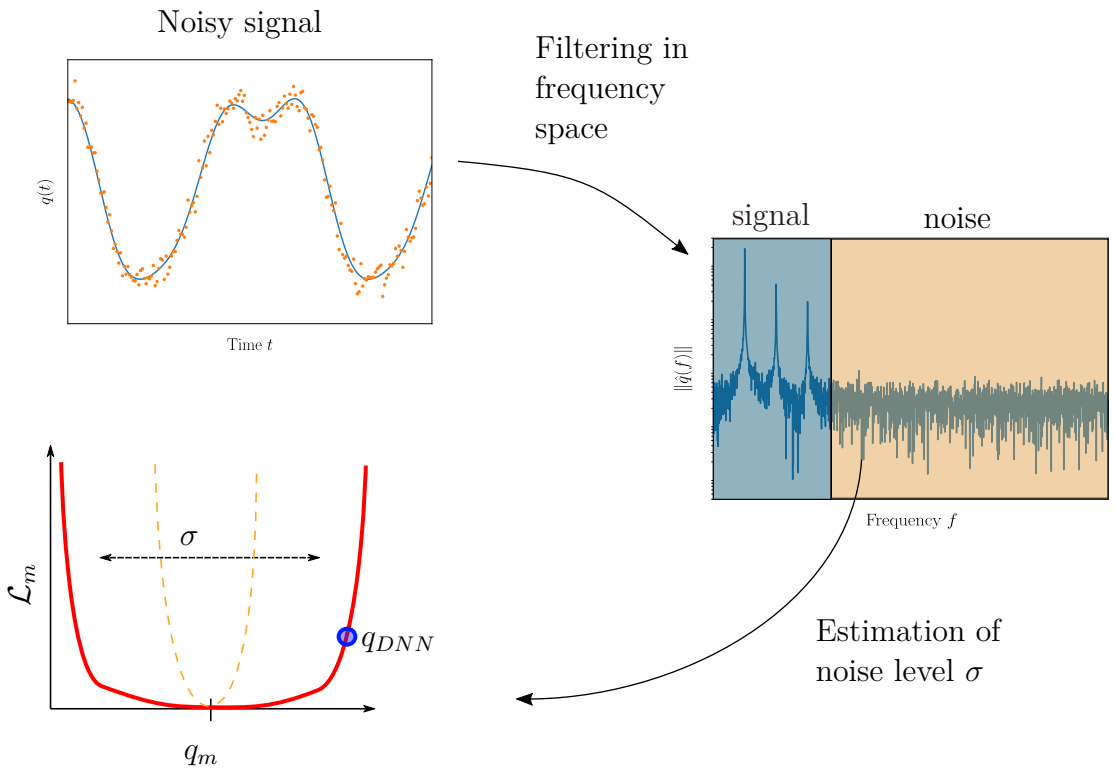


Figure A.21 Ideas about modification in the quadratic penalization of fitting errors. An estimation of noise level could be an input to use an adapted distance function with a flat area of the order of σ (red solid line) instead of the quadratic difference $(q_{DNN} - q_m)^2$ (orange dashed line).