Computers and Mathematics with Applications 143 (2023) 383–396

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

An extended physics informed neural network for preliminary analysis of parametric optimal control problems

Nicola Demo [a], Maria Strazzullo [b], Gianluigi Rozza [a,*]

[a] Mathematics Area, mathLab, SISSA, Via Bonomea, 265, Trieste, 34136, Italy
[b] DISMA, Politecnico di Torino, C.so Duca degli Abruzzi, 24, Torino, 10129, Italy

ARTICLE INFO

Keywords:
Physics-informed neural network
Deep learning
Optimal control problem
Parametric partial differential equations

ABSTRACT

In this work we propose an application of physics informed supervised learning strategies to parametric partial differential equations. Indeed, even if the latter are indisputably useful in many research fields, they can be computationally expensive most of all in a real-time and many-query setting. Thus, our main goal is to provide a physics informed learning paradigm to simulate parametrized phenomena in a small amount of time. The physics information will be exploited in many ways, in the loss function (standard physics informed neural networks), as an augmented input (extra feature employment) and as a guideline to build an effective structure for the neural network (physics informed architecture). These three aspects, combined together, will lead to a faster training phase and to a more accurate parametric prediction. The methodology has been tested for several equations and also in an optimal control framework.

1. Introduction

Machine Learning represents a growing impact research topic, widespread in several fields of applications, see for example [8,21,25,45]. This massive improvement and employment of such a technique is related to the growing disposal of available data and computing resources. Yet, the analysis of a complex system needs a huge quantity of data in order to be reliable and meaningful. However, collecting data to train can be unbearable for many reasons. Indeed, the costs of the collection can be expensive and data usually feature scattered information. This lack of knowledge might give non-robust results that can be inconsistent with respect to the known behaviour of a physical phenomenon. In this context, in the last few years, the Physics Informed Neural Networks (PINNs) have been conceived and developed in order to tackle this issue. The main idea behind PINNs is to add some physical information to the Neural Network (NN) in order to reach more reliable and complete predictions. The term physics usually translates to a mathematical model, such as, in our case, partial differential equations (PDEs). For an overview on the topic, the interested reader might refer to [40]. The promising results of this seminal paper, paved the way to many applications and extensions, see, e.g., this far-from-exhaustive list [5,17–19,28,29,33,41,47], where PINNs and its variations have been applied to many fields. This work focuses on the need of reliably predict physical phenomena in a parametric setting for complex systems. Motivated and inspired by [27] and the latest works [9,24], we want to find a PINN-based strategy to deal with equations where a parameter might change the physical features or the properties of the model itself. In many industrial and scientific contexts, there is a need of solving parametric PDEs (PDE($\boldsymbol{\mu}$)s) for many values of a parameter $\boldsymbol{\mu}$. Indeed, we have in mind the need for fast simulations in *real-time* and *many-query* applications, where $\boldsymbol{\mu}$-dependent solutions of a PDE($\boldsymbol{\mu}$) are required in a small amount of time. Thus, we propose to adapt the PINN paradigm to a parametric setting. As already mentioned, the PINNs are a valuable tool to predict a natural phenomenon according to a mathematical model, not relying on collected data information. However, the model equations, in several fields, are not sufficient to completely describe and understand a natural phenomenon. Is there a way to change the system at hand in order to achieve a *desired configuration*, which can represent expected behaviour extrapolated from scattered *in situ* data collection or a convenient profile to reach? Towards this goal, PDE($\boldsymbol{\mu}$)s constrained optimal control problems (OCP($\boldsymbol{\mu}$)s) can be employed in order to steer the problem solution towards a given configuration. Furthermore, we are interested in those applications where both the equation and the desired profile are parameter dependent. In this parametric setting, OCP ($\boldsymbol{\mu}$) s have been successfully employed in fluid dynamics, see e.g. [6,7,32,35], and in many scientific applications,

* Corresponding author.
E-mail addresses: nicola.demo@sissa.it (N. Demo), maria.strazzullo@polito.it (M. Strazzullo), gianluigi.rozza@sissa.it (G. Rozza).

https://doi.org/10.1016/j.camwa.2023.05.004

Available online 31 May 2023

such as biomedical [2,22,43,48] and environmental ones [37,38,42,43]. The main goal of the previous works was the application of model order reduction [12,36] to provide a low dimensional function space to solve the parametric instances in a faster way with respect to standard mesh-based approximation techniques such as Finite Elements, Finite Volumes, Spectral Methods, see for example [39]. In this work, we want to exploit PINN as a complete machine learning-based way to predict OCP($\boldsymbol{\mu}$)s solutions. It is clear that more complex problems are related to a possibly time consuming training phase. To accelerate this procedure, some ruses have been conceived, it is the case of activation functions [15,16] or Prior Dictionary PINNs [34]. The strategy we propose is based on exploiting additional known information, the *features*, in the input of the NN: this will allow to converge faster to a loss minimum not paying in the net prediction accuracy. Furthermore, we propose a different interpretation of the PINN paradigm, where the physics information is not only used to force the system to reach the expected model behaviour, but also to change the structure of the NN one is dealing with. The main novelty of this work, to the best of our knowledge, is the numerical investigation of tailored parametric PINN strategy for OCP($\boldsymbol{\mu}$)s. The application of PINN to optimal control problems has been recently investigated in [31]. However, differently from [31], we focus on the benefits of exploiting physics informed structures and extra features as a general strategy to deal with system of multiple equations, most of all in the parametric setting. Moreover, we combine the physics informed structures with extra features as an augmented input to accelerate the training procedure. We stress that parametric PDEs have been successfully tackled in [9,24,27,46]. However, in this contribution, we propose a different approach that totally complies with the original PINNs formulation of [40] with very encouraging results. This aspect is very important in terms of coding and applicability since the proposed approaches can be easily implemented for any PINN-code.

The work is outlined as follows: in Section 2, we will introduce the PINN structure in a parametric setting for one dimensional output and multi-dimensional outputs. In the same Section we will also present the main idea behind the employment of the extra features to accelerate the training phase. Section 3 represents a first step of the employment of PINNs for multi-variable output in the context of OCP($\boldsymbol{\mu}$)s, for Poisson and Stokes equations. Finally, conclusions follow in Section 4. Moreover, we present some additional results in Appendix A, where we validate the proposed methodology on forward one-dimensional problems such as Poisson equation [1] and Burgers equation [40]. The numerical tests have been tackled both in a parametric and non-parametric formulation.

2. Methodology

This Section focuses on the various methodologies we relied on to tackle several PDE-based problems. First of all, we will describe PINNs-based methods in Section 2.1. As already mentioned in Section 1, one of the main novelty of the work is their extension to parametrized OCP($\boldsymbol{\mu}$)s. Section 2.2 concerns the employment of *extra-features* that will help the convergence of the neural network, reducing the computational time needed to train the model. Finally, we will also present how to exploit the physics information to change the architecture of the used PINN to have more accurate results in Section 2.3. This will be helpful when dealing with systems of multiple equations. This technique, already successfully employed for multi–fidelity problems [3,11,30], allowed us to reach better results with respect to standard PINN approach.

2.1. PINNs

Let us assume to be provided by a PDE($\boldsymbol{\mu}$) endowed with its boundary conditions of the form $G : \mathbb{Y} \to \mathbb{Q}^*$ that reads

$$G(w(\mathbf{x}, \boldsymbol{\mu})) = f(\mathbf{x}, \boldsymbol{\mu}), \tag{1}$$

where $w \doteq w(\mathbf{x}, \boldsymbol{\mu}) \in \mathbb{Y}$ is the unknown physical quantity we are taking into consideration in a domain $\Omega \subset \mathbb{R}^d$, while $f(\boldsymbol{\mu}) \in \mathbb{Q}^*$ is an external forcing term, with $\mathbb{Y}$ and $\mathbb{Q}$ two suited Hilbert spaces. The system changes with respect to a parameter $\boldsymbol{\mu} \in \mathcal{D} \subset \mathbb{R}^D$, of dimension $D \geq 1$. The parameter $\boldsymbol{\mu}$ represents physical and/or geometrical features of the problem at hand. In this work, we will focus on physical parametrization only. With this notation, we are referring to a broad class of PDE($\boldsymbol{\mu}$)s: from time-dependent nonlinear problems, to linear steady ones and to coupled systems (as OCP($\boldsymbol{\mu}$)s). Indeed, when dealing with time-dependent problems, we are implicitly considering a time evolution of w in the time interval $[0, T]$ in the expression (1). Thus, depending on the context, the variable $\mathbf{x}$ can be interpret as $\mathbf{x} \doteq x$ for steady problems and as $\mathbf{x} \doteq (x, t)$ in the time-dependent case, where $x \in \mathbb{R}^d$ represents the spatial coordinate vector while $t \in [0, T]$ is the time variable. Along this contribution we will also consider PDEs that do not depend on a parameter, in that case $G : \mathbb{Y} \to \mathbb{Q}^*$ reads

$$G(w(\mathbf{x})) = f(\mathbf{x}). \tag{2}$$

The same argument for time-dependent problems applies to the non-parametric context of (2). We aim at finding data-driven solutions to (1) or (2). For the sake of brevity we will describe the methodology only in the parametric case. Indeed, this more general formulation, will easily adapt to the context of non-parametric PDEs. Building on the strategy firstly conceived in [40], we propose a PINN able to capture the value of the solution w for several parametric instances. To this end, we define the residual of (1) as:

$$r(w(\mathbf{x}, \boldsymbol{\mu})) \doteq G(w(\mathbf{x}, \boldsymbol{\mu})) - f(\mathbf{x}, \boldsymbol{\mu}). \tag{3}$$

Now let us imagine to have been provided of some boundary values, say N_b points $\{\mathbf{x}_i^b\}_{i=1}^{N_b}$, of N_p internal points $\{\mathbf{x}_i^p\}_{i=1}^{N_p}$ and of $N_{\boldsymbol{\mu}}$ points in the parameter space, i.e. $\{\boldsymbol{\mu}_i\}_{i=1}^{N_{\boldsymbol{\mu}}}$. We want to exploit the data information given by the residual in a NN $\hat{w}$ that takes as input some domain coordinates and a parametric instance and gives back a prediction of the physical phenomenon. In this sense, the network will be *physics informed* and will exploit the residual value in order to converge towards a physical meaningful solution w. The differential structure of the residual can be derived by applying automatic differentiation based on the chain rule for differentiating compositions of functions [3,40]. In order to embed the physical behaviour in the NN at hand, we define the following mean squared error loss:

$$MSE^{\boldsymbol{\mu}} \doteq \frac{1}{N_{\boldsymbol{\mu}}} \sum_{i=1}^{N_{\boldsymbol{\mu}}} (MSE_b^{\boldsymbol{\mu}_i} + MSE_p^{\boldsymbol{\mu}_i}), \tag{4}$$

where

- given the boundary values of the solution w_i^b for $i = 1, \dots, N_b$ at the collocation points $\{\mathbf{x}_i^b\}_{i=1}^{N_b}$ the *boundary loss* is

$$MSE_b^{\boldsymbol{\mu}_i} \doteq \frac{1}{N_b} \sum_{k=1}^{N_b} |\hat{w}(\mathbf{x}_k^b, \boldsymbol{\mu}_i) - w(\boldsymbol{\mu}_i)_k^b|^2; \tag{5}$$

- while the *residual loss* is:

$$MSE_p^{\boldsymbol{\mu}_i} \doteq \frac{1}{N_p} \sum_{k=1}^{N_p} |r(\hat{w}(\mathbf{x}_k, \boldsymbol{\mu}_i))|^2. \tag{6}$$

In this context, $(\mathbf{x}_k^b, w(\boldsymbol{\mu}_i)_k^b)$ for $k = 1, \dots, N_b$ are the boundary training data that might be $\boldsymbol{\mu}-$ dependent, while $\{\mathbf{x}_i^p\}_{i=1}^{N_p}$ are the collocation point for the $\boldsymbol{\mu}-$ dependent residual. Namely:

- the first set of inputs enforces the boundary conditions in space and, possibly, in time,
- the residual evaluated in the internal points is informative about the structure of the physical model one is interested in.

To the best of our knowledge, it is the first time that the parameters are considered as inputs of the model, together with the points of the domain in space and time. Indeed, in seminal paper about PINN, see e.g. [40] the parameters of the problem were fixed or the PINN was used in parameter identification, also in an inverse problem Bayesian setting [47]. Instead, in this contribution, inspired by [27], we want to take a first step in the employment of PINNs in many-query contexts, where several parameter evaluations are required to deeper analyse the physical phenomenon at hand. The main advantage of this parametric approach relies in the versatility of the considered model, where many configurations can be learnt from a finite parametric training data $\{\boldsymbol{\mu}_i\}_{i=1}^{N_\mu}$. The PINN can respond, in a total non-intrusive way, to the need of reliable solutions of PDE($\boldsymbol{\mu}$)s for several values of $\boldsymbol{\mu}$ in *real-time* and *many-query* contexts, where many evaluations of the equation (1) are required in a small amount of time. Indeed, paying a reasonable amount of time training the net, it is possible to fast predict faithfully a new parametric instance.

Remark 1 *(The non-parametric case).* It is clear that, for the non parametric case, we are dealing with the structure first presented in [40]. For the sake of clarity, we report the structure we used in this simplified setting. First of all, the residual is defined as $MSE \doteq MSE_b + MSE_p$, where

$$MSE_b \doteq \frac{1}{N_b}\sum_{k=1}^{N_b} |\hat{w}(\mathbf{x}_k^b) - w_k^b|^2 \quad \text{and} \quad MSE_p \doteq \frac{1}{N_p}\sum_{k=1}^{N_p} |r(\hat{w}(\mathbf{x}_k))|^2, \tag{7}$$

where, naturally, from (2), the non-parametric residual is given by

$$r(w(\mathbf{x})) \doteq G(w(\mathbf{x})) - f(\mathbf{x}). \tag{8}$$

Namely, in this case, it is sufficient to be provided of N_b boundary points $\{\mathbf{x}_i^b\}_{i=1}^{N_b}$ with the related PDE boundary values w_k^b together with N_p internal points to evaluate (8), since the NN $\hat{w}$ takes input only the domain coordinates $\mathbf{x}$.

2.2. Extra features

By definition, NNs are a composition of nonlinear functions. Intuitively this implies that the greater the number of functions involved within the composition is, the more complex the output of the network will be. Previous works [16,18] have shown that increasing the depth of the network allows to learn PDEs with a high nonlinearity, indeed. On the other hand, increasing the number of hidden layers — usually equal to the number of activation functions — makes the training phase even more expensive from the computational viewpoint and may introduce other numerical problems, e.g. vanishing gradient.

One possible way to contain the dimension of the model and preserving at the same time the output complexity can be the extension of the features we use as input of the network. In a typical PINN framework, the features are the dimensions of the input domain, i.e. the spatial and temporal coordinates, and, in case of parametric problems, also the parameters. For example in an unsteady parametric problem the input is $\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_d & \mu_1 & \dots & \mu_D & t \end{bmatrix}$. We can exploit our a priori knowledge about the physical equations to extend the information we use as input, evaluating one (or more) input variable using one (or more) kernel function.[1] We define these functions as $\{k_i(\mathbf{x})\}_{i=1}^{N_f}$ and we will refer to them as *extra features* from here on. The new input is then

$$\mathbf{x}_{\text{extra}} = \begin{bmatrix} \mathbf{x} & k_1(\mathbf{x}) & \dots & k_{N_f}(\mathbf{x}) \end{bmatrix} \tag{9}$$

of dimension $d + D + 1 + N_f$. Of course, also functions defined into subspaces of the input space are valid as well. For a concrete example, let us imagine to dealing with a generic problem that shows a quadratic dependency by the first spatial dimension and an exponential behaviour in time: a reasonable input should be $\mathbf{x}_{\text{extra}} = \begin{bmatrix} x_1 & \dots & x_d & t & x_1^2 & e^t \end{bmatrix}$. Selecting a proper set of extra features, the model will learn an easier correlation between input and output, resulting in a faster training and a smaller architecture. On the practical side, the extra features are problem dependent and they have to be tuned to avoid loss of performances. We present here only few guidelines to select good features, postponing the deeper discussion regarding their effectiveness to Section A.1. In case of problems with external forces, the analytical function representing such force could be a valid option for an extra feature. In this way, the NN uses the new feature as a sort of initial guess of its learning procedure. Similarly, simple functions that satisfy the boundary (or initial) conditions could lead the training procedure to been improved. We remark that we can moreover adding learnable parameters within the extra features, allowing for adaptive extra features that, as we will discuss in Section A.1, improve the training step both in terms of accuracy and computational needed.

2.3. PINNs for multiple equations problems

In this Section we are going to extend the idea of using a NN to approximate the physical behaviour given by system of multiple equations. Namely, we are facing a problem of the form

$$\mathcal{G}(w(\mathbf{x}, \boldsymbol{\mu})) = \mathcal{F}(\mathbf{x}, \boldsymbol{\mu}), \tag{10}$$

where the solution is made of n variables $(w_1(\mathbf{x}, \boldsymbol{\mu}), \dots, w_n(\mathbf{x}, \boldsymbol{\mu}))$, while the left and the right hand side are defined as:

$$\mathcal{G}(w(\mathbf{x}, \boldsymbol{\mu})) \doteq \begin{bmatrix} G_1(w(\mathbf{x}, \boldsymbol{\mu})) \\ \vdots \\ G_n(w(\mathbf{x}, \boldsymbol{\mu})) \end{bmatrix} \quad \text{and} \quad \mathcal{F}(\mathbf{x}, \boldsymbol{\mu}) \doteq \begin{bmatrix} f_1(\mathbf{x}, \boldsymbol{\mu}) \\ \vdots \\ f_n(\mathbf{x}, \boldsymbol{\mu}) \end{bmatrix}. \tag{11}$$

It is clear that, in order to solve the problem (10) in a PINN fashion, a new definition of the mean squared error loss is needed, to take into account all the involved equations and boundary conditions. First of all we define the residual of the j-th equation as

$$r_j(w(\mathbf{x}, \boldsymbol{\mu})) \doteq G_j(w(\mathbf{x}, \boldsymbol{\mu})) - f_j((\mathbf{x}, \boldsymbol{\mu})). \tag{12}$$

We recall that problem (10) is related to a set of boundary conditions. Thus, let $J = \{1, \dots, m\}$ be the indices of the number of boundary conditions applied while let us define $I = \{1, \dots, n\}$. Now, given $\boldsymbol{\mu} \in \mathcal{D}$ and the related sets of boundary points $\{\mathbf{x}_i^b\}_{i=1}^{N_b^l}$ and boundary conditions $\{w(\boldsymbol{\mu})_i^b\}_{i=1}^{N_b^l}$ for $l \in J$ we can define:

- the new boundary loss as

$$MSE_b^{\boldsymbol{\mu}_i} \doteq \sum_{l \in J} \frac{1}{N_b^l}\sum_{k=1}^{N_b^l} |\hat{w}(\mathbf{x}_k^b, \boldsymbol{\mu}_i) - w(\boldsymbol{\mu}_i)_k^b|^2; \tag{13}$$

- the new residual loss as

$$MSE_p^{\boldsymbol{\mu}_i} \doteq \sum_{j \in I} \frac{1}{N_p}\sum_{k=1}^{N_p} |r_j(\hat{w}(\mathbf{x}_k, \boldsymbol{\mu}_i))|^2, \tag{14}$$

where $\hat{w}$ is a NN capable to approximate the global solution of the system of equations considered. At the end, the global loss can be exactly defined as we already did in (4). The problem can be solved through the PINN approach already presented in Section 2.1 and the extra features arguments also applies and easily adapts in this context. Furthermore, this structure can be easily specified to the non-parametric case as presented in Remark 1. In the next Section we are going to question ourselves on the possibility of building a *physics informed architecture* (PI-Arch) to better deal with problems with more than one variable. In this framework, the model does not only influence the loss function, but also the structure of the NN itself.

[1] which should be differentiable.

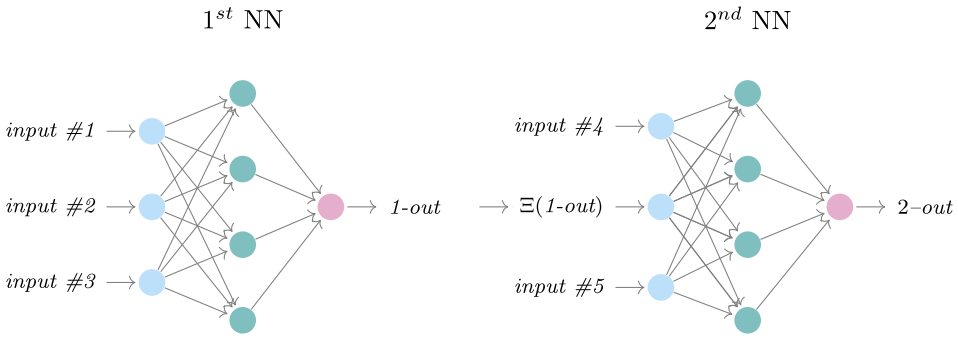

Fig. 1. PI-Arch example.

2.4. The PI-Arch strategy for multiple equations problems

This Section focuses on the idea of exploiting the PINN paradigm in a more complete way. Indeed, once provided of some physical information about the problem at hand, it can be exploited not only in the loss, but also in the structure of the employed NN. We are going to extend the idea of using a NN to approximate the physical behaviour given by system of multiple equations. The idea builds on multi–fidelity approaches as used in [3,11,30] and it was guided by the numerical tests we performed on OCP($\boldsymbol{\mu}$)s. The comment on the numerical results is postponed in Section 3.1. We now focus on a general description of the adopted strategy. We recall that we are working in the framework of multiple variables described by system (10). Namely we are dealing with n equations and m boundary conditions. First of all, the physics is taken into account in the loss definition, indeed the global loss is defined as in (4), where $MSE_p^{\boldsymbol{\mu}}$ and $MSE_b^{\boldsymbol{\mu}}$ are exactly defined as in (14) and (13). Nonetheless, we stress that the same arguments of Remark 1 apply also in this more complicated context with analogous definitions. The idea is to change the architecture of the NN $\hat{w}$ exploiting the structure of the model behind the phenomenon we are studying. Namely, we would like to adapt the structure in order to be more accurate in the prediction of the outputs. Let us suppose to have built a tailored (combination of) NN (NNs) to approximate the PDE($\boldsymbol{\mu}$) solution. Let us consider $H_1 \subset I$. Namely, a first NN takes the inputs and predicts a part of the variables $\{\hat{w}_{h_1}(\mathbf{x},\mu)\}_{h_1 \in H_1}$. This latter set of predicted solution will be called the *one-level output* (1-out). An improper subset of the one-level output together with the initial input (or a part of it) will be the input for a new NN which will predict the *two-level output* (2-out) $\{\hat{w}_{h_2}(\mathbf{x},\mu)\}_{h_2 \in H_2}$, where $H_2 \subset I \setminus H_1$. These output variables combined with the one-level output (or a part of it) and the initial input (or part of it) will lead to the *three-level output* (3-out) for the set of indices $H_3 \subset I \setminus \{H_1 \cup H_2\}$, and so on. The process is repeated until the $k-$level, where the final output will be $\{\hat{w}_{h_k}(\mathbf{x},\mu)\}_{h_k \in H_k}$, with

$$\bigcup_k H_k = I.$$

Now let us image that from the equations, directly or indirectly, we can derive a sort of relation, say Ξ, between the outputs. Namely, we are able to connect 1-out to the 2-out through

$$\Xi(\text{1-out}) \approx \text{2-out}. \tag{15}$$

Thus, this information pave the way to two different strategies:

- if confident enough, replace the NN connecting the 1-out to the 2-out with Ξ(1-out);
- build a NN capable to approximate Ξ(1-out).

A schematic representation of the proposed architecture is given in Fig. 1. Such approach aims to divide the NN in hand into a combination of different (typically smaller) NNs, thus isolating the relations between the outputs of the model — that we know a priori from the physics equations — that otherwise have to be learned during the training. In this way, we exploit the physical information not only for the loss definition, but also in the structure of the model. For this reason, we refer to this strategy in the rest of the manuscript as *physics informed architecture* (PI-Arch).

3. PINNs for optimal control problems

In this Section we are going to test our methodology in a more complex setting, where a coupled system of equations is solved. We will deal with OCP($\boldsymbol{\mu}$)s: this mathematical tool is used to change the classical behaviour of a physical variable to reach a given desired configuration. Optimal Control framework is thus an input-output system that, given an observable, tries to drive the solution towards this data thanks to an external variable called *control* [4,10,13,26,44]. OCP($\boldsymbol{\mu}$)s have been employed in many applications in several scientific and industrial fields, the interested reader might refer to [23] for an overview. In the next Section we will introduce the problem setting and we will show some numerical results based on PI-Arch application.

3.1. Problem formulation

Here, we here provide the continuous formulation for general OCP($\boldsymbol{\mu}$)s. Indeed, the following setting is suited to nonlinear problem as well as time dependent ones. However, we will focus on steady problems: the choice is guided by the numerical results we will discuss in the Section. After introducing the problem at hand, we rely on the Lagrangian formalism [10,13,14] to solve the constrained minimization problem. Let $\Omega \subset \mathbb{R}^d$, with $d = 2,3$ be the spatial domain where the studied physical phenomena is occurring. We assume to be provided by *state equation* $G : \mathbb{Y} \to \mathbb{Q}^*$ of the form

$$G(y(\mathbf{x},\boldsymbol{\mu})) = f(\mathbf{x},\boldsymbol{\mu}). \tag{16}$$

We call $y(\mathbf{x},\boldsymbol{\mu})$ the *state variable*, i.e. the physical variable we want to steer towards a desired profile $y_{\text{d}} \doteq y_{\text{d}}(\boldsymbol{\mu}) \in \mathbb{Y}_{\text{obs}} \supseteq \mathbb{Y}$. Also in this case, $f(\mathbf{x},\boldsymbol{\mu}) \in \mathbb{Q}^*$ is an external forcing term. We now denote with the notation $\mathcal{L}(\cdot,\cdot)$ the space of the continuous linear functions between two function spaces. As already specified, the aim of the problem, i.e. to steer the solution behaviour towards a desired profile, is reached through a *control variable* $u(\mathbf{x},\boldsymbol{\mu}) \in \mathbb{U}$. Here, $\mathbb{U}$ is another Hilbert space. The control variable acts in $\Omega_u \subset \Omega$, the *control domain*. In this work, we will assume $\Omega_u = \Omega$, namely, we will test the proposed methodology for *distributed* OCP($\boldsymbol{\mu}$)s. In order to change the classical solution behaviour, we need to define a *controlled system*, i.e. $\mathcal{E} : \mathbb{Y} \times \mathbb{U} \to \mathbb{Q}^*$ such that:

$$\mathcal{E}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu})) \doteq G(y(\mathbf{x},\boldsymbol{\mu})) - C(u(\mathbf{x},\boldsymbol{\mu})) - f(\mathbf{x},\boldsymbol{\mu}) = 0. \tag{17}$$

The operator $C \in \mathcal{L}(\mathbb{U},\mathbb{Q}^*)$ changes the original system to reach the desired variable y_{d}. Theoretically, this goal is represented by the solution of the following constrained minimization problem: given a $\boldsymbol{\mu} \in D$ and an observation $y_{\text{d}} \in \mathbb{Y}_{\text{obs}}$, find the pair $(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu})) \in \mathbb{Y} \times \mathbb{U}$ which solves

$$\min_{y(\mathbf{x},\boldsymbol{\mu})\in\mathbb{Y},u(\mathbf{x},\boldsymbol{\mu})\in\mathbb{U}} J(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu})) \text{ such that } \mathcal{E}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))=0, \tag{18}$$

where $J:\mathbb{Y}\times\mathbb{U}\times\mathbb{Y}_{\mathrm{obs}}\to\mathbb{R}$ a *cost functional* of the form:

$$J(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});y_{\mathrm{d}}) \doteq \frac{1}{2}\|y(\mathbf{x},\boldsymbol{\mu})-y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu})\|^2_{\mathbb{Y}_{\mathrm{obs}}}+\frac{\alpha(\boldsymbol{\mu})}{2}\|u(\mathbf{x},\boldsymbol{\mu})\|^2_{\mathbb{U}}, \tag{19}$$

where $\alpha(\boldsymbol{\mu})\in(0,1]$ is a *penalization term*. The reader interested in the well-posedness of the model may refer to [13] for example. The optimal control problem can be solved through a Lagrangian argument [13,44], defining an arbitrary *adjoint variable* $z(\mathbf{x},\boldsymbol{\mu})$ in $\mathbb{Y}$. The use of this variable will allow us to solve the problem (18) in an unconstrained minimization fashion. In order to recover the notation we employed in the previous Sections, we define the *global variable* $w(\mathbf{x},\boldsymbol{\mu})\doteq(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}),z(\mathbf{x},\boldsymbol{\mu}))$. Then, we can define the following *Lagrangian functional*

$$\begin{aligned}\mathscr{L}(w(\mathbf{x},\boldsymbol{\mu});y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu})) \doteq J(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu});y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu}))\\ +\langle z(\mathbf{x},\boldsymbol{\mu}),\mathcal{E}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))\rangle_{\mathbb{Q}\mathbb{Q}^*},\end{aligned}$$

where $\langle\cdot,\cdot\rangle_{\mathbb{Q}\mathbb{Q}^*}$ is the duality pairing of the spaces $\mathbb{Q}$ and $\mathbb{Q}^*$ and we are assuming $\mathbb{Y}\subset\mathbb{Q}$. In this setting, it is well-known in literature [4,10,13,26,44] that the minimization problem (18) is equivalent to the solution $(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}),z(\mathbf{x},\boldsymbol{\mu}))\in\mathbb{Y}\times\mathbb{U}\times\mathbb{Y}$ of the following *optimality system*:

$$\begin{cases} D_y\mathscr{L}(w(\mathbf{x},\boldsymbol{\mu});y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu}))[\omega]=0 & \forall\omega\in\mathbb{Y},\\ D_u\mathscr{L}(w(\mathbf{x},\boldsymbol{\mu});y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu}))[\kappa]=0 & \forall\kappa\in\mathbb{U},\\ D_z\mathscr{L}(w(\mathbf{x},\boldsymbol{\mu});y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu}))[\zeta]=0 & \forall\zeta\in\mathbb{Y},\end{cases} \tag{20}$$

where $D_yL(w(\mathbf{x},\boldsymbol{\mu});y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu}))$ is the Fréchet differential of the Lagrangian functional with respect to the state variable. The same notation is used for the other variables. The optimality system (20) reads: given $\boldsymbol{\mu}\in\mathcal{D}$, find the optimal solution $(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}),z(\mathbf{x},\boldsymbol{\mu}))\in\mathbb{Y}\times\mathbb{U}\times\mathbb{Y}$ such that

$$\begin{cases} y(\mathbf{x},\boldsymbol{\mu})+D_y\mathcal{E}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))^*(z(\mathbf{x},\boldsymbol{\mu}))=y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu}),\\ \alpha(\boldsymbol{\mu})u(\mathbf{x},\boldsymbol{\mu})-C^*(z(\mathbf{x},\boldsymbol{\mu}))=0,\\ G(y(\mathbf{x},\boldsymbol{\mu}))-C(u(\mathbf{x},\boldsymbol{\mu}))=f(\mathbf{x},\boldsymbol{\mu}).\end{cases} \tag{21}$$

It is now clear that we are dealing with a system of the form introduced in Section 2.3 and can be written in the following compact form: given $\boldsymbol{\mu}\in\mathcal{D}$, find $w(\mathbf{x},\boldsymbol{\mu})\in\mathbb{Y}\times\mathbb{U}\times\mathbb{Y}$ such that

$$\mathcal{G}(w(\mathbf{x},\boldsymbol{\mu}))=\mathcal{F}(\mathbf{x},\boldsymbol{\mu}), \tag{22}$$

with

$$\mathcal{G}(w(\mathbf{x},\boldsymbol{\mu}))\doteq\begin{bmatrix} y+D_y\mathcal{E}(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))^*(z(\mathbf{x},\boldsymbol{\mu}))\\ \alpha(\boldsymbol{\mu})u(\mathbf{x},\boldsymbol{\mu})-C^*(z(\mathbf{x},\boldsymbol{\mu}))\\ G(y(\mathbf{x},\boldsymbol{\mu}))-C(u(\mathbf{x},\boldsymbol{\mu}))\end{bmatrix} \text{ and } \mathcal{F}(\mathbf{x},\boldsymbol{\mu})\doteq\begin{bmatrix} y_{\mathrm{d}(\mathbf{x},\boldsymbol{\mu})}\\ 0\\ f(\mathbf{x},\boldsymbol{\mu})\end{bmatrix}.$$

We want to remark that this formulation can be simplified in a non-parametrized setting, where the input of the PINN will only consist in the variable $\mathbf{x}$, see e.g., Remark 1. In the next Section we will present some numerical results on the application of a PI-Arch to an optimal control test problem in both parametrized and non-parametrized version.

3.2. Numerical results: Poisson problem

For this numerical test, we consider a steady OCP($\boldsymbol{\mu}$) in the physical domain $\Omega=[-1,1]\times[-1,1]$. The parametric minimization problem reads: given $\boldsymbol{\mu}\doteq(\mu_1,\mu_2)\in[0.5,3]\times[0.01,1]$ find $(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))\in H^1_0(\Omega)\times L^2(\Omega)$ such that

$$\min_{(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))}\frac{1}{2}\|y(\mathbf{x},\boldsymbol{\mu})-\mu_1\|^2_{L^2(\Omega)}+\frac{\mu_2}{2}\|u(\mathbf{x},\boldsymbol{\mu})\|^2_{L^2(\Omega)},$$

constrained to

$$\begin{cases} -\Delta y(\mathbf{x},\boldsymbol{\mu})=u(\mathbf{x},\boldsymbol{\mu}) & \text{in } \Omega,\\ y(\mathbf{x},\boldsymbol{\mu})=0 & \text{on } \partial\Omega.\end{cases} \tag{23}$$

Namely, we choose $y_{\mathrm{d}}(\mathbf{x},\boldsymbol{\mu})\equiv\mu_1$ all over the physical domain. Moreover, we also exploited a parametric penalization term. Thus, the minimization problem seeks a distributed control term $u(\mathbf{x},\boldsymbol{\mu})$ that is able to steer the solution towards a parametric instance μ_1. Applying the Lagrangian arguments of Section 3.1 to problem (23), we end up with the following optimality system once defined the adjoint variable $z(\mathbf{x},\boldsymbol{\mu})\in H^1_0(\Omega)$

$$\begin{cases} y(\mathbf{x},\boldsymbol{\mu})-\Delta z(\mathbf{x},\boldsymbol{\mu})=\mu_1 & \text{in } \Omega,\\ z(\mathbf{x},\boldsymbol{\mu})=0 & \text{on } \partial\Omega,\\ \mu_2u(\mathbf{x},\boldsymbol{\mu})=z(\mathbf{x},\boldsymbol{\mu}) & \text{in } \Omega,\\ -\Delta y(\mathbf{x},\boldsymbol{\mu})=u(\mathbf{x},\boldsymbol{\mu}) & \text{in } \Omega,\\ y(\mathbf{x},\boldsymbol{\mu})=0 & \text{on } \partial\Omega.\end{cases} \tag{24}$$

Thus, we want to tackle this problem with the employment of a PI-Arch, say once again $\imath\hat{w}$, which takes as input $(\mathbf{x},\boldsymbol{\mu})$ and gives as a result output the predicted values of the three involved variables $(y(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}),z(\mathbf{x},\boldsymbol{\mu}))$. Our first attempt was to use the formulation proposed in Section 2.3. Namely, a standard PINN was built. The prediction was based on a 3–Layers NN. The first two layers are made by 40 neurons, while the last hidden layer had only 20 neurons. We employed an ADAM optimizer with a Softplus activation function and LR = 0.002. The number of the collocation points are given by: $N_p=900$, $N_b=200$ and $N_{\boldsymbol{\mu}}=50$. In order to accelerate the training phase and to help the function in reaching the right boundary conditions, we employed the following extra feature:

$$k_1(x_0,x_1)\doteq(1-x_0^2)(1-x_1^2). \tag{25}$$

The plots in Fig. 4 depict the results after a 10000 epochs training. On the top row, from left to right, we observe the predicted control, adjoint and state variables for $\boldsymbol{\mu}=(3,1)$. The results in this case were quite as expected but, once evaluated the parameter $\boldsymbol{\mu}=(3,0.01)$ an issue came out, as represented in the bottom row of Fig. 4. Indeed, the standard PINN approach in this case was not able to recover the relation $\mu_2u(\mathbf{x},\boldsymbol{\mu})=z(\mathbf{x},\boldsymbol{\mu})$. Namely, the qualitative and the quantitative behaviour of the adjoint variable $z(\mathbf{x},\boldsymbol{\mu})$ is completely different with respect to the control variable $u(\mathbf{x},\boldsymbol{\mu})$. For this reason, we decided to exploit the optimality equation in order to build a PI-Arch aware of such a relation between the two variables. A schematic representation of the used structure is presented in Fig. 2.

Namely, a first NN approximates the 1-out given by the control and the state variables once provided the input $(\mathbf{x},\boldsymbol{\mu},k_1(x_0,x_1))$. Then, the control variable combined with the parameter μ_2 is used to define the adjoint variable in another NN as $\Xi(1\text{-out})=\mu_2u(\mathbf{x},\boldsymbol{\mu})$, to force a meaningful physical behaviour. Fig. 3 shows how the PI-Arch can be effective if compared to standard PINN strategies. Namely, imposing the physics directly in the structure of $\imath\hat{w}$ can be really helpful in increasing the accuracy of the prediction. Indeed, we are able not only to recover a good approximation for the case $\boldsymbol{\mu}=(3,1)$, but also for a smaller value of the penalization parameter, i.e. $\boldsymbol{\mu}=(3,0.01)$: the control and the adjoint variable are now fully coherent with the physics of the problem. To test the accuracy of the PI-Arch structure decided to compare the prediction of $\imath\hat{w}$ in the domain point (0, 0) to the FE solution in $\mu_1=1,2,3$ and $\mu_2=1,0.1,0.01$. The results are depicted in Fig. 3 and Fig. 5. Comparing the standard PINN model with the PI-Arch one, we highlight that the state variable prediction — for $\mu_2=0.01$ — is closer to the FE solution with the improved architecture. While, we observe a good prediction for the state variable for all the values of μ_2 (right plot), the quality of the approximation slightly worsen for the control variable when dealing with $\mu_2=0.01$. However, the results are quite satisfactory and promising. We remark that the PI-Arch is problem-dependent and we believe that small modifications of such a structure together with a longer training phase may lead to more reliable results. However, the topic of the right tuning of the $\imath\hat{w}$ parameters and structure go beyond the goals of this work, where we want to provide a non–intrusive tool capable to

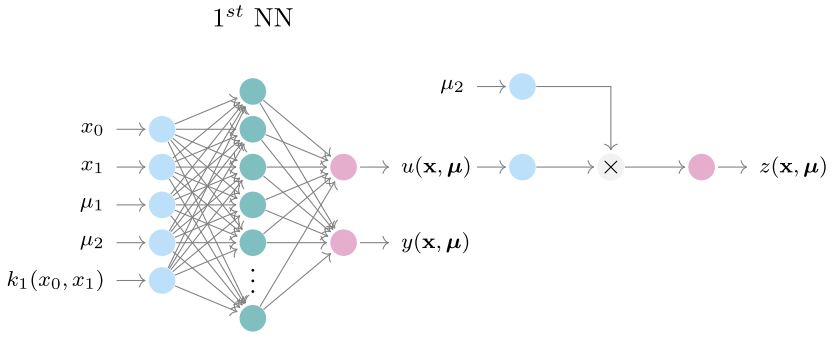

Fig. 2. PI-Arch used to solve problem (23).

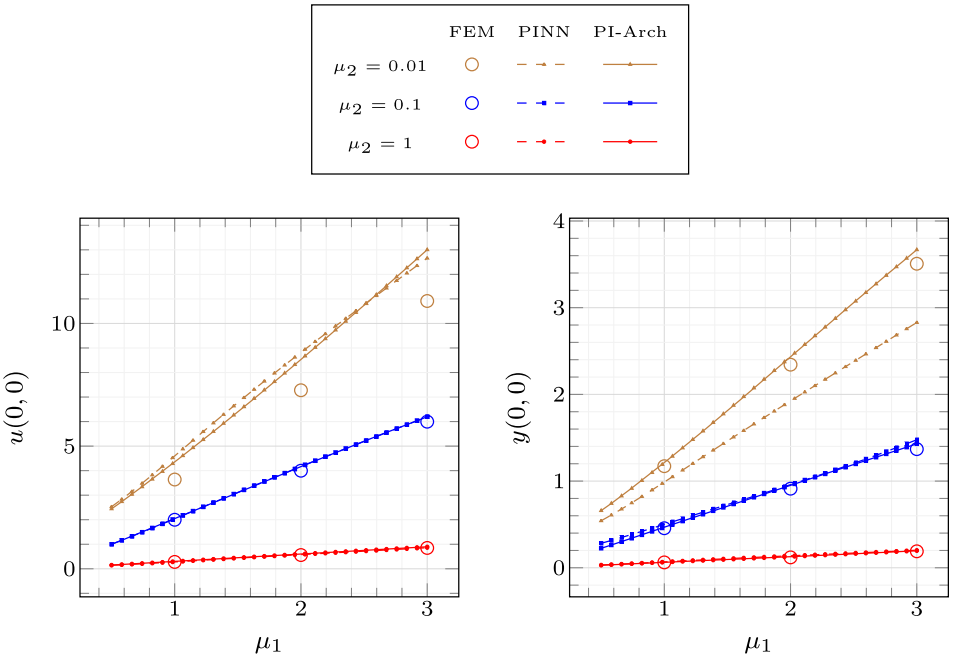

Fig. 3. Prediction for $\mu_2 = 1, 0.1, 0.01$ with respect to μ_1 in $(x_0, x_1) = (0, 0)$ compared to the FE approximation for $\mu_1 = 1, 2, 3$. *Left.* Control variable. *Right.* State variable.

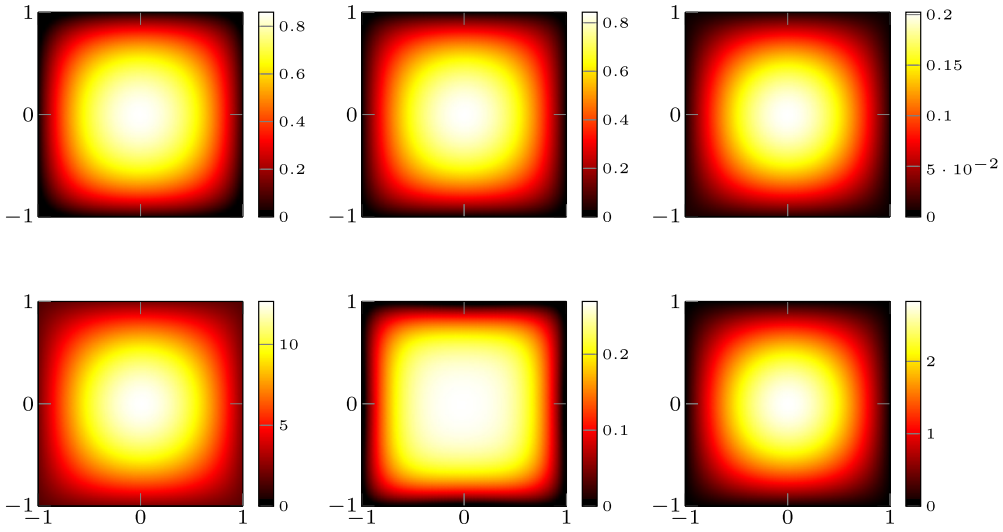

Fig. 4. Parametric Optimal Control Problem. *Top row*. PINN approximation for $\boldsymbol{\mu} = (3, 1)$. *Bottom row*. PINN approximation for $\boldsymbol{\mu} = (3, 0.01)$. *Left.* The control variable u. *Center.* The adjoint variable z. *Right.* The state variable y.

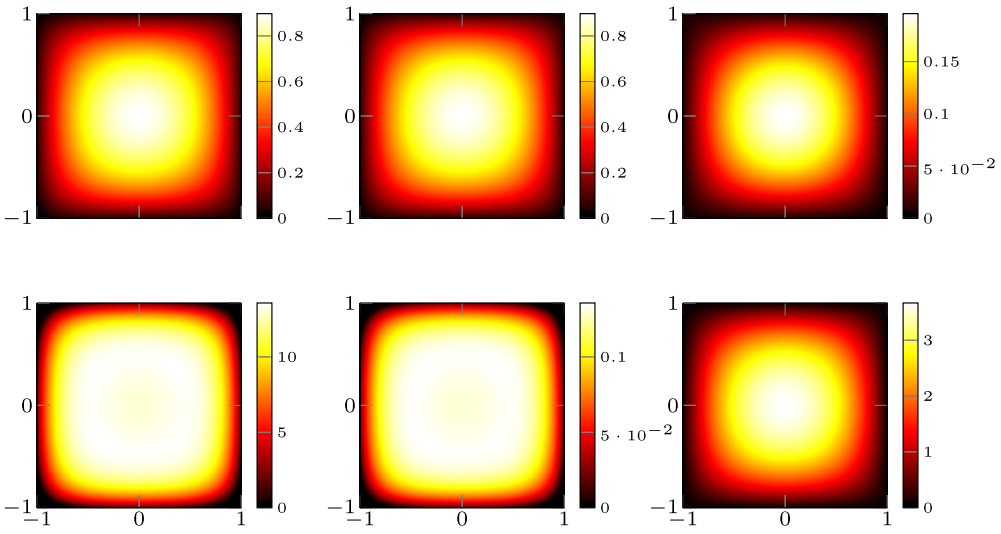

Fig. 5. Parametric Poisson Optimal Control Problem. *Top row*. PI-Arch approximation for $\boldsymbol{\mu} = (3, 1)$. *Bottom row*. PI-Arch approximation for $\boldsymbol{\mu} = (3, 0.01)$. *Left.* The control variable u. *Center.* The adjoint variable z. *Right.* The state variable y.

answer to parametric studies needs in a small amount of time without paying in accuracy.

3.3. Numerical results: Stokes problem

In the last experiment, we deal with a steady OCP($\boldsymbol{\mu}$) governed by Stokes equations. The physical domain is $\Omega = [0, 1] \times [0, 2]$. We call $\Gamma_{\mathrm{D}} \doteq \{0\} \times [0, 2]$ and $\Gamma_{\mathrm{D}} \doteq \{1\} \times [0, 2]$. Here, we consider a single parameter related to the forcing term: $\boldsymbol{\mu} \in [0.5, 1.5]$. The penalization parameter is fixed and we set $\alpha = 0.008$ Indeed, we want to find $(v(\mathbf{x}, \boldsymbol{\mu}), p(\mathbf{x}, \boldsymbol{\mu}), u(\mathbf{x}, \boldsymbol{\mu})) \in [H_0^1(\Omega]^2 \times L^2(\Omega) \times [L^2(\Omega)]^2$ such that

$$\min_{(v(\mathbf{x},\boldsymbol{\mu}),u(\mathbf{x},\boldsymbol{\mu}))} \frac{1}{2}\|v(\mathbf{x}, \boldsymbol{\mu}) - x_2\|^2_{L^2(\Omega)} + \frac{\alpha}{2}\|u(\mathbf{x}, \boldsymbol{\mu})\|^2_{L^2(\Omega)},$$

constrained to (26)

$$\begin{cases} -0.1\Delta v(\mathbf{x}, \boldsymbol{\mu}) + \nabla p(\mathbf{x}, \boldsymbol{\mu}) = f(\mathbf{x}, \mu_1) + u(\mathbf{x}, \boldsymbol{\mu}) & \text{in } \Omega, \\ \nabla \cdot v(\mathbf{x}, \boldsymbol{\mu}) = 0 & \text{in } \Omega, \\ v(\mathbf{x}, \boldsymbol{\mu})_1 = x_2 \text{ and } v(\mathbf{x}, \boldsymbol{\mu})_2 = 0 & \text{on } \Gamma_{\mathrm{D}}, \\ -p(\mathbf{x}, \boldsymbol{\mu})\mathbf{n}_1 + 0.1\dfrac{\partial v(\mathbf{x}, \boldsymbol{\mu})_1}{\partial \mathbf{n}_1} \text{ and } v(\mathbf{x}, \boldsymbol{\mu})_2 = 0 & \text{on } \Gamma_{\mathrm{N}}. \end{cases}$$

The subscripts $_1$ and $_2$, indicate the first and the second component of a vector field, respectively. The formulation is totally similar to the one proposed in Section 3.2 we can apply standard Lagrangian arguments and the minimization problem translates in the solution of the following system (Fig 5):

$$\begin{cases} -0.1\Delta z(\mathbf{x}, \boldsymbol{\mu}) + \nabla r(\mathbf{x}, \boldsymbol{\mu}) = \begin{bmatrix} x_2 - v(\mathbf{x}, \boldsymbol{\mu})_1 \\ 0 \end{bmatrix} & \text{in } \Omega, \\ z(\mathbf{x}, \boldsymbol{\mu})_1 = z(\mathbf{x}, \boldsymbol{\mu})_2 = 0 & \text{on } \Gamma_{\mathrm{D}}, \\ \nabla \cdot z(\mathbf{x}, \boldsymbol{\mu}) = 0 & \text{in } \Omega, \\ -r(\mathbf{x}, \boldsymbol{\mu})\mathbf{n}_1 + 0.1\dfrac{\partial z(\mathbf{x}, \boldsymbol{\mu})_1}{\partial \mathbf{n}_1} \text{ and } z(\mathbf{x}, \boldsymbol{\mu})_2 = 0 & \text{on } \Gamma_{\mathrm{N}}. \\ \alpha u(\mathbf{x}, \boldsymbol{\mu}) = z(\mathbf{x}, \boldsymbol{\mu}) & \text{in } \Omega, \\ -0.1\Delta v(\mathbf{x}, \boldsymbol{\mu}) + \nabla p(\mathbf{x}, \boldsymbol{\mu}) = f(\mathbf{x}, \mu_1) + u(\mathbf{x}, \boldsymbol{\mu}) & \text{in } \Omega, \\ \nabla \cdot v(\mathbf{x}, \boldsymbol{\mu}) = 0 & \text{in } \Omega, \\ v(\mathbf{x}, \boldsymbol{\mu})_1 = x_2 \text{ and } v(\mathbf{x}, \boldsymbol{\mu})_2 = 0 & \text{on } \Gamma_{\mathrm{D}}, \\ -p(\mathbf{x}, \boldsymbol{\mu})\mathbf{n}_1 + 0.1\dfrac{\partial v(\mathbf{x}, \boldsymbol{\mu})_1}{\partial \mathbf{n}_1} \text{ and } v(\mathbf{x}, \boldsymbol{\mu})_2 = 0 & \text{on } \Gamma_{\mathrm{N}}, \end{cases} \tag{27}$$

where $(z(\mathbf{x}, \boldsymbol{\mu}), r(\mathbf{x}, \boldsymbol{\mu})) \in \times[H_0^1(\Omega]^2 \times L^2(\Omega)$ is the adjoint variable for this problem. Also in this case, using the PI-Arch structure $\hat{w}$ led to advantages in terms of accuracy in the prediction of the five variables, given $(\mathbf{x}, \boldsymbol{\mu})$ as inputs. The results with respect to the standard PINN proposed in Section 2.3 are depicted in Fig. 6. We build a standard PINN with 4–Layers NN where all the hidden layers are made of 40 neurons. For the prediction, we exploited an ADAM optimizer with a Softplus activation function and LR $= 0.003$. The collocation points are collected using a latin hypercube strategy, for a total of $N_p = 400$, $N_b = 1800$ and $N_{\boldsymbol{\mu}} = 10$.

No extra-features have been applied in this case, since x_2, i.e. the Dirichlet boundary condition for the first component of the state velocity, is an input itself, already. The plots in Figure depict the results after a 10000 epochs training. On the top row, from left to right, we observe the predicted control, adjoint and state variables for $\boldsymbol{\mu} = 1$. In this case, an issue came out, as represented in the bottom row of Fig. 4. Indeed, the standard PINN approach was not able to recover the relation $\alpha u(\mathbf{x}, \boldsymbol{\mu}) = z(\mathbf{x}, \boldsymbol{\mu})$. Namely, the qualitative and the quantitative behaviour of the adjoint variable $z(\mathbf{x}, \boldsymbol{\mu})$ is completely different with respect to the control variable $u(\mathbf{x}, \boldsymbol{\mu})$.

To avoid this issue, we explicit the optimality equation in the PI-Arch structure, as already done in the previous test case. The strategy is totally analogous: a first NN (with the same parameter of the standard PINN described above) predicts the 1-out that predicts the control, the state variables, and the adjoint variable $r(\mathbf{x}, \boldsymbol{\mu})$ from the input $(\mathbf{x}, \boldsymbol{\mu})$. Then, the adjoint variable $z(\mathbf{x}, \boldsymbol{\mu})$ is the result of another NN defined as $\Xi(1\text{-out}) = \alpha u(\mathbf{x}, \boldsymbol{\mu})$: we are imposing the physical constraint directly. Fig. 6 shows how the PI-Arch can be effective if compared to standard PINN strategies. Namely, imposing the physics directly in the structure of $\hat{w}$ can be really helpful in increasing the accuracy of the prediction. Indeed, we are able to recover a good approximation for the case $\boldsymbol{\mu} = 1$, even using the penalization factor $\alpha = 0.008$, which implies different order of magnitude in the predicted unknowns. The control and the adjoint variable are now fully coherent with the physics of the problem, contrary to the standard architecture. The results are quite satisfactory and promising, even if it must be said that more consolidated frameworks — e.g. finite elements — are surely able to achieve better accuracy. We remark that the PI-Arch is problem-dependent and we believe that small modifications of such a structure together with a longer training phase may lead to more reliable results. However, the topic of the right tuning of the $\hat{w}$ parameters and structure go beyond the goals of this work, where we want to provide a non–intrusive tool capable to answer to parametric studies needs in a small amount of time without paying in accuracy.

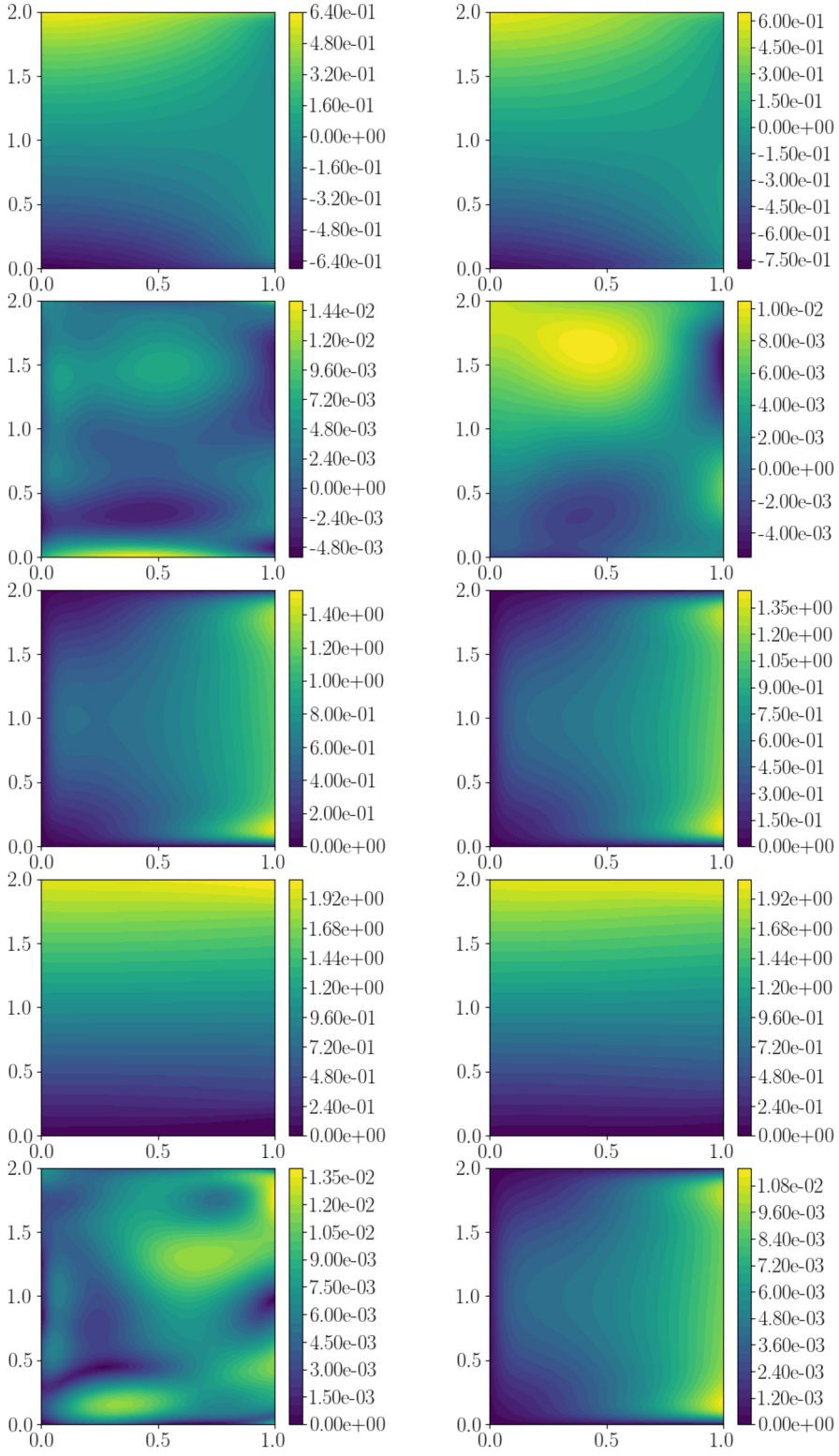

Fig. 6. Parametric Stokes Optimal Control Problem. The field p, r, u, v and z are shown for $\boldsymbol{\mu} = 1$, from the top to the bottom. *Left column*. Standard FNN approximation. *Right column*. PI-Arch approximation.

4. Conclusions

We propose a PINN extension to parametric settings in a real-time and many-query context. The main goal of this contribution was to adapt the PINN paradigm to the need for fast simulations for several parametric instances in a small amount of time. Indeed, PINN is a valuable tool in this framework. After the training phase, several $\boldsymbol{\mu}$-dependent outputs can be computed in a rapid sequence. However, another aspect to consider is how long this training phase could be. To overcome a possibly unbearable PINN training, we used the extra features: namely, an augmented input drives the NN towards the loss minimum in a smaller number of epochs. Furthermore, we propose the PI-Arch technique to build tailored NNs to deal with complex problems. The methodologies have been tested both on forward problems and on more complex systems of equations, such as OCP($\boldsymbol{\mu}$)s. Besides the promising results we have shown, many open questions and points remain of interest.

First of all, we would like to treat more complex problems in the OCP($\boldsymbol{\mu}$)s framework. Indeed, optimal control is a mathematical tool suited in many data-based contexts, such as inverse problems and data assimilation, to bridge the gap between data and mathematical modelling. Thus, the combination with PINNs seems a natural choice to integrate some data information in the problem at hand without spoiling its physical meaning. The presented results are just a first step that moves in the direction of cases of actual interest.

Another improvement concerns the choice of the extra features and the PI-Arch structure. In this work, they are assumed to be known. We are aware that this is not the case for many real applications. Thus, many natural questions might arise. Among them: is there a way to appropriately choose the features? Is there a way to learn them? What is the best way to connect the PINN architecture with this knowledge?

For sure, the presented methodology enhanced with the proposed improvements would pave the way to the use of PINN in very complex settings, ranging from several application fields in science and industry.

Data availability

Data will be made available on request.

Acknowledgements

This work was partially supported by European Union Funding for Research and Innovation — Horizon 2020 Program — in the framework of European Research Council Executive Agency: H2020 ERC CoG 2015 AROMA-CFD project 681447 “Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics” P.I. Gianluigi Rozza. The work was also supported by INdAM-GNCS: Istituto Nazionale di Alta Matematica —- Gruppo Nazionale di Calcolo Scientifico. We thank the support from PRIN project NA-FROM-PDEs (MIUR).

Appendix A. Some additional results

We are going to discuss the advantages of using extra features to accelerate the training phase reaching the best approximation results for the physical phenomenon one is dealing with, not necessarily in the constrained optimization framework. The paradigm of PINN is to add the physical information you have on the system to reach solutions that are related to PDEs models. The employment of extra features totally complies with the PINN fundamentals: add previous knowledge to your system to reach a more reliable solution and, most of all, in a smaller training time. Finally, we will test the PINN methodology in a parametrized setting, too.

A.1. Poisson problem

The first problem we are going to tackle is a Poisson problem. The test case is a slightly modified version of the problem presented in [1]. We consider the two-dimensional steady case, given by the following continuous model:

$$\begin{cases} \Delta w(\mathbf{x}) = f(\mathbf{x}) & \text{in } \Omega, \\ w(\mathbf{x}) = 0 & \text{on } \partial\Omega, \end{cases} \tag{A.1}$$

where Ω is the unit square $[0,1]\times[0,1]$, a point in Ω is $\mathbf{x} = (x_0, x_1)$ and the forcing term is defined as

$$f(\mathbf{x}) \doteq \sin(\pi x_0)\sin(\pi x_1).$$

The Poisson problem (A.1) has the following analytical solution:

$$w(\mathbf{x}) = -\frac{\sin(\pi x_0)\sin(\pi x_1)}{2\pi^2}. \tag{A.2}$$

After having defined the residual as we did in (8), we choose $N_p = 100$ and $N_b = 40$ following an equispaced rule. We deal with the problem through a 2–Layer NN of 10 neurons for each layer. As optimizer, we exploited ADAM [20] with a learning rate (LR) fixed at 0.003 and a Softplus activation function. We show three numerical examples to state how the features can be useful in order to accelerate the training procedure and reach faster convergence rates.

- (*No extra features*) First of all, Fig. A.7 shows a comparison between the analytical solution of problem (A.1) and the PINN solution. They coincide. The accuracy of the PINN is assured also by the right plot representing the absolute value of the pointwise difference between the solutions, say the *pointwise error*, that reaches the value of 10^{-3} after 1000 epochs.
- (*With extra features*) For the same NN architecture, we tested the effect of the extra features in this context. Guided by the knowledge on the forcing term, we enforced the input with the extra feature given by

  $$k_1(x_0, x_1) \doteq \sin(x_0\pi)\sin(x_1\pi). \tag{A.3}$$

  From Fig. A.8, it is clear that we are gaining an order of magnitude in the relative error with respect to standard PINN approach (we refer to the right plot).
- (*Learnable extra features*) This phenomenon is highlighted in the third example, where we exploit a *learnable feature*. Namely, we enforce the following information on the system:

  $$k_1(x_0, x_1) \doteq \beta_0 \sin(\alpha_0 x_0 + \gamma_0)\beta_1 \sin(\alpha_1 x_1 + \gamma_1), \tag{A.4}$$

  where $\alpha_0, \alpha_1, \beta_0, \beta_1, \gamma_1$ and γ_2 are real values to be determined and learned by the NN.

The advantages of the latter approach are remarkable since, after 1000 epochs, we reach an absolute pointwise error of 10^{-9} (Fig. A.9.), exploiting a fixed LR $= 0.008$. In this test case, we keep the same number of boundary and internal points as before ($N_b = 40$ and $N_p = 100$) but we remove all the hidden layers in the model, converging to a simple architecture constituted by only input and output layers. This of course has been possible since the learnable feature we provided can be fit to the analytical solution of the problem, without the needed of using nonlinear activation functions. However, the usage of such features does not affect the time of the training phase which still remain in the order of minutes. From the three numerical examples we reported, a remarkable improvement is noticed in terms of relative error for a fixed number of epochs. This is also confirmed by Fig. A.10, where the loss values of the different approaches are depicted. First of all, the learnable extra feature is capable to reach machine precision loss around 1500 epochs, where the standard PINN loss is stuck around 10^{-3}. The decay of the loss is helped even by the extra feature (A.3), reaching the threshold of 10^{-3} for only 500 epochs. These experiments heuristically prove how important is to add all the information you have on your system in order to have a reliable NN prediction in a small amount of training time.

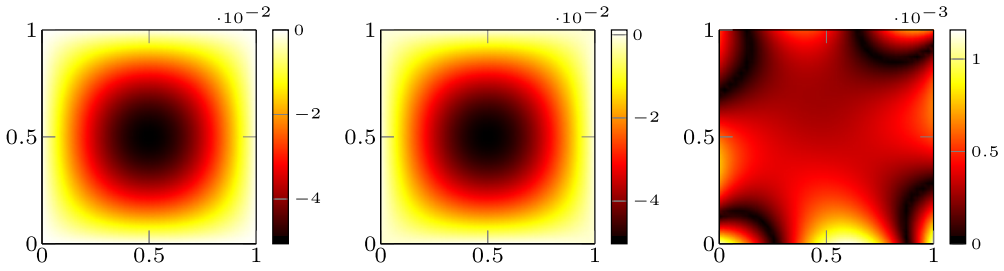

Fig. A.7. Poisson without extra features. *Left.* Analytical solution. *Center.* PINN solution. *Right.* Pointwise error between the analytical and the PINN solution.

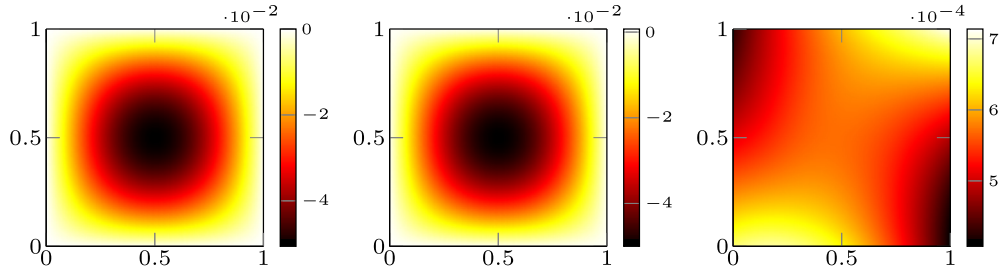

Fig. A.8. Poisson with extra features. *Left.* Analytical solution. *Center.* PINN solution. *Right.* Pointwise error between the analytical and the PINN solution.

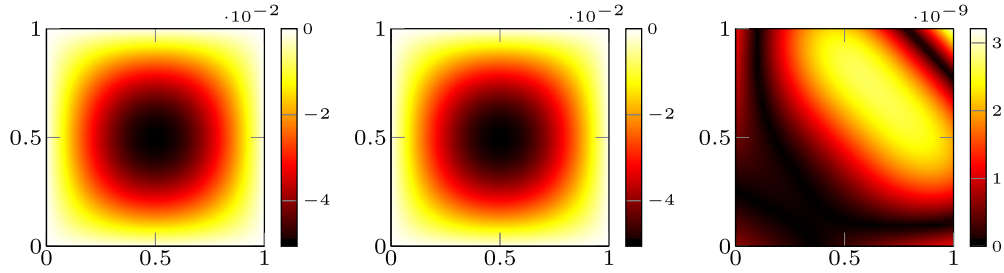

Fig. A.9. Poisson with learnable extra features. *Left.* Analytical solution. *Center.* PINN solution. *Right.* Pointwise error between the analytical and the PINN solution.

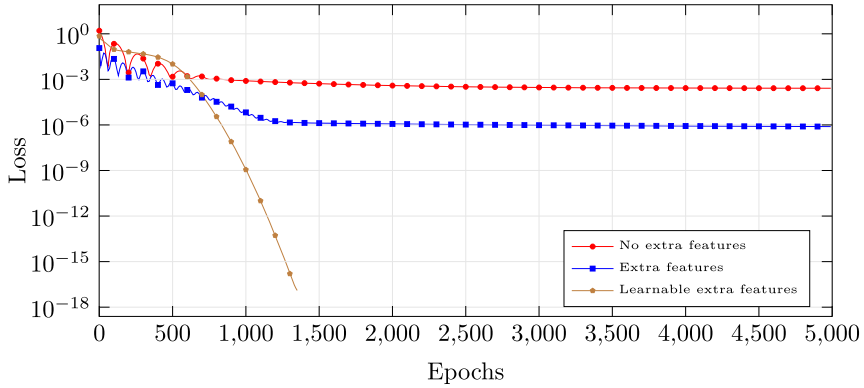

Fig. A.10. Poissoin problem. Comparison between the three approaches (no extra features, with extra features and with learnable extra features) in terms of loss value with respect to the number of epochs.

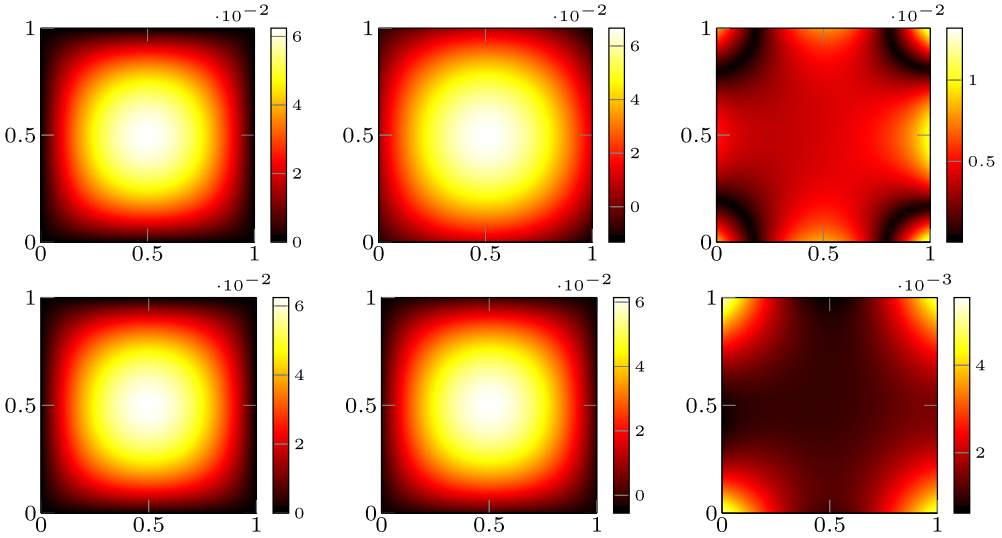

Fig. A.11. Second Poisson problem. *Top row*. Representative solution and pointwise error without extra features. *Bottom row*. Representative solution and pointwise error with extra features. *Left.* Analytical solution. *Center.* PINN solution. *Right.* Pointwise error between the analytical and the PINN solution.

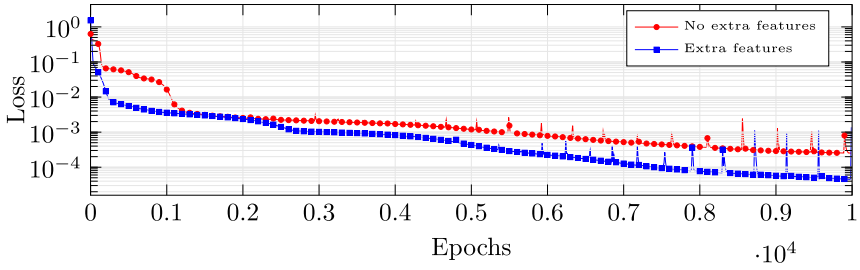

Fig. A.12. Second Poissoin problem. Comparison between the two approaches (no extra features and with extra features) in terms of loss value with respect to the number of epochs.

A.1.1. Poisson problem with a different forcing term

The previous results have shown a remarkable gain in the usage of extra features. It must be said that in that specific case the forcing term we impose is linearly related with the analytical solution. It implies that, using such forcing term as an extra feature, the input-output mapping we want to discover during the learning procedure results simplified.

To better test the generability of the framework, we keep the original model (A.1) varying the forcing term such as:

$$f(\mathbf{x}) \doteq -2(x_1(1-x_1)+x_0(1-x_0)).$$

The analytical solution of the problem becomes so:

$$w(\mathbf{x}) = x_0(1-x_0)x_1(1-x_1). \tag{A.5}$$

Regarding the neural network architecture, we keep the same setting of the previous experiments: 2 hidden layers composed by 10 neurons each, with the Softplus activation function. The optimization has carried out by means of the ADAM algorithm with learning rate LR = 0.003. The cardinality of the point set where we evaluate the residuals are $N_p = 100$ and $N_b = 40$, distributed using a cartesian grid. As before, we solve the problem using PINN, initially without any extra features, then using the forcing term as extra feature. Formally:

$$k_1(\mathbf{x}) \doteq -2(x_1(1-x_1)+x_0(1-x_0)). \tag{A.6}$$

The comparison between the two approaches — with and without the feature — aims to empirically prove the effectiveness of the extra features when the latter are not linearly related with the solution of the problem. Fig. A.11 presents the results obtained after a fixed amount of epochs (1.0000×10^4). The error distribution shows a better accuracy by involving the extra feature, which is demonstrated also by the visual comparison of the PINN solution with respect to the analytical solution. The error with extra features reaches an order of magnitude less than the error without features. This behaviour is also confirmed by the loss function in the two cases represented in Fig. A.12: the optimization convergence is increased by using the forcing term as an additional features also in this case.

A.2. Burgers problem

We are now going to test the example presented in [40] concerning Burgers's equation. Also in this case, our goal is to state the advantages of employing some previous knowledge on the system enriching the input through some extra features. Considering $\mathbf{x} = (x, t)$, the PDE we are dealing with is:

$$\begin{cases} w(\mathbf{x})_t + w(\mathbf{x})w(\mathbf{x})_x - \dfrac{0.01}{\pi} w(\mathbf{x})_{xx} = 0 & \text{for } x \in [-1,1] \text{ and } t \in [0,1], \\ w(1,t) = w(-1,t) = 0, \\ w(x,0) = -\sin(\pi x). \end{cases} \tag{A.7}$$

Let us fix the following architecture for the PINN we want to approximate the physical phenomenon at hand. We employed a 3–Layer NN, where the number of neurons — from first to last layer — are equal to 20, 10 and 5. As the previous test cases, we used ADAM optimizer with LR = 0.006 and a hyperbolic tangent as an activation function. The $N_p = 8.000 \times 10^3$ internal points are distributed using a latin hyper-cube sampling, whereas the $N_b = 150$ boundary points are selected with a

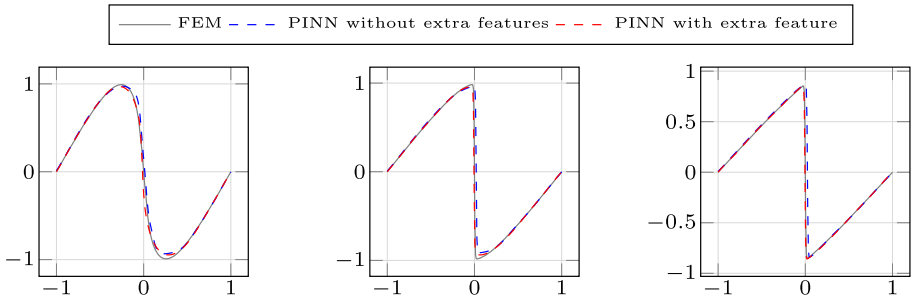

Fig. A.13. Burgers problem with and without extra features after 1.0000×10^4 epochs. *Left.* $t = 0.25$. *Center.* $t = 0.5$. *Right.* $t = 0.75$.

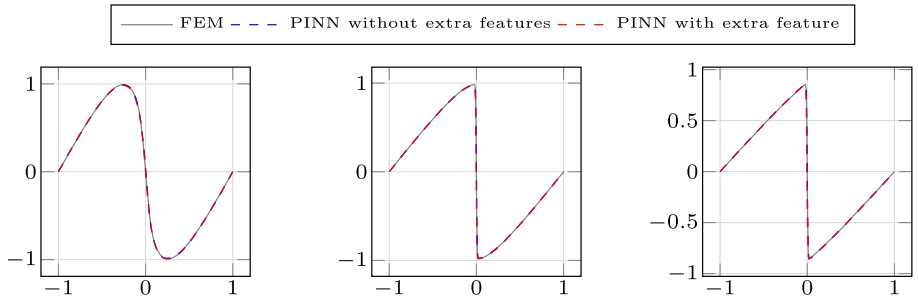

Fig. A.14. Burgers problem with and without extra features when loss is less than 1×10^{-4}. *Left.* $t = 0.25$. *Center.* $t = 0.5$. *Right.* $t = 0.75$.

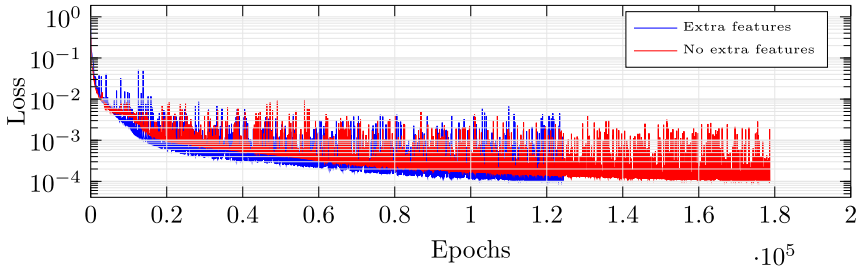

Fig. A.15. Burgers problem. *Left.* Comparison between the two approaches (no extra features and with extra features) in terms of loss value with respect to the number of epochs. *Right.* Representation of the boundary and internal collocation points.

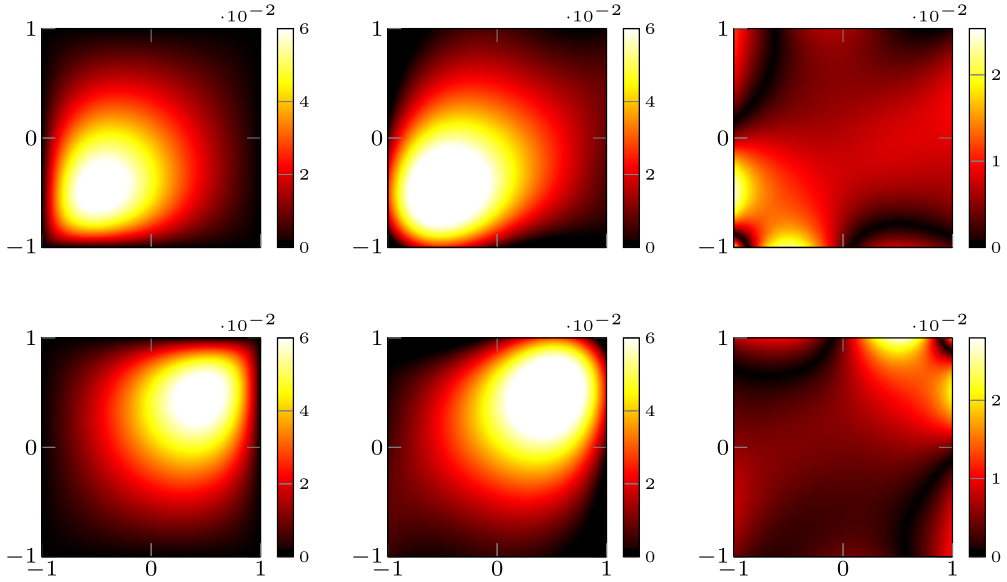

Fig. A.16. Parametric Poisson. *Top row*. Representative solution and pointwise error for $\boldsymbol{\mu} = (-0.8, -0.8)$. *Bottom row*. Representative solution and pointwise error for $\boldsymbol{\mu} = (0.8, 0.8)$. *Left.* Numerical solution. *Center.* PINN solution. *Right.* Pointwise error between the numerical and the PINN solution.

uniform random distribution. Figs. A.13 and A.14 show the comparison between standard PINN without features and with the extra features, after a fixed number of epochs (1.0000×10^4) and once the loss has reached the prescribed tolerance (1×10^{-4}).

Let us comment on the results we obtained. In Fig. A.13 we show the improvements obtained by exploiting the following feature (that is nothing but the initial condition we impose in the problem) in the training procedure

$$k_1(x) \doteq \sin(x\pi). \tag{A.8}$$

It is possible to note that we gain accuracy at instants $t \in \{0.25, 0.5, 0.75\}$. The advantage of involving an extra features is however less remarkable than in the previous numerical experiments: our hypothesis is that the initial condition does not present the shock we expect after a certain time in the Burgers solution, requiring a great number of epochs to learn it accurately. Consistently, Fig. A.15 presents such marginal differences also in the loss trend. The use of the extra feature allows us to reach the imposed tolerance (1×10^{-4}) after 1.20000×10^5, contrarily to standard PINN formulation that requires 1.80000×10^5 for the same threshold. Of course, selecting a tolerance threshold as the stopping criteria of the learning procedure makes both the approaches — with and without extra features — converging to the truth solution (Fig. A.14). Also in this more complicated case, we can state that introducing the feature information can be of advantage in order to reduce the training time. We would like to stress that in term of the pointwise error, the results might be improved choosing tailored features of working with more hidden layers and/or neurons. However, this went beyond the aim of this experiment.

A.3. Parametric Poisson problem

This Section focuses on the application of PINN in a parametric context. We propose the following problem: given a parametric instance $\boldsymbol{\mu} \doteq (\mu_1, \mu_2) \in [-1,1]\times[-1,1]$, find the solution of

$$\begin{cases} -\Delta w(\mathbf{x},\boldsymbol{\mu}) = f(\mathbf{x},\boldsymbol{\mu}) & \text{in } \Omega, \\ w(\mathbf{x},\boldsymbol{\mu}) = 0 & \text{on } \partial\Omega, \end{cases} \tag{A.9}$$

where $\Omega = [-1,1]\times[-1,1]$ and the forcing term is

$$f(\mathbf{x},\boldsymbol{\mu}) = e^{-2((x_0-\mu_1)^2+(x_1-\mu_1)^2)}. \tag{A.10}$$

Our aim is to recover the solution $w(\mathbf{x},\boldsymbol{\mu})$ for several parametric instances after a training phase which takes into consideration how the system changes with respect to these physical parameters. The information is given by the modified loss (4). In order to achieve this prediction goal, we used an ADAM optimizer over a 3–Layers NN of 20 neurons for each hidden layer. We chose a Softplus activation function and LR = 0.03. The number of the collocation points are given by: $N_p = 400$, $N_b = 80$ and $N_{\boldsymbol{\mu}} = 40$, all considered with an equispaced distribution. The plots in Fig. A.16 shows satisfactory results for a 1000 epochs training. It depicts the behaviour of two parametric solutions, i.e. $w(\mathbf{x},(-0.8,-0.8))$ and $w(\mathbf{x},(0.8,0.8))$, respectively in the top and bottom row. For both the values we show a FE numerical simulation and its PINN counterpart together with the pointwise error which stays below the value of $3\cdot10^{-2}$. Also in this case, the input features of the NN model are enriched by the forcing term of the equations in order to improve the training. The extra feature is defined as $k_1(x) \doteq e^{-2((x_0-\mu_1)^2+(x_1-\mu_1)^2)}$.

References

[1] A. Atangana, A. Kılıçman, Analytical solutions of boundary values problem of 2D and 3D Poisson and biharmonic equations by homotopy decomposition method, Abstr. Appl. Anal. 2013 (Sep 2013) 380484.

[2] F. Ballarin, E. Faggiano, A. Manzoni, A. Quarteroni, G. Rozza, S. Ippolito, C. Antona, R. Scrofani, Numerical modeling of hemodynamics scenarios of patient-specific coronary artery bypass grafts, Biomech. Model. Mechanobiol. 16 (4) (2017) 1373–1399.

[3] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018).

[4] P.B. Bochev, M.D. Gunzburger, Least-Squares Finite Element Methods, vol. 166, Springer-Verlag, New York, 2009.

[5] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks for heat transfer problems, J. Heat Transf. 143 (6) (2021) 060801.

[6] J.C. de los Reyes, F. Tröltzsch, Optimal control of the stationary Navier–Stokes equations with mixed control-state constraints, SIAM J. Control Optim. 46 (2) (2007) 604–629.

[7] L. Dede, Optimal flow control for Navier–Stokes equations: drag minimization, Int. J. Numer. Methods Fluids 55 (4) (2007) 347–366.

[8] C.H. Ding, I. Dubchak, Multi-class protein fold recognition using support vector machines and neural networks, Bioinformatics 17 (4) (2001) 349–358.

[9] S. Goswami, A. Bora, Y. Yu, G.E. Karniadakis, Physics-informed neural operators, arXiv preprint, arXiv:2207.05748, 2022.

[10] M.D. Gunzburger, Perspectives in Flow Control and Optimization, vol. 5, SIAM, Philadelphia, 2003.

[11] M. Guo, A. Manzoni, M. Amendt, P. Conti, J.S. Hesthaven, Multi-fidelity regression using artificial neural networks: efficient approximation of parameter-dependent output quantities, Comput. Methods Appl. Mech. Eng. 389 (2022) 114378.

[12] J.S. Hesthaven, G. Rozza, B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations, SpringerBriefs in Mathematics, Springer, Milano, 2015.

[13] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, Optimization with PDE Constraints, vol. 23, Springer Science & Business Media, Antwerp, 2008.

[14] K. Ito, K. Kunisch, Lagrange Multiplier Approach to Variational Problems and Applications, vol. 15, SIAM, Philadelphia, 2008.

[15] A.D. Jagtap, K. Kawaguchi, G. Em Karniadakis, Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, Proc. R. Soc. A 476 (2239) (2020) 20200334.

[16] A.D. Jagtap, K. Kawaguchi, G.E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, J. Comput. Phys. 404 (2020) 109136.

[17] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems, Comput. Methods Appl. Mech. Eng. 365 (2020) 113028.

[18] X. Jin, S. Cai, H. Li, G.E. Karniadakis, NSFnets (Navier-Stokes flow nets): physics-informed neural networks for the incompressible Navier-Stokes equations, J. Comput. Phys. 426 (2021) 109951.

[19] E. Kharazmi, Z. Zhang, G.E. Karniadakis, hp-VPINNs: variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Eng. 374 (2021) 113547.

[20] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[21] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.

[22] T. Lassila, A. Manzoni, A. Quarteroni, G. Rozza, A reduced computational and geometrical framework for inverse problems in hemodynamics, Int. J. Numer. Methods Biomed. Eng. 29 (7) (2013) 741–776.

[23] G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, S. Ulbrich, Trends in PDE Constrained Optimization, Springer, New York, 2014.

[24] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, arXiv preprint, arXiv:2111.03794, 2021.

[25] X. Lin, Y. Rivenson, N.T. Yardimci, M. Veli, Y. Luo, M. Jarrahi, A. Ozcan, All-optical machine learning using diffractive deep neural networks, Science 361 (6406) (2018) 1004–1008.

[26] J.L. Lions, Optimal Control of System Governed by Partial Differential Equations, vol. 170, Springer-Verlagr, Berlin and Heidelberg, 1971.

[27] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (Mar 2021) 218–229.

[28] M. Mahmoudabadbozchelou, M. Caggioni, S. Shahsavari, W.H. Hartt, G. Em Karniadakis, S. Jamali, Data-driven physics-informed constitutive metamodeling of complex fluids: a multifidelity neural network (MFNN) framework, J. Rheol. 65 (2) (2021) 179–198.

[29] X. Meng, Z. Li, D. Zhang, G.E. Karniadakis, PPINN: parareal physics-informed neural network for time-dependent PDEs, Comput. Methods Appl. Mech. Eng. 370 (2020) 113250.

[30] M. Motamed, A multi-fidelity neural network surrogate sampling method for uncertainty quantification, Int. J. Uncertain. Quantificat. 10 (4) (2020).

[31] S. Mowlavi, S. Nabi, Optimal control of PDEs using physics-informed neural networks, arXiv preprint, arXiv:2111.09880, 2021.

[32] F. Negri, A. Manzoni, G. Rozza, Reduced basis approximation of parametrized optimal flow control problems for the Stokes equations, Comput. Math. Appl. 69 (4) (2015) 319–336.

[33] G. Pang, M. D'Elia, M. Parks, G.E. Karniadakis, nPINNs: nonlocal physics-informed neural networks for a parametrized nonlocal universal Laplacian operator. Algorithms and applications, J. Comput. Phys. 422 (2020) 109760.

[34] W. Peng, W. Zhou, J. Zhang, W. Yao, Accelerating physics-informed neural network training with prior dictionaries, arXiv preprint, arXiv:2004.08151, 2020.
[35] M. Pošta, T. Roubíček, Optimal control of Navier–Stokes equations by Oseen approximation, Comput. Math. Appl. 53 (3–4) (2007) 569–581.
[36] C. Prud'Homme, D.V. Rovas, K. Veroy, L. Machiels, Y. Maday, A. Patera, G. Turinici, Reliable real-time solution of parametrized partial differential equations: reduced-basis output bound methods, J. Fluids Eng. 124 (1) (2002) 70–80.
[37] A. Quarteroni, G. Rozza, L. Dedè, A. Quaini, Numerical approximation of a control problem for advection-diffusion processes, in: IFIP Conference on System Modeling and Optimization, Springer, 2005, pp. 261–273.
[38] A. Quarteroni, G. Rozza, A. Quaini, Reduced basis methods for optimal control of advection-diffusion problems, Technical report, RAS and University of Houston, 2007.
[39] A. Quarteroni, A. Valli, Numerical Approximation of Partial Differential Equations, vol. 23, Springer Science & Business Media, Berlin and Heidelberg, 2008.
[40] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[41] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.
[42] M. Strazzullo, F. Ballarin, R. Mosetti, G. Rozza, Model reduction for parametrized optimal control problems in environmental marine sciences and engineering, SIAM J. Sci. Comput. 40 (4) (2018) B1055–B1079.
[43] M. Strazzullo, Z. Zainib, F. Ballarin, G. Rozza, Reduced order methods for parametrized non-linear and time dependent optimal flow control problems, towards applications in biomedical and environmental sciences, in: Numerical Mathematics and Advanced Applications ENUMATH 2019, Springer, 2021, pp. 841–850.
[44] F. Tröltzsch, Optimal Control of Partial Differential Equations, Graduate Studies in Mathematics, vol. 112, Verlag, Wiesbad, 2010.
[45] D. Wang, W. Liao, Modeling and control of magnetorheological fluid dampers using neural networks, Smart Mater. Struct. 14 (1) (2004) 111.
[46] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed DeepONets, Sci. Adv. 7 (40) (2021) eabi8605.
[47] L. Yang, X. Meng, G.E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, J. Comput. Phys. 425 (2021) 109913.
[48] Z. Zainib, F. Ballarin, S. Fremes, P. Triverio, L. Jiménez-Juan, G. Rozza, Reduced order methods for parametric optimal flow control in coronary bypass grafts, toward patient-specific data assimilation, Int. J. Numer. Methods Biomed. Eng. (2020) e3367.