A physics-informed multi-fidelity approach for the estimation of differential equations parameters in low-data or large-noise regimes

F. Regazzoni [1,*], S. Pagani [1,*,†], A. Cosenza [1], A. Lombardi [1], A. Quarteroni [1,2]

[1] MOX-Department of Mathematics, Politecnico di Milano, Milan, Italy
[2] Institute of Mathematics, EPFL, Lausanne, Switzerland (*Professor Emeritus*)
[*] *These authors equally contributed to this work*
[†] *Corresponding author*

{francesco.regazzoni,stefano.pagani,alfio.quarteroni}@polimi.it
{alessandro.cosenza,alessandro3.lombardi}@mail.polimi.it

This paper is dedicated to the memory of Claudio Baiocchi

Abstract

In this paper, we propose a multi-fidelity approach for parameter estimation problems based on physics-informed neural networks (PINNs). The proposed methods apply to models expressed by linear or nonlinear differential equations, whose parameters need to be estimated starting from (possibly partial and noisy) measurements of the model's solution. To overcome the limitations of PINNs in case only few and/or significantly noisy data are available, we propose to train a first artificial neural network (ANN), that provides a low-fidelity solution, using a dataset constructed with numerical simulations of the parametrized mathematical model. Then, we express the solution as the sum of the low-fidelity solution and a correction term, provided by a second ANN, which is trained by simultaneously minimizing the distance of the solution from the data and the residual of the differential equation, evaluated at a suitable set of collocation points. For the setup of the multi-fidelity formulation, we propose two alternative approaches. In the first one, we train the low-fidelity solution with data obtained for a fixed value of the parameters, representing an initial guess for the parameters to be estimated (possibly deriving from prior knowledge). In the second approach, the low-fidelity solution encodes the parametric dependence, being trained on data generated by sampling the parameter space and including the parameters themselves among its independent variables. Numerical tests performed for two benchmark problems, a steady-state and a time-dependent advection-diffusion-reaction PDEs, show that the first approach allows the convergence of the PINN training to accelerate significantly. The second approach, in addition to improving the convergence speed, increases the accuracy of the parameters estimate. This is emphasized in cases characterized by a small number of available measurements (possibly affected by noise). Finally, we show an application of the proposed multi-fidelity approach in the context of cardiac electrophysiology. Specifically, we employ a multi-fidelity PINN to estimate an unknown parameter of a nonlinear model describing the ionic dynamics of cardiac cells, starting from noisy measurements of the transmembrane potential.

© Copyright 2021, EMS Press (European Mathematical Society)

This preprint has been published in *Atti della Accademia Nazionale dei Lincei, Classe di Scienze Fisiche, Matematiche e Naturali. Rendiconti Lincei - Matematica e Applicazioni* 32(3): 437–470 (https://doi.org/10.4171/RLM/943)

1 Introduction

Artificial Neural Networks (ANNs) are powerful architectures made of interconnected nodes, known as artificial neurons, which can learn arbitrary complex input-output functions by processing paired input and output samples (the training data) [18]. Each artificial neuron is characterized by an activation function, which is a function that, e.g., maps large negative inputs to zero (deactivated neuron) and large positive inputs to one (activated neuron), and a set of *parameters*, namely weights and biases. These *parameters* modulate the activation of each neuron: weights scale the neuron's inputs, while the bias controls the tendency of the neuron to be active or not. An artificial neural network is therefore an input-output function that depends on the designed topology, i.e. on the number of neurons and their connections, and on the set of parameters (weights and biases) that characterize each neuron. The training process consists of identifying the parameters that allow the network to reproduce – as faithfully as possible – the available output values given the paired inputs. This is achieved by finding the set of parameters that minimize a cost functional (loss) in which the discrepancy between the network evaluation and the actual data is penalized. Besides the ANN parameters, the training process is determined by a set of so–called *hyperparameters*, which are variables used to control the training process itself. These include architecture hyperparameters (e.g. number of neurons and layers) and algorithm hyperparameters (e.g. the settings of the minimization algorithm).

This learning strategy typically presents numerous challenges due to the high dimensionality of the parameters space and the non-convexity of the cost functional. Identifying an optimal set of parameters is often an arduous task, especially in the so-called *small data regime* when a little amount of (noisy) data is available. This is typically the case of biological or engineering systems, where the goal is to model physical processes starting from data acquired from few sensors or from a small cohort of patients.

A possible strategy to overcome these data limitations consists of introducing in the loss functional an additional term (a regularization term constraining the ANN solution to a manifold of differential equations' solutions) that encodes prior physical knowledge of the observed phenomenon. This strategy, firstly proposed in [25, 17], gave rise to the so-called Physics Informed Neural Networks (PINNs) [30, 31], which have found application in numerous fields, from the identification of the flux in an aneurysm [32] to the reconstruction of cardiac activation maps [33].

PINNs' success is not limited to the approximation of differential problems' solution: their structure permits to treat unknown model's parameters as additional parameters of the neural network, thus solving state/parameter estimation problems within a single minimization (training) procedure (without changing the framework). This makes PINN attractive for parameter estimation problems, compared to PDE-constrained optimization routines, where gradient descent methods are built on the repeated (and computationally expensive) numerical approximation of differential problems [3].

In many realistic scenarios, high-fidelity mathematical models, that accurately describe biological or engineering systems, are formed by coupled (typically nonlinear) equations. This makes the regularization term (formed by the residual of the equations) less effective, making the search for sub-optimal local minima, sufficiently close to the global minimum, more complex. Sometimes it is more effective to rely on simplified models to obtain a more effective regularization term. These low-fidelity models may result from simplifications of equations (simplified physics approximation), which might partially describe the phenomenon. As an example, in the field of cardiac electrophysiology, the equations of the bidomain model (high-fidelity) are frequently approximated with the monodomain equation, which represents only the transmembrane potential, or the eikonal equation that allows describing only the dynamics of the activation front [12, 27]. Black-box data-driven models, built from a set of data linked to the observed phenomenon (such as numerical simulations of a high-fidelity

model), may also be adopted as low-fidelity models.

Multi-fidelity methods have been developed in recent years to combine models of different fidelities to efficiently solve complex tasks such as optimization or uncertainty quantification [24, 11, 15]. In these procedures, neural networks have been adopted as low-cost surrogate models that allow for speeding up (Quasi)-Monte Carlo algorithms [19] or parameter estimation routines [23, 20]. This idea has also been adapted in the PINN framework in [22], where an architecture composed of multiple neural networks is trained on data coming from different fidelity models.

In this work we propose a new multi-fidelity PINN, based on the multi-fidelity strategy of *adaptation*, which consists in improving the accuracy of a low-fidelity prediction with a correction combining data coming from the high-fidelity model (PDE residual) with the available (possibly corrupted by noise) measures. This is perform adaptively during the training procedure, leveraging a pre-trained low-fidelity model, such as a neural network or a general surrogate model. This strategy is alternative to the one presented in [22], where the training procedure is simultaneous and the multi-fidelity is expressed in the availability of additional data. Compared to standard approaches to PDE-constrained optimization, in which surrogate models or reduced-order models are integrated within gradient-based minimization algorithms [10, 21, 7, 8, 38, 2], in our framework the low-fidelity model is integrated directly into the PINN and the training procedure is designed to directly solve the parameter estimation problem.

We develop a training procedure divided into two different stages: the low-fidelity model is firstly built starting from a dataset generated, e.g., from a numerical model of the physical phenomenon; then the multi-fidelity network is trained using measurements data, the low-fidelity guess, and the residual of the equation modeling the physical phenomenon. Compared to a traditional PINN, the training of the network does not start from a completely random solution but relies on the low-fidelity solution, which acts as a priori information that can then be corrected using the new information coming from the measured data and by regularizing it through the mathematical equation (crucial for the task of model parameter estimation). This strategy speeds up the convergence of the training process, even in situations where the low-fidelity model represents a solution that is far from the current one. On the contrary, a multi-fidelity network built simultaneously might be affected by numerous local minima created by the interaction of poorly trained neural nets, from which the optimizer might hardly escape. The process can therefore be seen as a solution refinement process, obtained through increasingly fine regularization processes, moving from a data-driven low fidelity surrogate to the high-fidelity equation.

We rely on two different approaches for the low-fidelity model construction: a fixed-parameter approach versus a parameterized one, which mimics the construction of a reduced-order model. The first approach allows regularizing the solution with respect to space and time, with a consequent improvement of the speed of convergence of the training procedure. The second one, instead, drives the parameter identification, reducing the error in the estimation even in cases where standard PINNs are far from the true solutions.

We emphasize that the proposed approach is general and that the multi-fidelity network can be based on all types of low-fidelity models, such as data-driven surrogate models [10] or projection-based reduced-order models [28, 14]. This new paradigm might significantly improve, both in terms of accuracy and overall computational time, parameter estimation tasks that are currently solved through computationally intensive PDE-constrained optimization techniques.

The rest of this article is organized as follows. Sec. 2 introduces the parameter estimation problem and describes the new multi-fidelity approaches. Numerical results are described in Sec. 3. Finally, conclusions are drawn in Sec. 4.

2 Methods

In this work, we consider PDE-constrained optimal control problems [34, 3], focussing on the finite-dimensional control case (i.e. with control variable expressed by a finite vector of unknown parameters $\boldsymbol{\mu} \subset \mathcal{P} \subset \mathbb{R}^p$), under the form:

$$\begin{cases} \min\limits_{\boldsymbol{\mu}\in\mathcal{P}} \; J(\boldsymbol{\mu}) \\ \text{s.t.} \quad \alpha\dfrac{\partial u}{\partial t} + \mathcal{L}(u;\boldsymbol{\mu}) = 0 \quad \text{in } \Omega \times [0,T], \\ \text{where} \quad J(\boldsymbol{\mu}) = \|Cu(\mathbf{x},t;\boldsymbol{\mu}) - y(\mathbf{x},t)\|^2_{\mathcal{Y}} + R(u;\boldsymbol{\mu}) \end{cases} \tag{1}$$

where $u = u(\mathbf{x}, t; \boldsymbol{\mu})$ is the solution of the so-called forward problem, $\mathbf{x}$ is the space variable defined in the domain Ω, t is the time variable defined in the interval $[0, T]$. Here, $\mathcal{L}$ is a (possibly nonlinear) differential operator. Stationary PDEs can be recast in the form (1) by setting $\alpha = 0$ and restricting the domain to Ω. Boundary conditions could be included as well. In the cost functional J, the function $C : \mathcal{X} \to \mathcal{Y}$ is a linear observation operator, mapping the solution of the forward problem onto the space of observation $\mathcal{Y} = \mathcal{Y}(\Omega)$, while R is a suitable regularization term. In this form, the cost functional measures the discrepancy between the numerical model output Cu and the set of available measurements y (the target).

After temporal and spatial discretization of the PDE (e.g. by Finite Differences of Finite elements, respectively [26]), standard solution approaches are the descent methods [9, 29], which are iterative procedures based on repeated steps along descent directions, defined at each step by the numerical approximation of the forward and the adjoint (or sensitivities) problems. The effectiveness of these procedures has been demonstrated in many applications ranging from aerodynamics to solid mechanics [4]. However, the resolution in the case of high-dimensional parametric space and multiscale and multiphysics differential problems is still challenging, due to the high-computational costs associated with the computation of the gradient of the cost functional and the large number of iterations required. These challenges are common to those found in identifying neural network parameters (high dimensionality of the parameters space and the non-convexity of the loss functional). For this reason, methods based on automatic differentiation and stochastic gradients are promising tools in this context. In Sec. 2.1 we describe the PINN approach for the simultaneous PDE solution and parameter estimation from (noisy) measurements, while in Sec. 2.2 we introduce our novel multi-fidelity strategy.

2.1 Neural Network-based approaches to the optimal control problem

The PDE-constrained optimal control problem (1) can be reformulated as a statistical learning problem: find a function $u(\mathbf{x}, t)$, expressed through an Artificial Neural Network (ANN) $\mathcal{NN}$, which minimizes the cost functional J while satisfying - in a suitable sense - the PDE. Typically, observations of the solution are only available at given spatio-temporal locations $(\mathbf{x}_i^H, t_i^H)$, $i = 1, \ldots, N_{\text{obs}}^H$. Hence, the observations are given by $y_i^H = Cu(\mathbf{x}_i^H, t_i^H; \boldsymbol{\mu}_{ex}) + \epsilon_i$, where ϵ_i denotes the measurement error and where the solution u is associated with $\boldsymbol{\mu} = \boldsymbol{\mu}_{ex}$, i.e. the "exact" value of the parameter. This dataset is usually denoted as training data.

The optimal control problem is then expressed in the following way:

$$\begin{cases} \min\limits_{\boldsymbol{\mu},\mathbf{W}} \ \mathcal{J}(\boldsymbol{\mu},\mathbf{W}) \\ \text{s.t.} \quad u(\mathbf{x},t) = \mathcal{NN}(\mathbf{x},t;\boldsymbol{\mu},\mathbf{W}) \\ \text{where} \quad \mathcal{J}(\boldsymbol{\mu},\mathbf{W}) = \dfrac{\omega_{\text{fit}}}{N_{\text{obs}}^H} \displaystyle\sum_{i=1}^{N_{\text{obs}}^H} |Cu(\mathbf{x}_i^H, t_i^H) - y_i^H|^2 + R(u;\boldsymbol{\mu},\mathbf{W}), \end{cases} \tag{2}$$

where the cost functional J is substituted by the empirical cost functional $\mathcal{J}$; the latter can be viewed as a Monte-Carlo approximation of J. Here, $\mathbf{W} = [\mathbf{w}, \mathbf{b}]$ represents the parameters (weights $\mathbf{w}$ and biases $\mathbf{b}$, respectively) of the ANN, which acts as additional unknowns of the problem. The scalar hyperparameter $\omega_{\text{fit}} > 0$ weights the contribution of data misfit with respect to the regularization term R. It is important to remark that after reformulating (1) as (2), the physical information associated with the PDE is lost. To recover it, one possiblity is offered by PINNs, which encode the PDE into the regularization term R.

More precisely, the regularization term R is chosen as the absolute value of the PDE residual:

$$R_{\text{phys}}(u;\boldsymbol{\mu},\mathbf{W}) = \frac{\omega_{\text{phys}}}{N_c} \sum_{i=1}^{N_c} \left| \alpha \frac{\partial u(\mathbf{x}_i^c, t_i^c)}{\partial t} + \mathcal{L}(u(\mathbf{x}_i^c, t_i^c);\boldsymbol{\mu}) \right|^2, \tag{3}$$

where $(\mathbf{x}_i^c, t_i^c)$, $i = 1, \ldots, N_c$ is a set of N_c collocation points, at which we penalize the residual of the PDE in strong form, and $\omega_{\text{phys}} \geq 0$ is a balancing factor. Should the boundary conditions be available, they could be enforced in (3) by summing the absolute value of the associated residual. For simplicity, in this work we do not consider this case, as the generalization of the proposed methods is straightforward.

Provided that sufficiently regular activation functions are employed, the ANN is infinitely many times continuously differentiable ($u \in \mathcal{C}^\infty(\Omega \times [0,T])$). Therefore, the differential operators $\frac{\partial}{\partial t}$ and $\mathcal{L}$ can be meaningfully applied to u. Moreover, Automatic Differentiation (AD) tools – at the basis of virtually any Machine Learning software library – can be used to practically and efficiently evaluate the term (3), enabling a fast implementation of this method.

Although PINNs allow solving optimal control problems of the type of (1) by exploiting the powerful tools developed in the context of Machine Learning and with little and intuitive implementation effort, training these ANNs can present several difficulties [37, 36], especially for few and noisy data. In the low-data regime, indeed, the problem turns out to be ill-posed and characterized by many local minima, which makes the training phase very difficult. To overcome these difficulties, in this paper we rely on a multi-fidelity approach.

2.2 Multi-fidelity approach

We consider a multi-fidelity approach to tackle the ill-posedness of the optimal control problem. Our goal is to combine ANNs built on different datasets and with different loss functions to solve the parametric optimal control problem. We first express the solution $u(\mathbf{x},t)$ as the sum of a low-fidelity model solution u_L, and an ANN ($\mathcal{NN}_H$), acting as a correction of the low-fidelity guess:

$$u(\mathbf{x},t) = u_L + \mathcal{NN}_H(\mathbf{x}, t, u_L; \mathbf{W}_H), \tag{4}$$

In this paper, u_L is obtained with an additional ANN (named as low-fidelity ANN), but in general it can be constructed through different techniques, including fitting techniques of projection-based reduced-order modeling techniques.

For the training of these two ANNs we consider two sets of data:

1. *Low-fidelity dataset*: approximations of the parametric solution $y_i^L = Cu(\mathbf{x}_i^L, t_i^L; \boldsymbol{\mu}_i^L)\}$, $i = 1, \dots, N_{\text{obs}}^L$, obtained, e.g. with the chosen numerical approximation of the forward problem corresponding to different samples $\boldsymbol{\mu}_i^L$ in the parameter space. Note that N_{obs}^L could be arbitrary large, since it can be built through the offline execution of numerical simulations;

2. *High-fidelity dataset*: y_i^H, $i = 1, \dots, N_{\text{obs}}^H$, which consists of a set of available measurements.

Compared to the usual PINN approach, in which the physics-informed component is contained only in the regularization term R based on the model residual, we also augment the training dataset with additional data potentially obtained through numerical simulations. This has a double advantage: it allows creating *offline* the *low-fidelity dataset* used for the training of the low-fidelity surrogate u_L, which then acts as a regularizer of the optimal control problem.

We propose two different strategies for solving parameter estimation problems with multi-fidelity PINNs, that we respectively denote by MPINN-v1 and MPINN-v2. Before introducing them, we describe an alternative multi-fidelity PINN framework, that we denote by MPINN-v0, following the approach presented in [22].

2.2.1 MPINN-v0

We consider a collection of solution snapshots obtained for different samples of the parameter $\boldsymbol{\mu}$, that is $\{u(\mathbf{x}_i^L, t_i^L; \boldsymbol{\mu}_i^L) \colon i = 1, \dots, N_{\text{obs}}^L\}$. Following [22], we here consider a more general formulation of (4), that is

$$u(\mathbf{x}, t) = \mathscr{L}_H(\mathbf{x}, t, u_L; \mathbf{W}_H) + \mathcal{NN}_H(\mathbf{x}, t, u_L; \mathbf{W}_H), \tag{5}$$

where $\mathscr{L}_H$ is a linear function and $\mathcal{NN}_H$ is a nonlinear one (specifically an ANN). The vector $\mathbf{W}_H$ collects the parameters of both maps $\mathscr{L}_H$ and $\mathcal{NN}_H$. The multi-fidelity PINN problem then reads

$$\begin{cases} \min\limits_{\boldsymbol{\mu}, \mathbf{W}_L, \mathbf{W}_H} \ \mathcal{J}(\boldsymbol{\mu}, \mathbf{W}_L, \mathbf{W}_H) \\ \quad \text{s.t.} \quad u_L(\mathbf{x}, t) = \mathcal{NN}_L(\mathbf{x}, t; \mathbf{W}_L), \\ \qquad\ \ u(\mathbf{x}, t) = \mathscr{L}_H(\mathbf{x}, t, u_L(\mathbf{x}, t); \mathbf{W}_H) + \mathcal{NN}_H(\mathbf{x}, t, u_L(\mathbf{x}, t); \mathbf{W}_H) \\ \quad \text{where} \quad \mathcal{J}(\boldsymbol{\mu}, \mathbf{W}_L, \mathbf{W}_H) = \dfrac{\omega_{\text{fit}}^L}{N_{\text{obs}}^L} \displaystyle\sum_{i=1}^{N_{\text{obs}}^L} |Cu_L(\mathbf{x}_i^L, t_i^L) - y_i^L|^2 + R_L(u_L; \mathbf{W}_L) \\ \qquad\qquad\qquad\qquad + \dfrac{\omega_{\text{fit}}^H}{N_{\text{obs}}^H} \displaystyle\sum_{i=1}^{N_{\text{obs}}^H} |Cu(\mathbf{x}_i^H, t_i^H) - y_i^H|^2 + R_H(u; \boldsymbol{\mu}, \mathbf{W}_H). \end{cases} \tag{6}$$

The scaling constants ω_{fit}^L and ω_{fit}^H allow to tune the relative weights of the loss function terms. As in [22], we do not regularize the low-fidelity map (that is, we set $R_L \equiv 0$), while we consider the physics-informed regularizer on the solution $u(\mathbf{x}, t)$ (that is, $R_H = R_{\text{phys}}$, defined in (3)). We remark that, within the MPINN-v0 approach, both the low-fidelity and high-fidelity maps are simultaneously trained. By doing so, the optimization problem might result computationally demanding. To overcome this potential issue and to enhance the robustness of the method, we split the optimization procedure in two sequential tasks, namely the training of the low-fidelity map and of the high-fidelity one.

2.2.2 MPINN-v1 with non parametric low-fidelity

Winthin the MPINN-v1 approach, the training procedure is divided into two stages. In the first stage we train the low-fidelity ANN ($\mathcal{NN}_L$) as a map starting from snapshots $\{u(\mathbf{x}_i^L, t_i^L; \boldsymbol{\mu}_{\text{guess}})\colon i = 1, \dots, N_{\text{obs}}^L\}$, for a fixed $\boldsymbol{\mu}_{\text{guess}}$, which can be provided by a priori knowledge. The problem can be written as:

$$\begin{cases} \min\limits_{\mathbf{W}_L} \ \mathcal{J}_L(\mathbf{W}_L) \\ \text{s.t.} \quad u_L(\mathbf{x}, t) = \mathcal{NN}_L(\mathbf{x}, t; \mathbf{W}_L) \\ \text{where} \quad \mathcal{J}_L(\mathbf{W}_L) = \dfrac{\omega_{\text{fit}}}{N_{\text{obs}}^L} \displaystyle\sum_{i=1}^{N_{\text{obs}}^L} |Cu_L(\mathbf{x}_i^L, t_i^L) - y_i^L|^2 + R_L(u_L; \mathbf{W}_L), \end{cases} \tag{7}$$

where $y_i^L = Cu(\mathbf{x}_i^L, t_i^L; \boldsymbol{\mu}_{\text{guess}})$, $i = 1, \dots, N_{\text{obs}}^L$. Here, R_L could be chosen as done in (3), in a PINN framework, or as a standard ANN regularizer.

In the second stage, we solve the optimal control problem in the multi-fidelity framework, which is:

$$\begin{cases} \min\limits_{\boldsymbol{\mu}, \mathbf{W}_H} \ \mathcal{J}(\boldsymbol{\mu}, \mathbf{W}_H) \\ \text{s.t.} \quad u(\mathbf{x}, t) = u_L(\mathbf{x}, t) + \mathcal{NN}_H(\mathbf{x}, t, u_L(\mathbf{x}, t); \mathbf{W}_H) \\ \text{where} \quad \mathcal{J}(\boldsymbol{\mu}, \mathbf{W}_H) = \dfrac{\omega_{\text{fit}}}{N_{\text{obs}}^H} \displaystyle\sum_{i=1}^{N_{\text{obs}}^H} |Cu(\mathbf{x}_i^H, t_i^H) - y_i^H|^2 + R_H(u; \boldsymbol{\mu}, \mathbf{W}_H), \end{cases} \tag{8}$$

where $R_H = R_{\text{phys}}$ is given by the physics-informed regularizer (3).

2.2.3 MPINN-v2 with parametric low-fidelity

In the second approach, we first train the low-fidelity ANN ($\mathcal{NN}_L$) as a parametric map starting from snapshots $\{u(\mathbf{x}_i^L, t_i^L; \boldsymbol{\mu}_i^L)\colon i = 1, \dots, N_{\text{obs}}^L\}$. The problem can be written as:

$$\begin{cases} \min\limits_{\mathbf{W}_L} \ \mathcal{J}_L(\mathbf{W}_L) \\ \text{s.t.} \quad u_L(\mathbf{x}, t; \boldsymbol{\mu}) = \mathcal{NN}_L(\mathbf{x}, t; \boldsymbol{\mu}, \mathbf{W}_L), \\ \text{where} \quad \mathcal{J}_L(\mathbf{W}_L) = \dfrac{\omega_{\text{fit}}}{N_{\text{obs}}^L} \displaystyle\sum_{i=1}^{N_{\text{obs}}^L} |Cu_L(\mathbf{x}_i^L, t_i^L; \boldsymbol{\mu}_i^L) - y_i^L|^2 + R_L(u_L; \mathbf{W}_L) \end{cases} \tag{9}$$

Unlike the non-parametric approach of Sec. 2.2.2, which is associated with a given value of the parameter $\boldsymbol{\mu}_{\text{guess}}$, here the low-fidelity map encodes the dependence of the solution on the parameter $\boldsymbol{\mu}$. This information will be later leveraged, in the multi-fidelity formulation, to support the identification of the parameter $\boldsymbol{\mu}$.

This stage might be optionally followed by a second one, in which we provide a first estimate of the unknown parameters using the low-fidelity map, minimizing the distance between model and measurements. This would provide an initial guess for $\boldsymbol{\mu}$, employed in the following step as starting point of the training.

In the last stage, we solve the optimal control problem in the multi-fidelity framework, which is:

$$\begin{cases} \min\limits_{\boldsymbol{\mu},\mathbf{W}_H} & \mathcal{J}(\boldsymbol{\mu},\mathbf{W}_H) \\ \text{s.t.} & u(\mathbf{x},t) = u_L(\mathbf{x},t;\boldsymbol{\mu}) + \mathcal{NN}_H(\mathbf{x},t,u_L(\mathbf{x},t;\boldsymbol{\mu});\mathbf{W}_H). \\ \text{where} & \mathcal{J}(\boldsymbol{\mu},\mathbf{W}_H) = \dfrac{\omega_{\text{fit}}}{N_{\text{obs}}^H}\displaystyle\sum_{i=1}^{N_{\text{obs}}^H} |Cu(\mathbf{x}_i^H,t_i^H) - y_i^H|^2 + R_H(u;\boldsymbol{\mu},\mathbf{W}_H) \end{cases} \tag{10}$$

which is now constrained on a multi-fidelity solution, depending on the unknown parameters and the solution of the low-fidelity ANN. In this stage we employ the following regularization term:

$$R_H(u;\boldsymbol{\mu},\mathbf{W}_H) = R_{\text{phys}}(u;\boldsymbol{\mu},\mathbf{W}_H) + \frac{\omega_{\text{lh}}}{N_c}\sum_{i=1}^{N_c} |u(\mathbf{x}_i^c,t_i^c) - u_L(\mathbf{x}_i^c,t_i^c;\boldsymbol{\mu})|^2. \tag{11}$$

The first term of (11) is the standard physics-informed loss, while the second term leverages the information gained during the low-fidelity training (9), penalizing the discrepancy between the low-fidelity parametric map, evaluated in the candidate parameter $\boldsymbol{\mu}$, and the solution u. The hyperparameter $\omega_{\text{lh}} \geq 0$ tunes the weight of this term in the global loss. For simplicity, we evaluate this latter term in the same collocation points used to enforce the PDE; however, an independent set of points could be considered as well.

We refer to Fig. 1 for a visual representation of the PINN and MPINN approaches described in this Section.

3 Results

In this section we describe the numerical results related to two benchmark problems and to an application in the field of cardiac electrophysiology. In all the tests reported in this paper we set for simplicity $R_L \equiv 0$. The observation points $(\mathbf{x}_i^H, t_i^H)$ and $(\mathbf{x}_i^L, t_i^L)$, as well as the collocation points $(\mathbf{x}_i^c, t_i^c)$, are selected by means of a Monte Carlo sampling strategies from uniform distributions. Other sampling strategies, such as Latin Hypercube sampling, could be adopted. To mimic the presence of measurement error, we artificially generate the noise ϵ_i. Unless otherwise specified, we generate ϵ_i by sampling from independent Gaussian distributions with zero mean and standard deviation σ, that will be specified case by case. To minimize the loss function, we first run 100 epochs with the Adam optimizer [16]. Then we switch to a more computationally demanding, but more accurate optimizer, namely the BFGS method [13]. Unless otherwise specified, we considered in the benchmark examples a maximum number of 2000 iterations for this method.

To evaluate and compare the results obtained with the different strategies considered in this paper, we introduce the following metrics:

$$\begin{aligned} \mathcal{J}_{\text{fit}} &= \sqrt{\frac{1}{N_{\text{obs}}^H}\sum_{i=1}^{N_{\text{obs}}^H} |Cu(\mathbf{x}_i^H,t_i^H) - y_i^H|^2}, \\ \mathcal{J}_{\text{phys}} &= \sqrt{\frac{1}{N_c}\sum_{i=1}^{N_c} \left|\alpha\frac{\partial u(\mathbf{x}_i^c,t_i^c)}{\partial t} + \mathcal{L}(u(\mathbf{x}_i^c,t_i^c);\boldsymbol{\mu})\right|^2}, \\ \mathcal{J}_{\text{lh}} &= \sqrt{\frac{1}{N_c}\sum_{i=1}^{N_c} |u(\mathbf{x}_i^c,t_i^c) - u_L(\mathbf{x}_i^c,t_i^c;\boldsymbol{\mu})|^2}, \end{aligned} \tag{12}$$

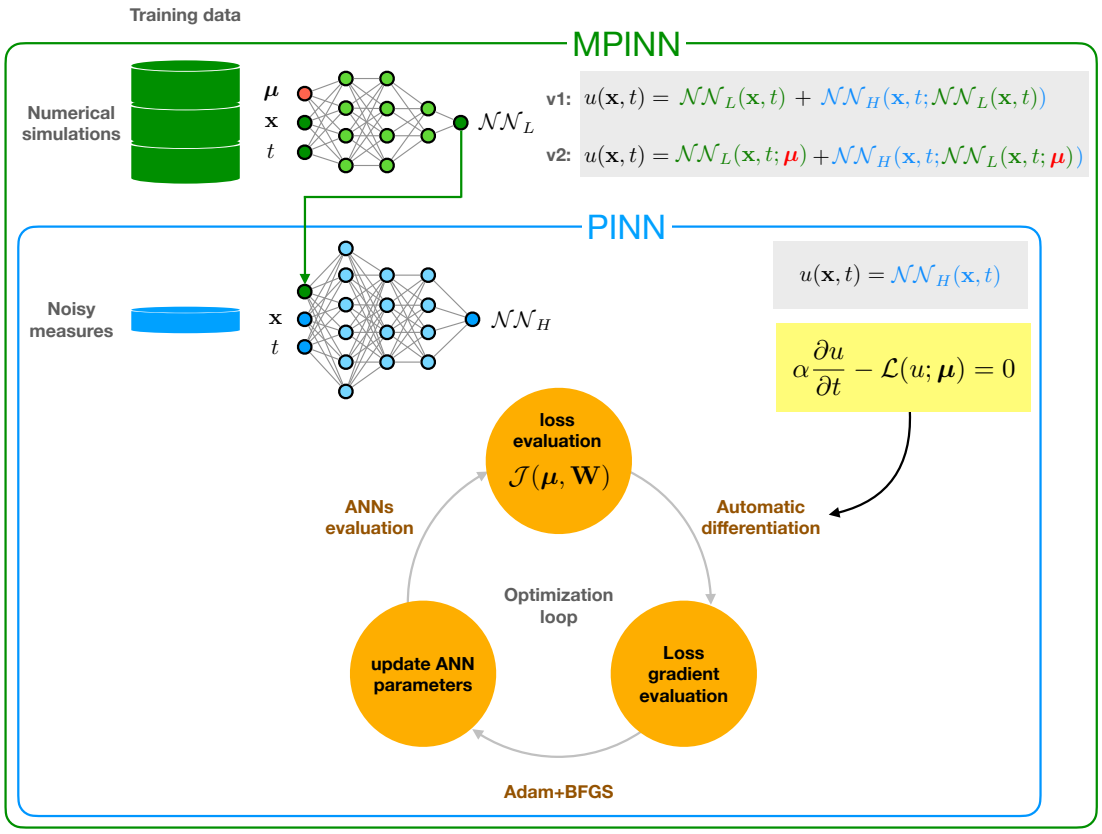

Figure 1: Training procedure of PINN and MPINN approaches (the dependence on the parameter vector distinguishes the two multi-fidelity versions).

corresponding to the root mean square (RMS) residuals, respectively associated with data fitting, with the satisfaction of the PDE and with the discrepancy between the numerical solution and the low-fidelity parametric map (the latter being defined for the MPINN-v2 approach only). We also define $\mathcal{J}_{\text{fit}}^{\text{test}}$ in a similar manner of $\mathcal{J}_{\text{fit}}$, but computed on a testing dataset, that is on points different than those employed to train the ANN. Finally, to evaluate the quality of the estimate of the parameters, we employ their relative error.

Every Machine Learning algorithm depends on a set of *hyperparameters*. In this context, besides the hyperparameters associated with the ANN architecture (number of layers and number of neurons per layer), the training process is controlled by scalar weight factors that tune the mutual contribution of the different components of the loss, that is ω_{fit}, ω_{phys} and (in the MPINN-v2 case) ω_{lh}. Note that the number of independent weight factors is equal to the total diminished by one, as we can fix without loss of generality e.g. $\omega_{\text{fit}} = 1$. In the tests reported in this paper, the weight factors were set according to the following strategy. In each test, we tuned the ANN architecture and the weight factors independently for each approach (PINN, MPINN-v0, MPINN-v1 and MPINN-v2) by trial and error. Once set, comparative analyses were performed for fixed values of the hyperparameters (their values are reported in the figures' captions or in the main text).

The results are obtained with the support of the Automatic Differentiation engine of TensorFlow [1]. To train the models, we employ the Adam optimizer of Keras [6] and the BFGS optimizer of SciPy [35].

3.1 Benchmark problems

In this section we introduce the two benchmark problems considered in this paper.

Advection-Diffusion-Reaction equation

We consider the following parametrized Advection-Diffusion-Reaction (ADR) problem: find $u = u(x)$ such that

$$-\mu\frac{d^2u(x)}{dx^2} + \gamma\frac{du(x)}{dx} - u(x) = f(x) \qquad x \in (0,1). \tag{13}$$

By choosing the forcing term as

$$f(x) = \frac{\gamma^2}{\sqrt{\mu}}\cos\left(\frac{x}{\sqrt{\mu}}\right)$$

the PDE is satisfied by the solution

$$u_{ex}(x) = \gamma\sin\left(\frac{x}{\sqrt{\mu}}\right).$$

Here the parameter vector reads $\boldsymbol{\mu} = (\mu, \gamma)^T$. The exact value of the parameter vector, employed to generate the training data, is $\boldsymbol{\mu}_{ex} = (0.25, 1)^T$. In the MPINN-v2 setting, for the construction of the parametric map, we sample the parameter in the set $\boldsymbol{\mu} \in [0.1, 2.0] \times [0.5, 3.0]$. In the PINN setting, we employ an ANN with three hidden layers of 10 neurons each. In the MPINN-v1 (respectively, MPINN-v2) setting, we employ one (respectively, three) hidden layer for $\mathcal{NN}_L$, and one (respectively, two) hidden layer for $\mathcal{NN}_H$, with 10 neurons per each layer. In the MPINN-v0 setting, we use the same architecture for $\mathcal{NN}_L$ and $\mathcal{NN}_H$ of the MPINN-v1 setting. The richer architecture of the low-fidelity ANN in the MPINN-v2 compared to the MPINN-v1 case is due to the need of including the parametric dependence into the low-fidelity model.

Parabolic equation

As a second benchmark problem, we consider the time-dependent PDE

$$\frac{du(x,t)}{dt} - \mu \frac{d^2u(x,t)}{dx^2} + \gamma\, x^2\, \frac{du(x,t)}{dx} = 0 \qquad x \in (-1,1),\, t \in (0,1), \tag{14}$$

which is satisfied by the solution

$$u_{ex}(x,t) = \exp\left(-\sqrt{\gamma\mu}\left(\frac{x^2}{2\mu} + t\right)\right).$$

The true value of the parameter vector $\boldsymbol{\mu} = (\mu, \gamma)^T$ is given by $\boldsymbol{\mu}_{ex} = (1,2)^T$. To construct the parametric map in the MPINN-v2 setting, we sample in the set $\boldsymbol{\mu} \in [0.1, 4.0] \times [0.1, 4.0]$. For this benchmark test, we employ a three hidden layers architecture in the PINN setting, and a two hidden layers architecture for both ANNs in the MPINN settings, with 10 neurons per each layers.

3.1.1 PINN

We first consider the PINN method, without a multi-fidelity approach. In Figures 2 and 3 we investigate the dependency of the accuracy of the solution with respect to the number of measurements N^H_{obs}. The trend is plotted for different choices of the noise magnitude σ and of the number of collocation points N_c. The test set comprises 1000 samples of the solution associated with the true values of the parameters in points randomly selected with a Monte Carlo algorithm. For the sake of brevity we only consider the ADR test case in these tests. Let us first consider the solutions obtained in the absence of noise ($\sigma = 0$). We notice that with only 4 observations both parameters are reconstructed accurately (relative error of the order of $10^{-5} \div 10^{-6}$). By increasing N^H_{obs} further, the reconstruction error slightly decreases. The number of collocation points impacts the parameter estimate, which is SPslightly more accurate for $N_c = 10000$ rather than for $N_c = 100$. The results show that the quality of the solution is heavily impacted by noise. For $\sigma = 0.005$, in fact, the error rises to the order of $10^{-1} \div 10^{-2}$. In the following sections we will show how the multi-fidelity approach yields an improvement in the quality of parameter estimation when considering noisy measurements.

3.1.2 MPINN-v0 and MPINN-v1

We now consider the MPINN-v1 method and compare it with the PINN and MPINN-v0 methods. In Figs. 4, 5, 6 and 7 we show, for the ADR test case, the evolution of the relative errors in the estimate of the parameters with respect either to the number of optimization epochs or with respect to the computational time. In Fig. 8, a similar test is performed for the parabolic problem. In all test cases, we consider $N^L_{\text{obs}} = 100$ points for the training of the low-fidelity ANN. Since the outcome of the training is aleatory (due to the randomness of the ANN weights initialization), we run 10 simulations for each setting, and we represent the geometric mean of the results by a solid line, and by a shaded area the region spanned by all the trajectories. In the figures, we consider different values for N^H_{obs} and σ (see captions).

In all the considered cases, the MPINN-v0, MPINN-v1 and PINN methods provide comparable results, on average, in terms of accuracy of the final estimation. The MPINN-v1 approach, however, is capable of providing an improvement in the convergence speed. In Fig. 4, the loss function traces are plotted against the number of iterations. However, the computational effort per iteration may differ from one method to another. Thus, to perform a fair comparison, in Fig. 5 we report the same traces shown in Fig. 4, plotted with respect to the computational time rather then the number

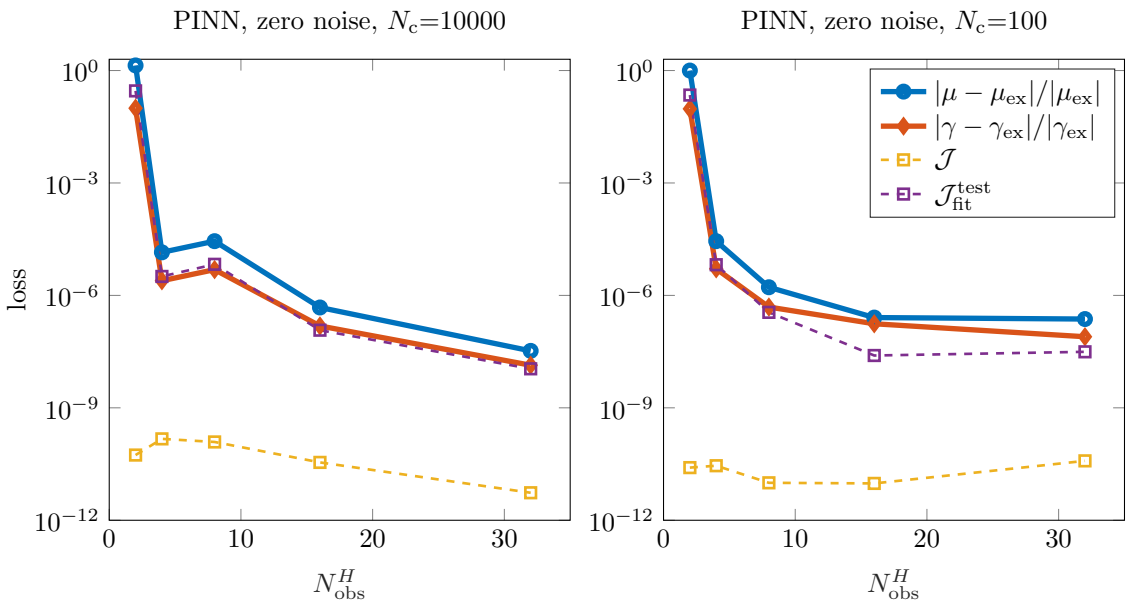

Figure 2: ADR test case: relative error in the estimate of the parameters μ and γ with respect to N^H_{obs}. The two figures differ in the number of colloction points used: $N_c = 10000$ (left) and $N_c = 100$ (right). Here, we consider $\omega_{\text{fit}} = 100$, $\omega_{\text{phys}} = 1$ and 1000 test points employed in the computation of $\mathcal{J}^{\text{test}}_{\text{fit}}$.

of iterations. By doing so, the advantage in terms of convergence speed of the MPINN-v1 method compared to the the PINN and the MPINN-v0 ones is even more noticeable. We remark that the computational times considered in the figure take into account also the cost needed to train the low fidelity network in the MPINN-v1 case. Therefore, from here on we report only loss trends with respect to the number of iterations. In fact, a comparison based on the computational time instead of the number of iterations would be more advantageous for the MPINN-v1 method, thanks to having split the training of the low fidelity network from the high-fidelity one, which makes the method efficient and robust. Indeed, MPINN-v1 trajectories appear to be more likely to escape the local minima in which the PINN and MPINN-v0 trajectories are trapped, especially in the case of noisy data. This affects the height of shaded areas: see, e.g., the left column of Fig. 6 or the right column of Fig. 7). In the parabolic case, this mechanism provides also a modest reduction of the mean error (solid line in Fig. 8). As a matter of fact, the MPINN-v1 approach does not modify the loss function with respect to the standard PINN approach. Nonetheless, it modifies the architecture of the ANN representing the numerical solution: while in the PINN approach the solution is represented by a single ANN, agnostic of the problem at hand, in the MPINN-v1 approach it is given by the sum of first term (the low-fidelity solution) plus a correction term. In case the low-fidelity solution is correlated to the exact solution, this provides a significant speedup in the training process. Clearly, the closer the low-fidelity solution to the exact one, the larger the speed up. This is confirmed by the numerical results: in fact, for each of the above settings, we consider two different guesses for the parameter $\boldsymbol{\mu}_{\text{guess}}$. These mimic two possible scenarios: in the first one, we suppose to have some a priori knowledge on the real values of the parameters, while in the second one we do not have any prior information and consequently we use an uninformed guess. For the sake of fairness, we employ

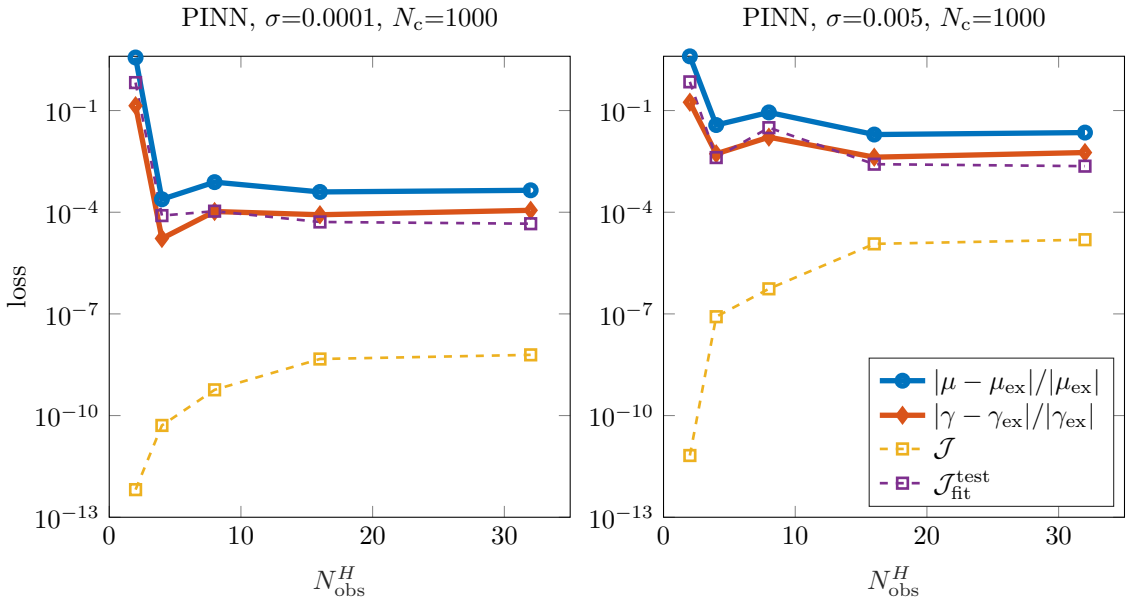

Figure 3: ADR test case with noise: relative error in the estimate of the parameters μ and γ with respect to N^H_{obs}. The two figures differ in the variance of the noise: $\sigma = 0.0001$ (left) and $\sigma = 0.005$ (right). Here, we consider $\omega_{fit} = 1$, $\omega_{phys} = 1$ and 1000 test points employed in the computation of $\mathcal{J}^{test}_{fit}$.

$\boldsymbol{\mu}_{guess}$ itself as initial guess in both the PINN and MPINN optimization problem. The results show that when $\boldsymbol{\mu}_{guess}$ is close to the exact solution, the optimizer converges more rapidly.

3.1.3 MPINN-v2

Let us now compare the results obtained with the MPINN-v2 method to those obtained with the PINN method. In this section, we focus on assessing the accuracy of MPINN-v2 method instead of the convergence speed since it is the distinguishing feature of this method. We consider three different cases: observation without noise, observation with Gaussian noise, observation with a noise that is nonlinearly correlated with the solution.

Case 1: no noise First, we consider the case without noise. In Fig. 9 we compare the trajectories of the different components of the loss function during the training process, for the ADR test case with $N^H_{obs} = 8$. The figure shows that, in the case considered here, the MPINN-v2 method is not better than the standard PINN method, other than in speeding up the early stages of the training process (as long as the weight ω_{lh} is sufficiently large). In fact, while with the MPINN-v2 method the loss values quickly stagnate on constant values, the PINN method is able to improve the quality of the solution by increasing the number of epochs.

This result is consistent with Fig. 10, in which we show the final error obtained in the estimate of the parameters with respect to N^H_{obs}. We observe that, while for a sufficiently large dataset the PINN method provides more accurate results, in the low-data regime the MPINN-v2 method is advantageous. In other terms, when the data provide little information, the multi-fidelity structure

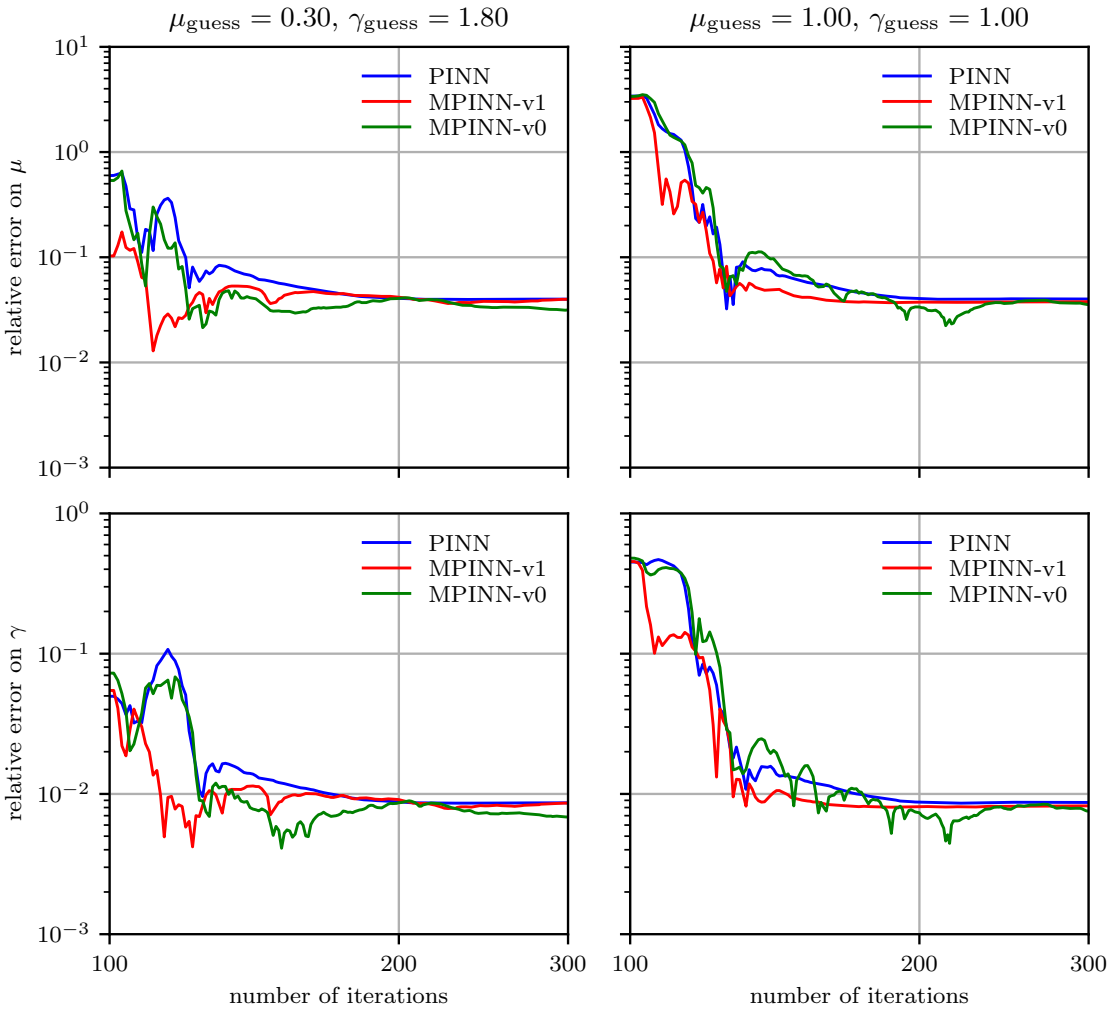

Figure 4: ADR test case with noise: relative error in the estimate of the parameters μ and γ with respect to number of epochs, for $N_{\text{obs}}^{H} = 16$ and $\sigma = 0.01$. The blue line refers to the PINN method, the red line to the MPINN-v2 method and the green one to the MPINN-v0 method. In the left column, the guess for the parameter (i.e. $\boldsymbol{\mu}_{\text{guess}} = (0.3, 1.8)^T$) is closer to the exact solution (i.e. $\boldsymbol{\mu}_{ex} = (0.5, 2)^T$) than the one of the right column (i.e. $\boldsymbol{\mu}_{\text{guess}} = (1, 1)^T$). The figure shows the results of 10 training processes (the shaded region is the area spanned by the trajectories; the solid line represents the geometric mean). Here, we consider $N_c = 1000$, $\omega_{\text{fit}} = 100$, $\omega_{\text{phys}} = 1$.

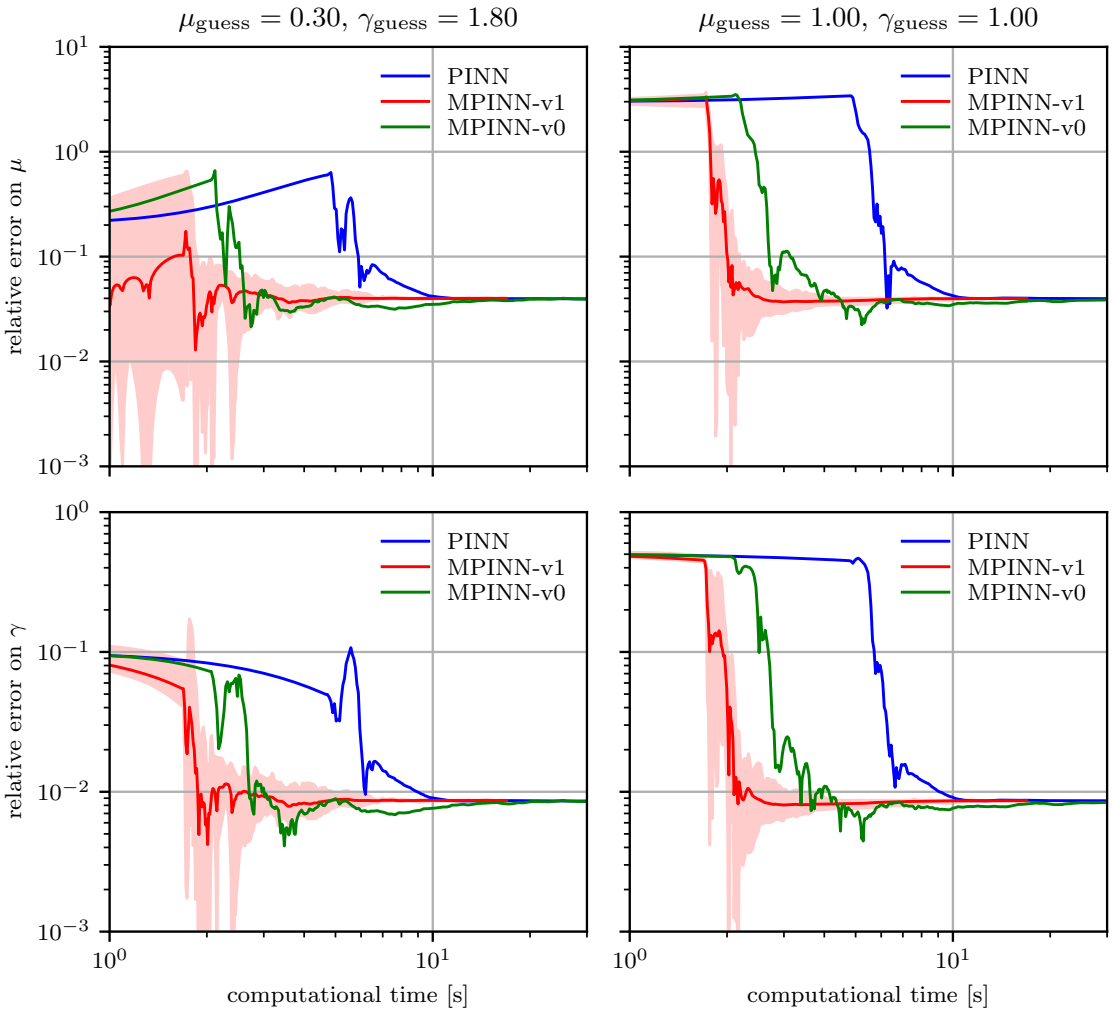

Figure 5: ADR test case with noise: relative error in the estimate of the parameters μ and γ with respect to computational time, for $N^H_{\text{obs}} = 16$ and $\sigma = 0.01$. The blue line refers to the PINN method, the red line to the MPINN-v2 method and the green one to the MPINN-v0 method. In the left column, the guess for the parameter (i.e. $\boldsymbol{\mu}_{\text{guess}} = (0.3, 1.8)^T$) is closer to the exact solution (i.e. $\boldsymbol{\mu}_{ex} = (0.5, 2)^T$) than the one of the right column (i.e. $\boldsymbol{\mu}_{\text{guess}} = (1, 1)^T$). The figure shows the results of 10 training processes (the shaded region is the area spanned by the trajectories; the solid line represents the geometric mean). Here, we consider $N_c = 1000$, $\omega_{\text{fit}} = 100$, $\omega_{\text{phys}} = 1$.

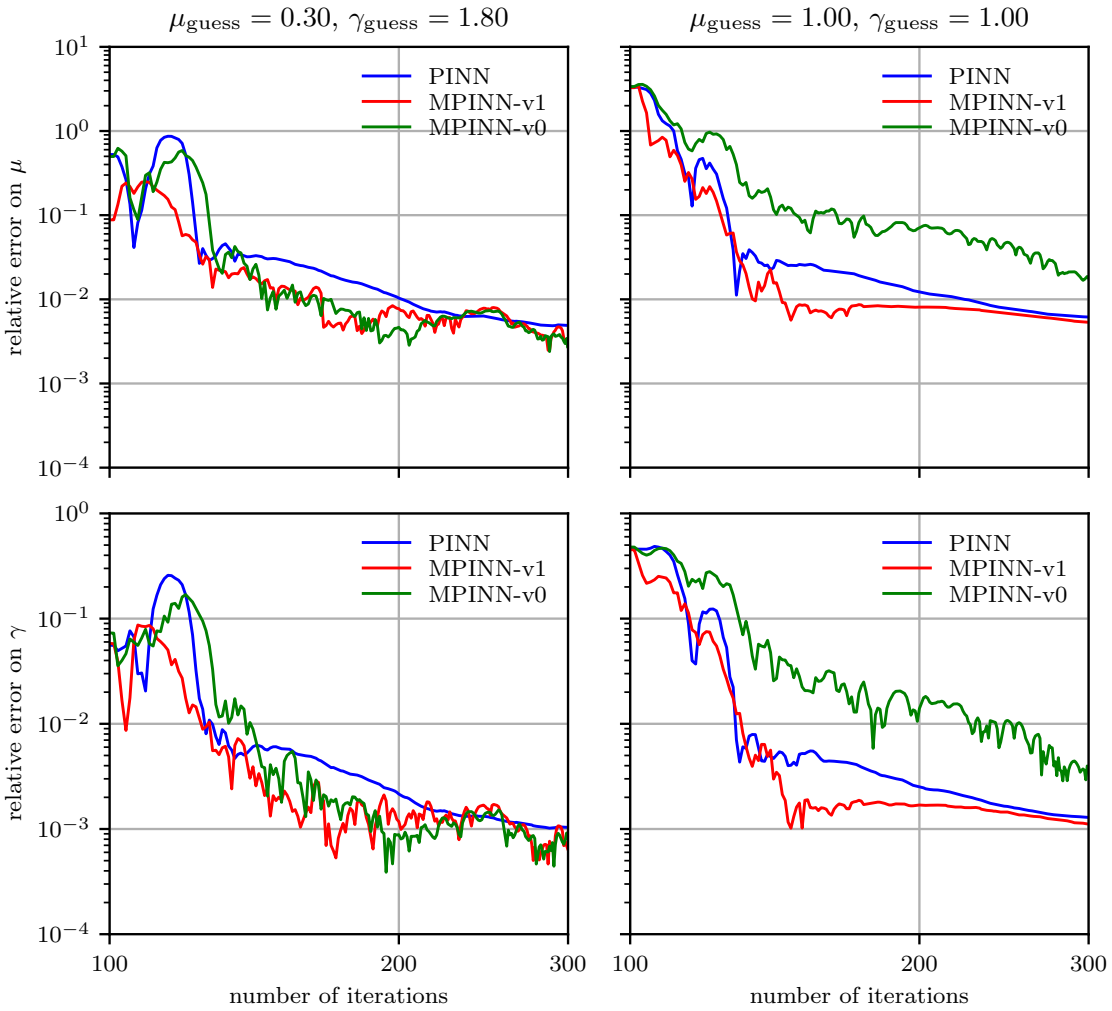

Figure 6: ADR test case with noise: relative error in the estimate of the parameters μ and γ with respect to number of epochs, for $N^H_{\text{obs}} = 16$ and $\sigma = 0.001$. The blue line refers to the PINN method, the red line to the MPINN-v2 method and the green one to the MPINN-v0 method. In the left column, the guess for the parameter (i.e. $\boldsymbol{\mu}_{\text{guess}} = (0.3, 1.8)^T$) is closer to the exact solution (i.e. $\boldsymbol{\mu}_{ex} = (0.5, 2)^T$) than the one of the right column (i.e. $\boldsymbol{\mu}_{\text{guess}} = (1, 1)^T$). The figure shows the results of 10 training processes (the shaded region is the area spanned by the trajectories; the solid line represents the geometric mean). Here, we consider $N_c = 1000$, $\omega_{\text{fit}} = 10$, $\omega_{\text{phys}} = 1$.

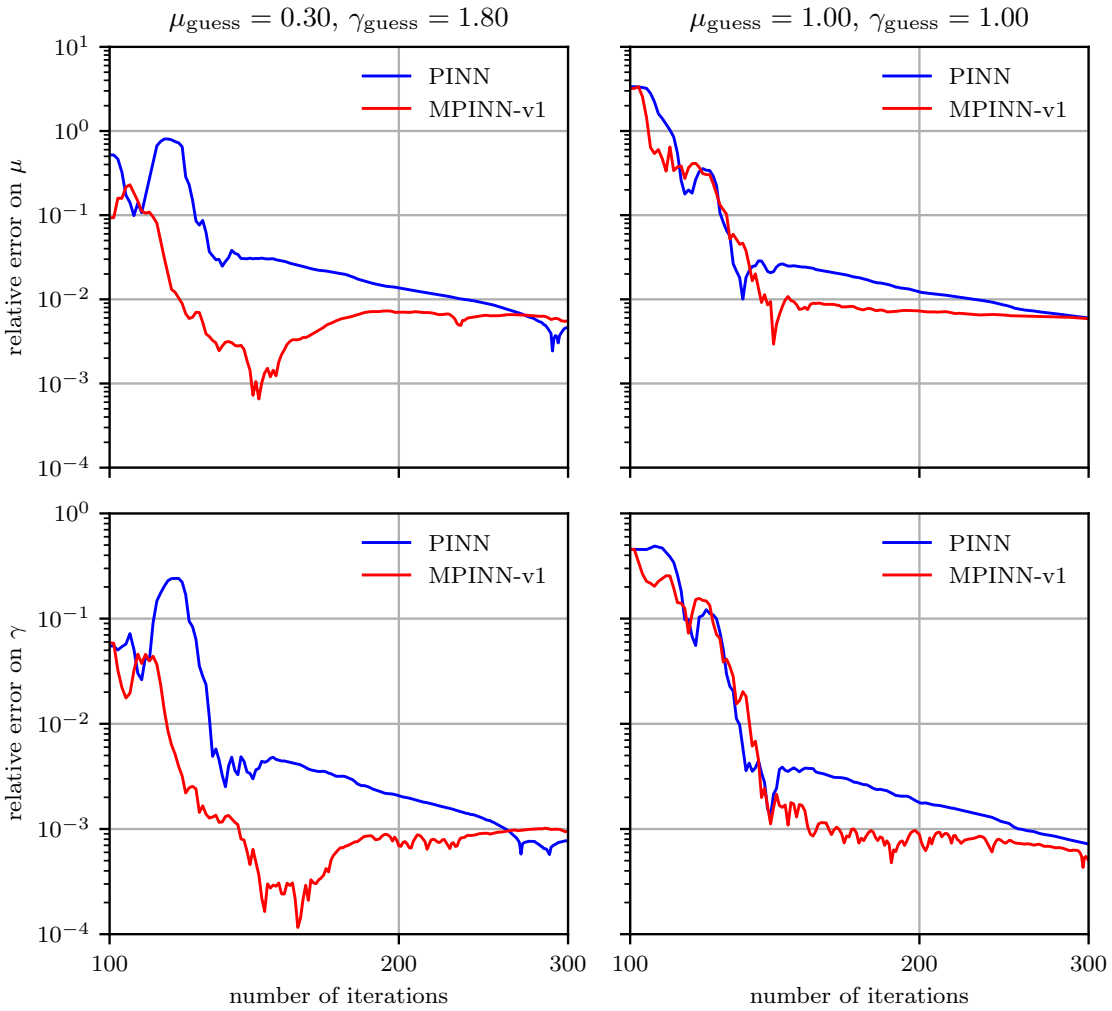

Figure 7: ADR test case with noise: relative error in the estimate of the parameters μ and γ with respect to number of epochs, for $N_{\text{obs}}^H = 8$ and $\sigma = 0.001$. The blue line refers to the PINN method, the red line to the MPINN-v1 method. In the left column, the guess for the parameter (i.e. $\boldsymbol{\mu}_{\text{guess}} = (0.3, 1.8)^T$) is closer to the exact solution (i.e. $\boldsymbol{\mu}_{ex} = (0.5, 2)^T$) than the one of the right column (i.e. $\boldsymbol{\mu}_{\text{guess}} = (1, 1)^T$). The figure shows the results of 10 training processes (the shaded region is the area spanned by the trajectories; the solid line represents the geometric mean). Here, we consider $N_c = 1000$, $\omega_{\text{fit}} = 10$, $\omega_{\text{phys}} = 1$.

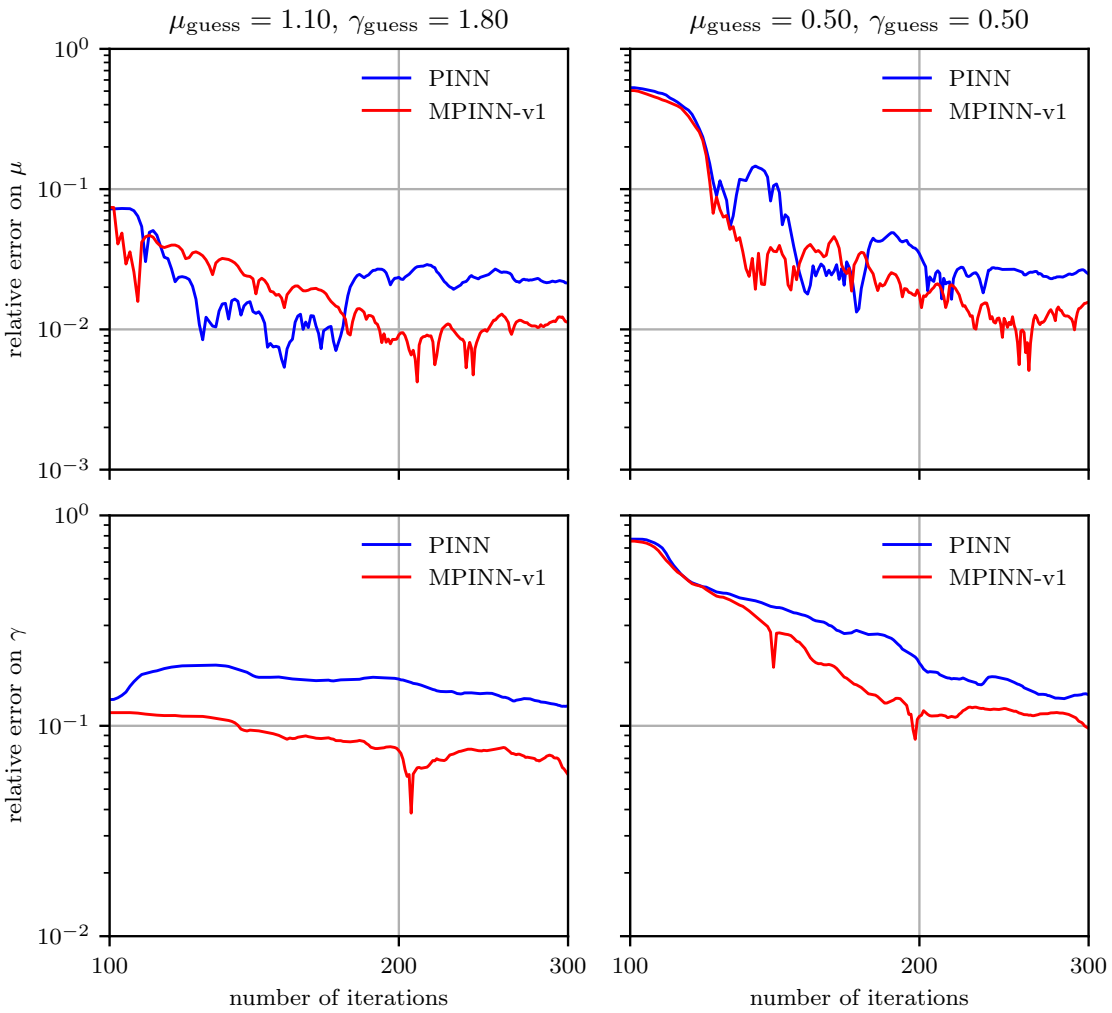

Figure 8: Parabolic test case with noise: relative error in the estimate of the parameters μ and γ with respect to number of epochs, for $N_{\text{obs}}^H = 16$ and $\sigma = 0.001$. The blue line refers to the PINN method, the red line to the MPINN-v1 method. In the left column, the guess for the parameter (i.e. $\boldsymbol{\mu}_{\text{guess}} = (1.1, 1.8)^T$) is closer to the exact solution (i.e. $\boldsymbol{\mu}_{ex} = (1, 2)^T$) than the one of the right column (i.e. $\boldsymbol{\mu}_{\text{guess}} = (0.5, 0.5)^T$). The figure shows the results of 10 training processes (the shaded region is the area spanned by the trajectories; the solid line represents the geometric mean). Here, we consider $N_c = 10000$, $\omega_{\text{fit}} = 1$, $\omega_{\text{phys}} = 1$.

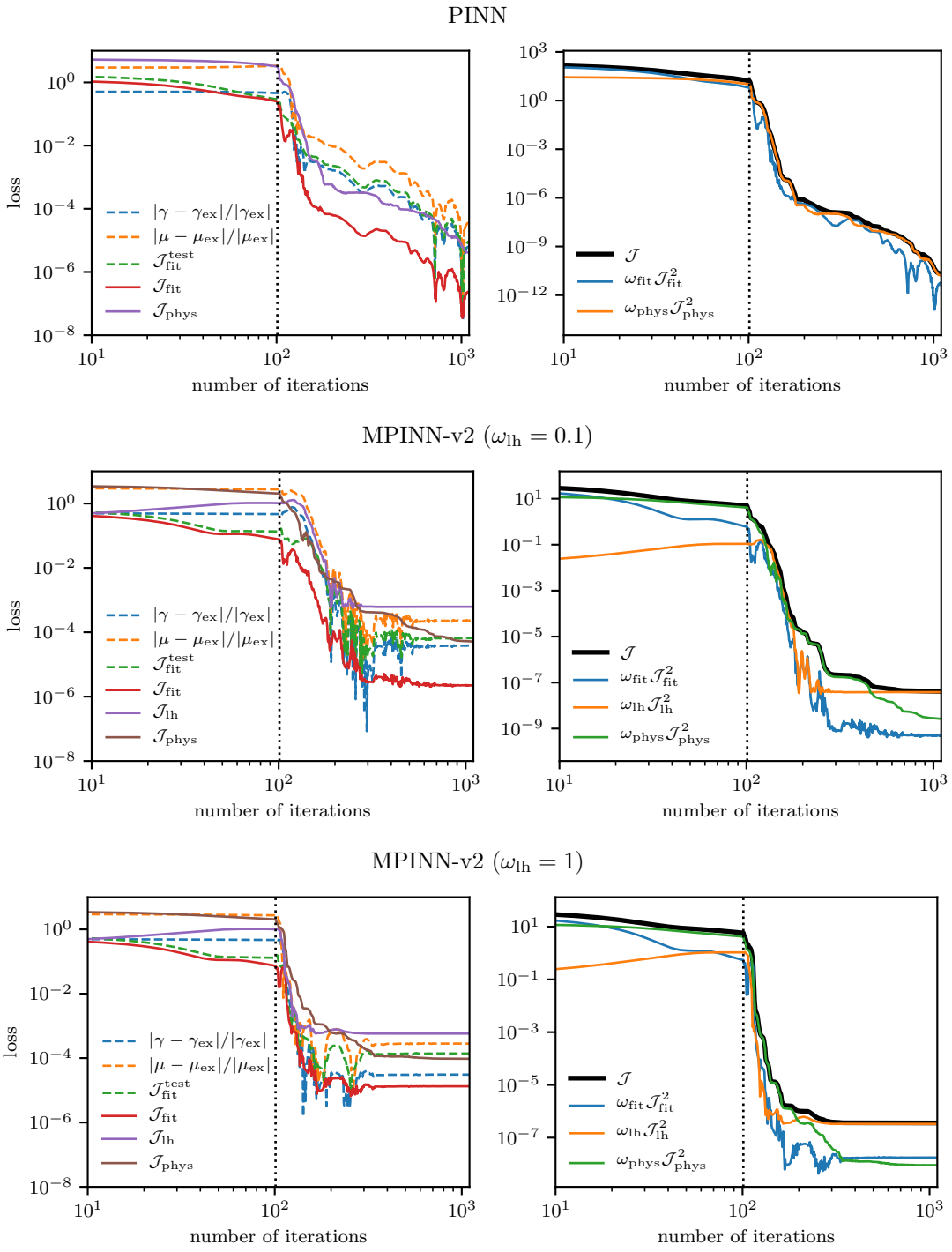

Figure 9: ADR test case: loss components and relative errors trend with respect to the epochs in the PINN model (top) vs the MPINN-v2 model (bottom), for $N_{obs}^H = 8$ measured points and without noise. Here, we consider $N_c = 1000$, $\omega_{fit} = 100$, $\omega_{phys} = 1$.

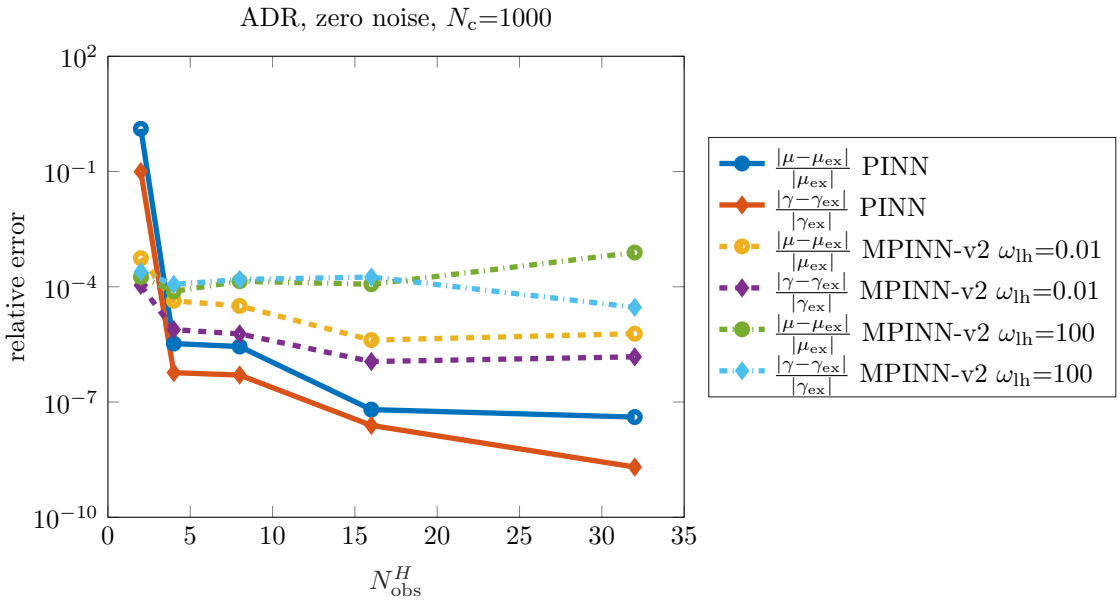

Figure 10: ADR test case: parameter estimation relative errors with exact measurements. We consider PINN vs MPINN-v2 trained using $N^L_{obs} = 10000$ and two different values of ω_{lh}. Here, we consider $\omega_{fit} = 100$ and $\omega_{phys} = 1$.

successfully leverages the information contained in the low-fidelity parametric map, thus improving the quality of the results.

Case 2: Gaussian noise We now consider the case with $\sigma = 0.01$ and we compare the PINN method with the the MPINN-v2 method. In Fig. 11 we show the loss components versus training epochs for the ADR problem. We test the PINN method versus the MPINN-v2 method for two different values of ω_{lh}, in the case $N^H_{obs} = 8$. By comparing the results with those of Fig. 9, we evince that the MPINN-v2 method is much more robust to noise than the standard PINN method. As a matter of fact, the former approach achieves a significantly higher accuracy than the latter in the case of noisy measurements. Moreover, a larger value of ω_{lh} has the double beneficial effect of accelerating convergence and improving accuracy.

In order to simulate scenarios with low-fidelity models of different accuracy, we show in Fig. 12 the loss components trajectories obtained with different values of N^L_{obs}. We observe that using only $N^L_{obs} = 100$ samples to build the parametric map, the parameter estimation is very accurate. Increasing N^L_{obs} to 1000 and to 10000 samples, the quality of the estimate slightly increases.

Then, we study the relative error with respect to the exact parameters when $N^H_{obs} = 2, 4, 8, 16, 32$ noisy measurements are considered (see Fig. 13). Unlike in the zero noise case, where the MPINN-v2 provides better estimates only in the low-data regime, our multi-fidelity is advantageous both with few and with many data affected by noise. As a matter of fact, we observe an error reduction up to two orders of magnitude in the parameters estimation compared with the standard PINN method. Finally, Fig. 14 shows tha same analysis of Fig. 13 for the parabolic problem, in the case of noisy data. The results show that the MPINN-v2 is advantageous also in this test case.

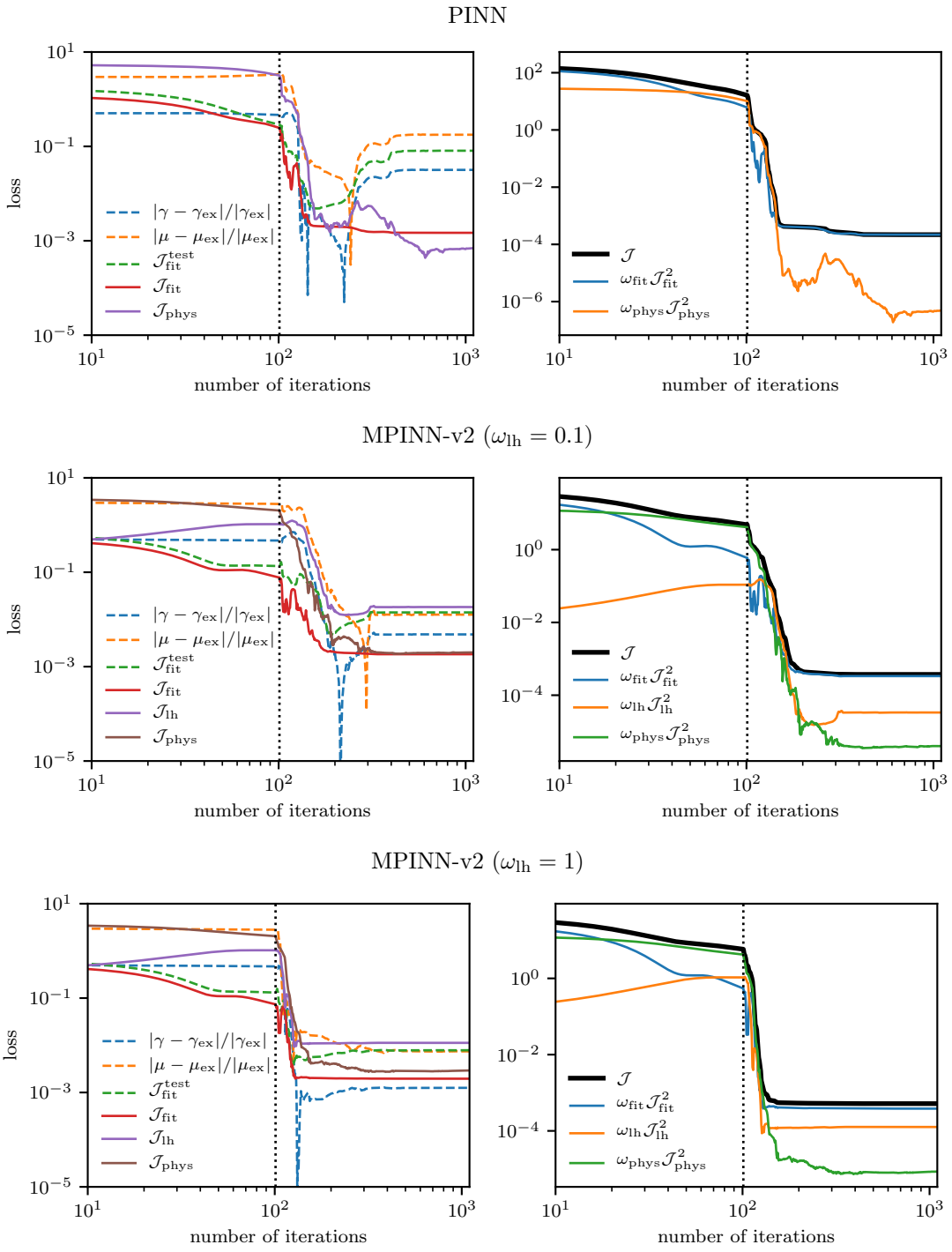

Figure 11: ADR test case with noise: loss components and relative errors trend with respect to the epochs in the PINN model (top) vs the MPINN-v2 model (bottom), for $N_{obs}^H = 8$ measured points and $\sigma = 0.01$. Here, we consider $N_c = 1000$, $\omega_{fit} = 100$, $\omega_{phys} = 1$.

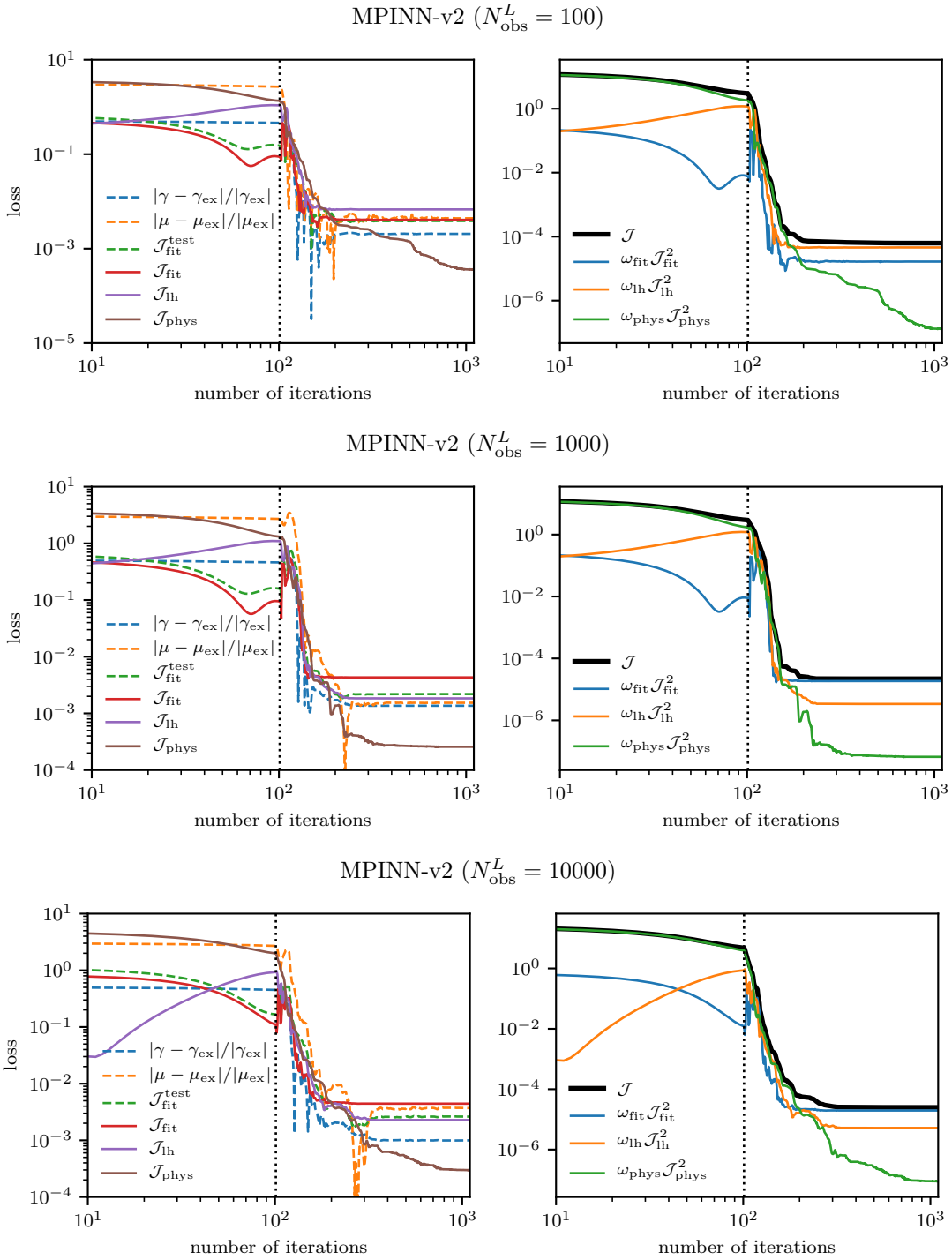

Figure 12: Loss components and relative errors trend with respect to the epochs in MPINN-v2 model with Gaussian noise on data ($\sigma = 0.01$, $N^H_{\text{obs}} = 8$). Here, we consider $N_c = 1000$, $\omega_{\text{fit}} = 1$, $\omega_{\text{phys}} = 1$, $\omega_{\text{lh}} = 1$.

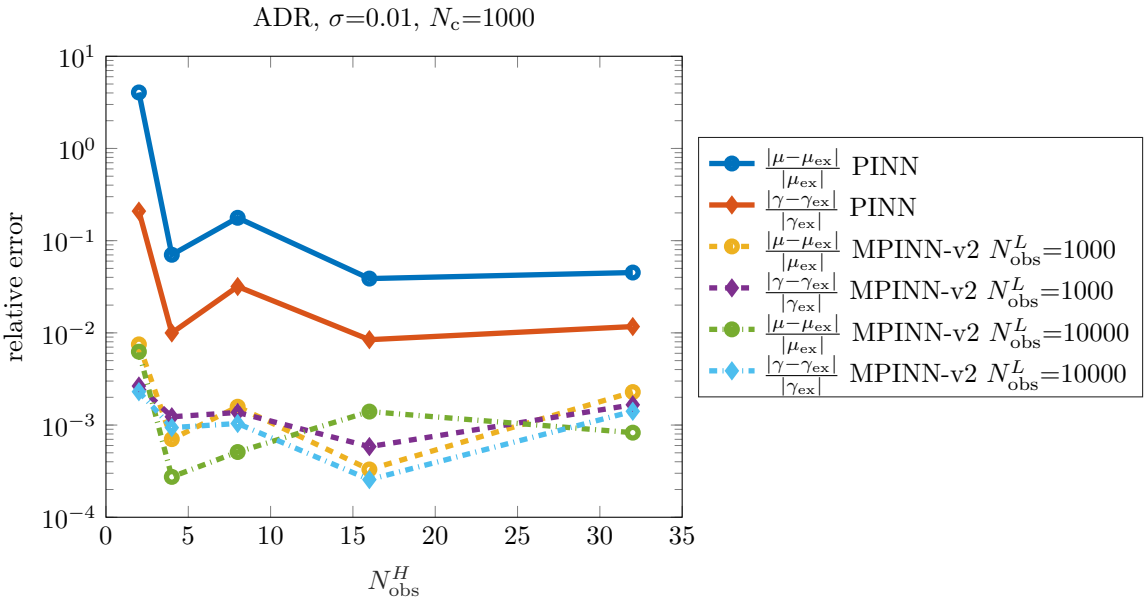

Figure 13: ADR test case with noise: parameter estimation relative errors, for PINN vs MPINN-v2 trained using $N^L_{\text{obs}} = 1000$ and $N^L_{\text{obs}} = 10000$, respectively. Here, we consider $N_c = 1000$, $\omega_{\text{fit}} = 1$, $\omega_{\text{phys}} = 1$ and $\omega_{\text{lh}} = 1$.

Case 3: correlated noise As a third case, we consider an observation error that is correlated (in a nonlinear manner) to the solution, according to

$$y(\mathbf{x}) = u_{\text{ex}}(\mathbf{x}; \boldsymbol{\mu}_{ex}) + 0.2\cos(u_{\text{ex}}(\mathbf{x}; \boldsymbol{\mu}_{ex})).$$

Also in this case, MPINN outperforms the PINN method (see Figure 15). The additive nonlinear component is able to learn the mismatch between the observation and the low-fidelity parametric map, thus correcting and mitigating the effect of noise.

3.2 Application to cardiac electrophysiology

We now consider an application of the proposed methods to a problem arising in the field of cardiac electrophysiology. The Bueno-Orovio model [5] (henceforth denoted as BO model) consists of a set of strongly nonlinear ordinary differential equations, that describe the ionic dynamics of the cardiac cells. It includes four state variables, respectively describing the transmembrane potential (u) and three *gating variables* (v, w, s) associated with the fraction of open ion channels across the cell

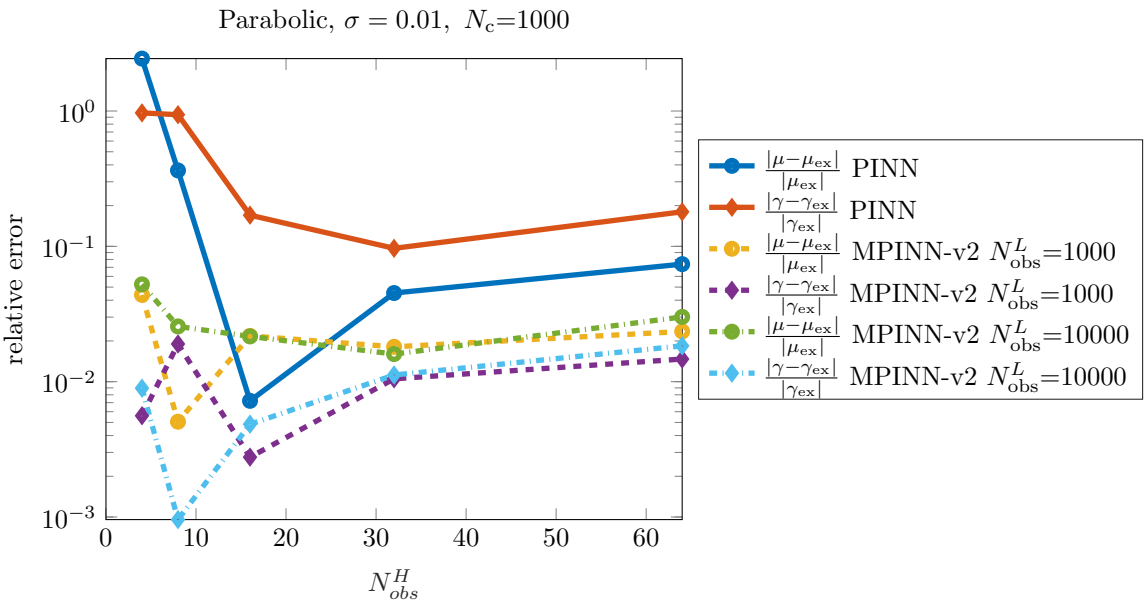

Figure 14: Parabolic problem with noise: parameter estimation relative errors, for PINN vs MPINN-v2 trained using $N^L_{\text{obs}} = 1000$ and $N^L_{\text{obs}} = 10000$, respectively. Here, we consider $N_c = 10000$, $\omega_{\text{fit}} = 1$, $\omega_{\text{phys}} = 1$ and $\omega_{\text{lh}} = 1$.

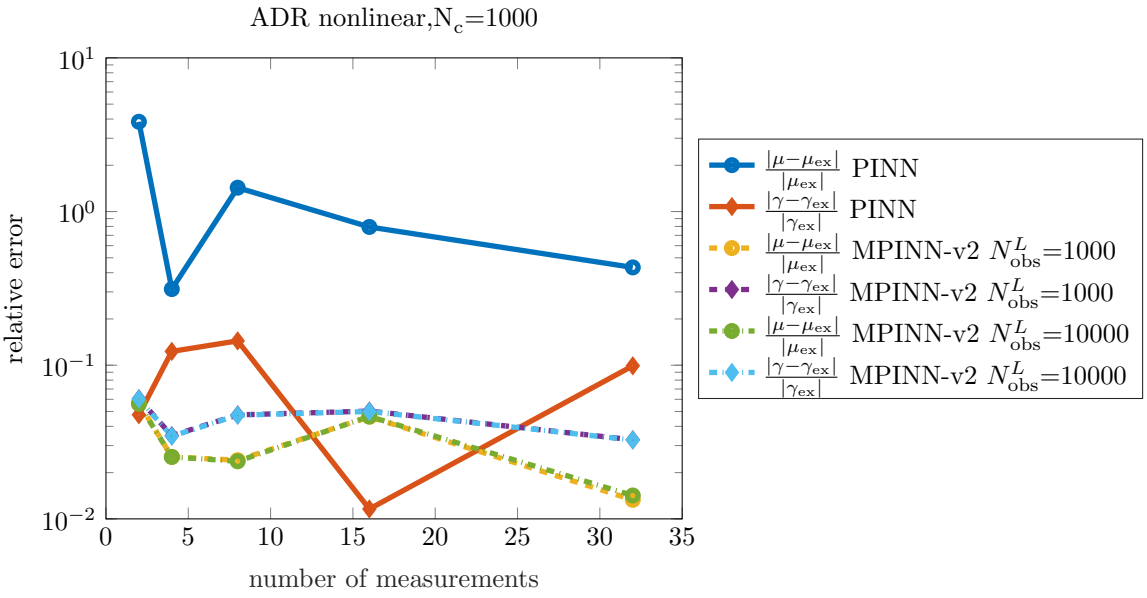

Figure 15: ADR test case with correlated noise: parameter estimation relative errors, for PINN vs MPINN-v2 trained using $N_{lf} = 1000$ and $N_{lf} = 10000$. Here, we consider $N_c = 1000$, $\omega_{\text{fit}} = 1$, $\omega_{\text{phys}} = 1$ and $\omega_{\text{lh}} = 1$.

membrane. The BO model reads as follows.

$$\begin{cases} \dfrac{du}{dt} = -(I_{\text{fi}} + I_{\text{so}} + I_{\text{si}}) + I_{\text{app}} \\ \dfrac{dv}{dt} = \dfrac{(1 - H(u-\theta_v))(v_\infty - v)}{\tau_v^-} - \dfrac{H(u-\theta_v)v}{\tau_v^+} \\ \dfrac{dw}{dt} = \dfrac{(1 - H(u-\theta_w))(w_\infty - w)}{\tau_w^-} - \dfrac{H(u-\theta_w)w}{\tau_w^+} \\ \dfrac{ds}{dt} = \dfrac{1}{\tau_s}\left(\dfrac{1+\tanh(k_s(u-u_s))}{2} - s\right). \end{cases} \tag{15}$$

Here $H(x-x_0)$ refers to the Heaviside function centered at x_0. The functions $I_{\text{fi}}, I_{\text{so}}$ and I_{si} represent the fast inward, slow outward and slow inward electric currents respectively, given by the following expressions:

$$I_{\text{fi}} = -\frac{vH(u-\theta_v)(u-\theta_v)(u_u-u)}{\tau_{\text{fi}}}, \tag{16}$$

$$I_{\text{so}} = \frac{(u-u_o)(1-H(u-\theta_w))}{\tau_o} + \frac{H(u-\theta_w)}{\tau_{\text{so}}}, \tag{17}$$

$$I_{\text{si}} = -\frac{H(u-\theta_w)ws}{\tau_{\text{si}}}. \tag{18}$$

The function $I_{\text{app}}(t)$ represents a prescribed applied current, needed to trigger the cellular excitation. The BO model permits to numerically reproduce a wide range of experimental conditions. However,

in practical applications, its parameters need to be suitably calibrated on a patient-specific basis. Due to the large number of parameters, one typically fixes most of them, based on literature data, and calibrates only the few most relevant ones. Specifically, in this paper we focus on the calibration of the parameter τ_{fi}, associated with the time constant of the fast inward current I_{fi}. The parameter τ_{fi} influences the action potential duration, that is the time lag between the depolarization and polarization phases of u. Hence, in this section we consider the problem of estimating τ_{fi} from noisy measurements of the transmembrane potential u and few measurements of the gating variables v, w and s.

In realistic scenarios, the applied current $I_{\text{app}}(t)$ is not easily measurable. Nonetheless, it plays a role only in the very first instants (order of 1 ms) of the transmembrane potential and it is typically set to zero for the remaining time. Hence, in the physical loss of Eq. (3), we set $I_{\text{app}} \equiv 0$ and we do not include time instants before $t = 2\,\text{ms}$.

To generate the training data, we use a numerical solver based on the Finite Difference method approximating the solution of the BO model. The solver uses a time step of $\Delta t = 0.1\,\text{ms}$ and a final time $T = 0.8\,\text{s}$. The resulting transmembrane potential solution u is sub-sampled at 25 ms intervals, and noise is artificially added by sampling from a normal distribution of zero mean and standard deviation σ. In addition to measurements on the transmembrane potential, we assume that we also have measurements of the gating variables v, w and s. Since, however, measurements of these variables are more difficult to obtain in practice, we here consider a much coarser sampling (one data point every 0.3 s). This is precisely the set of available measures adopted in the estimation of the parameter τ_{fi}.

Unlike the test cases considered above, the BO model is a vectorial problem. Hence, the losses defined in Eq. (12) can be independently defined for each component of Eq. (15). In what follows, we will denote the corresponding loss components with a superscript specifying the variable. Similarly, we identify with a superscript the associated weights. For instance, we denote by $\mathcal{J}^u_{\text{phys}}$ and $\mathcal{J}^v_{\text{phys}}$ the loss functions corresponding to the residual associated with the first line and second line of Eq. (15), respectively. The associated weights are respectively denoted by ω^u_{phys} and ω^v_{phys}. Similar notations apply to the other variables.

In the PINN setting we employ an ANN with three hidden layers, with 32, 24 and 16 neurons, respectively. In the MPINN setting, the same architecture is used for $\mathcal{NN}_H$, while 32, 16 and 8 neurons are employed for $\mathcal{NN}_L$. Low fidelity dataset comprises 75 numerical simulations subsampled with random interval smaller than 10 ms.

3.2.1 Training strategy

Due to the nonlinearity of the BO model equations, a one-shot training that starts from a random initialization of the ANN parameters typically leads to results far from the global minimum. As a matter of fact, if the initial guess for the solution is far from the exact one, the physical component of the loss function, instead of helping in the identification of the unknown parameter, leads the estimate of the parameter to diverge. Therefore, in this paper we propose a training divided into different stages, in which we tune differently the relative contribution of the different loss components.

Specifically, we split the training into two stages. In both stages, we set $\omega^u_{\text{fit}} = 10^{-2}$ and $\omega^v_{\text{fit}} = \omega^w_{\text{fit}} = \omega^s_{\text{fit}} = 1$.

- First, we perform a maximum of 500 BFGS iterations setting $\omega^u_{\text{phys}} = \omega^v_{\text{phys}} = \omega^w_{\text{phys}} = \omega^s_{\text{phys}} = 10^{-6}$. In this way the ANN is trained according to a virtually pure fitting problem, in which the (low but not null) weights of the physical losses prevent that the residuals of the differential equation diverge. When the MPINN-v2 approach is considered, in this stage we set $\omega^u_{\text{lh}} = \omega^v_{\text{lh}} = \omega^w_{\text{lh}} = \omega^s_{\text{lh}} = 1$. Moreover, in this case, we precede the 500 iterations of BFGS by another

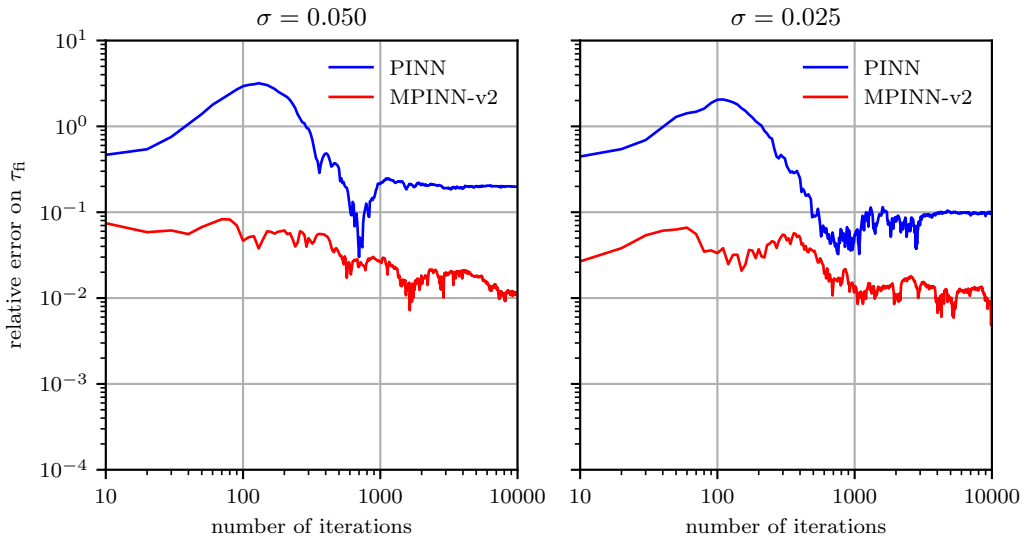

Figure 16: BO test case: relative error in the estimate of the parameter τ_{fi} with respect to number of epochs, for $\sigma = 0.05$ (left) and $\sigma = 0.025$ (right). The blue line refers to the PINN method, the red line to the MPINN-v2 method. The figure shows the results of 10 training processes (the shaded region is the area spanned by the trajectories; the solid line represents the geometric mean).

50 iterations (at most) in which we optimize solely with respect to the unknown parameter τ_{fi}. In this way we exploit the low-fidelity map to provide an educated guess to the parameter.

- Then, we perform 10000 BFGS iterations by increasing the weight factors of the physical losses to 10^{-3}, to enhance the residual minimization. For the MPINN-v2, we also decrease ω^{u}_{lh}, ω^{v}_{lh}, ω^{w}_{lh} and ω^{s}_{lh} to 0.05.

3.2.2 MPINN-v2 versus PINN approach

To compare the performance of PINN and MPINN-v2 methods, we perform 10 training runs for each method, starting from different random initializations of the ANN parameters. In Fig. 16 we show the trend with respect to the number of epochs of the second of the two stages described in Sec. 3.2.1 of the error obtained in the estimation of the parameter τ_{fi}, for two different values of the noise (i.e. $\sigma = 0.05$ and $\sigma = 0.025$). The results demonstrate that the MPINN-v2 method outperforms the PINN method. As a matter of fact, although the error obtained at the end of the training assumes different values in the 10 tests performed, the accuracy is always higher than in the PINN case. On average, the error obtained with the multi-fidelity approach is one order of magnitude lower than with the single-fidelity approach, for both cases considered ($\sigma = 0.05$ and $\sigma = 0.025$). Note that the error in the MPINN-v2 case starts lower than in the PINN case already from the first iteration of the second stage. This means that the first of the two stages described in Sec. 3.2.1 is able, in the multi-fidelity case, to provide, thanks to the low-fidelity parametric map, a good educated guess for the unknown parameter.

In Figs. 17 and 18 we show the detailed trend of the loss components in the second of the two stages described in Sec. 3.2.1, for two of the tests considered in Fig. 16. The figures refer to the

PINN and MPINN-v2 methods, respectively, in the case $\sigma = 0.025$. Note that, for both PINN and MPINN-v2, in the early epochs the loss is dominated by the physical components $\mathcal{J}^u_{\text{phys}}$, $\mathcal{J}^v_{\text{phys}}$, $\mathcal{J}^w_{\text{phys}}$ and $\mathcal{J}^s_{\text{phys}}$. When these are lowered by about an order of magnitude, $\mathcal{J}^u_{\text{fit}}$ and its gradients comes into play. Note that the latter component cannot be arbitrarily lowered, due to the noise which the measurements of u are subject to. If this happened, in fact, the model would suffer from overfitting.

We observe that in this example the values assumed by the different loss components are very similar in the single-fidelity and multi-fidelity cases. The difference lies in the $\mathcal{J}_{\text{lh}}$ terms, which are absent in the PINN. These terms lower the error in the MPINN estimate of τ_{fi} compared to the PINN case. Moreover, the remaining components of the loss are very similar in the two cases, despite having different errors with respect to the exact τ_{fi}. This reveals that the PINN formulation is severely ill-posed and that there exist multiple local minima. These minima share similar values of $\mathcal{J}$ for τ_{fi} estimates that are very different from each other. The effect of loss $\mathcal{J}_{\text{lh}}$ is thus to favor, among all these minima, those that correspond to more accurate reconstructions of the solution and parameter. In other words, $\mathcal{J}_{\text{lh}}$ improves the accuracy in parameter estimation acting as an additional regularizer to the physics-informed one.

4 Conclusions

In this work, we presented a new PINN multi-fidelity strategy that allows for the solution of parameter estimation problems. We compared the accuracy, the robustness and the efficiency of this new strategy with those of the standard PINN approach. As shown by the numerical results, the training of a PINN is especially challenging in configurations characterized by few noisy available measurements (small data regime), with a consequent poor simultaneous approximation of the solution and the unknown parameters (minima of the loss $\mathcal{J}$). It is in this context that our MPINN strategy, that integrates surrogate models in the PINN framework, turns out to be particularly effective.

We developed two different multi-fidelity approaches: MPINN-v1, based on a parameter-independent low-fidelity solution, acting only as initial guess for the parameters and the PDE solution to be estimated, and MPINN-v2, based on a parameter-dependent low-fidelity solution with a training procedure designed to simultaneously minimizing the distance between the PDE solution and the data, the distance between the PDE solution and the low-fidelity approximation for the same parameter value, and the residual of the differential equation.

Numerical tests were performed involving an advection-diffusion-reaction equation, a parabolic equation, and the Bueno-Orovio ionic model, focusing on the case of few noisy measurements. Both versions of MPINNs provided a speedup in the training process, converging more rapidly than a PINN. In addition, MPINN-v2 improved accuracy in estimating the unknown parameters. We showed in several numerical examples that MPINN-v2 reduces the relative error by one order of magnitude (on average) when compared to PINN.

We conclude that our multi-fidelity strategy is promising for solving realistic optimization problems involving noisy measurements and complex multiscale and multiphysics phenomena. Efficiency is guaranteed by the techniques employed in the training of neural networks, which are accelerated by the integration of the surrogate model approximation. Flexibility is instead provided by the possible integration of virtually any surrogate model, from simple data-driven black-box models to more complex projection-based models.

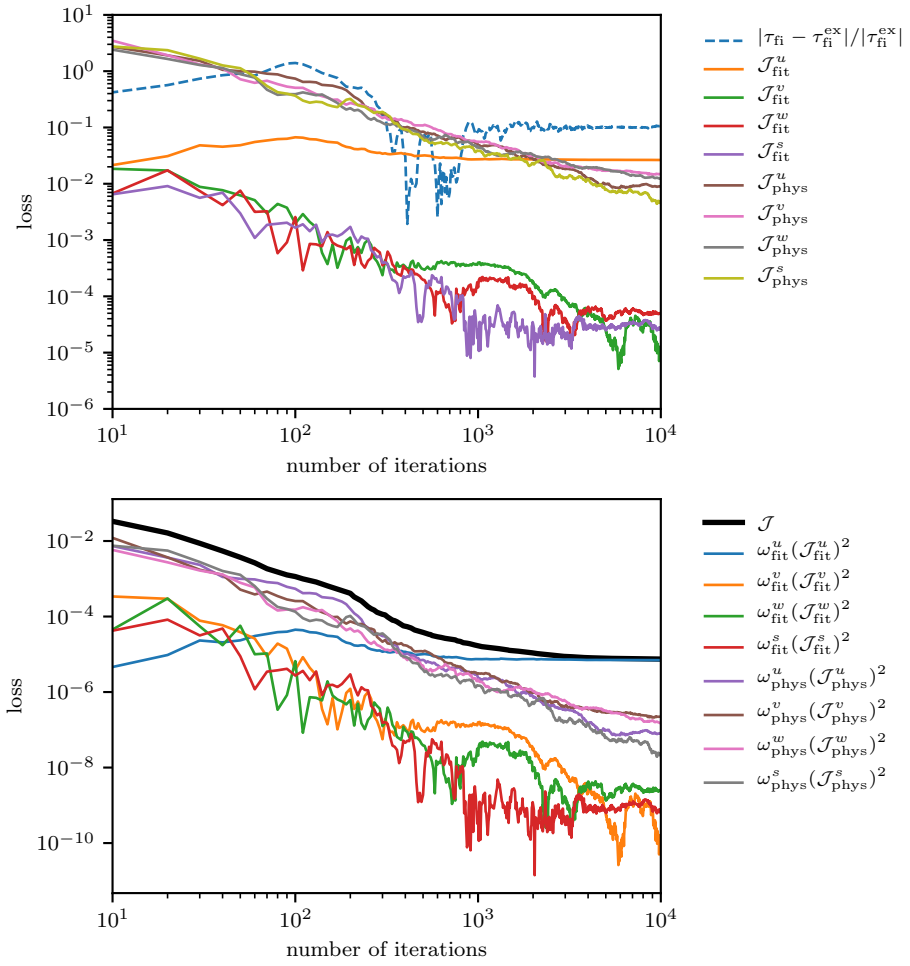

Figure 17: BO test case: loss components and relative errors trend versus the epochs in the PINN model.

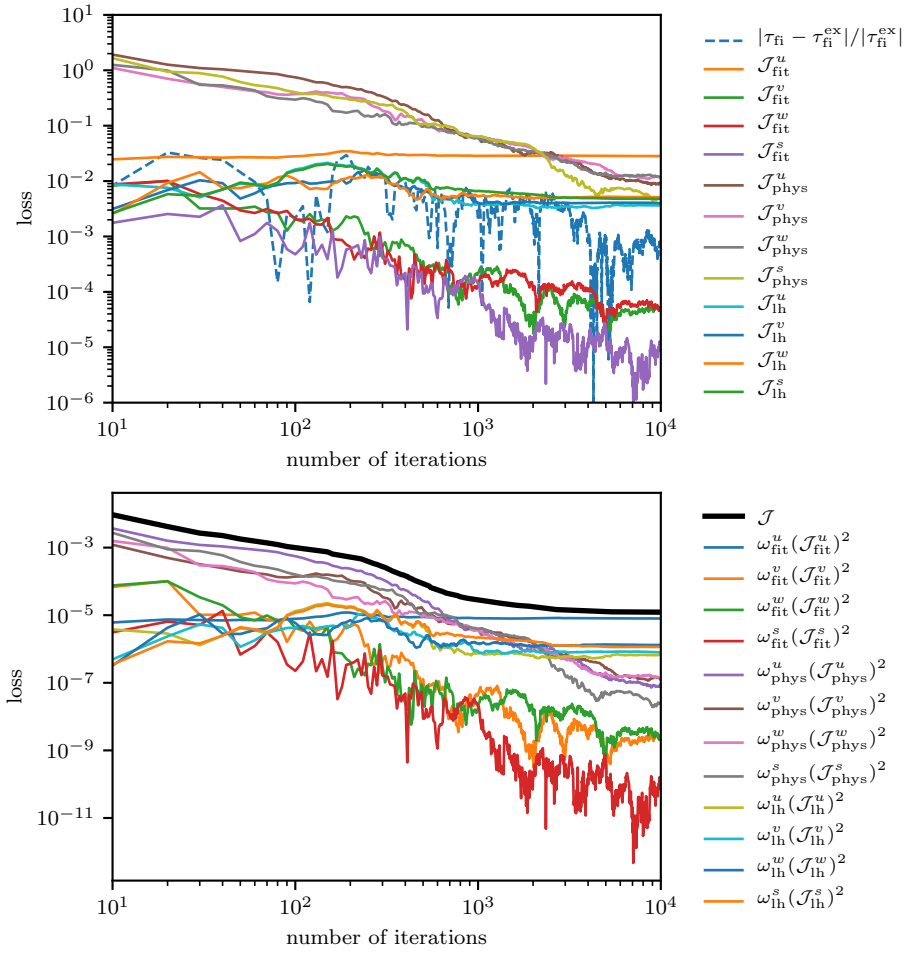

Figure 18: BO test case: loss components and relative errors trend versus the epochs in the MPINN-v2 model.

Acknowledgements

This project has been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 740132, iHEART - An Integrated Heart Model for the simulation of the cardiac function, P.I. Prof. A. Quarteroni) and by the Italian research project MIUR PRIN17 2017AXL54F "Modeling the heart across the scales: from cardiac cells to the whole organ".

References

[1] 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. Mané, 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. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] P. Benner, E. Sachs, and S. Volkwein. Model order reduction for pde constrained optimization. *Trends in PDE constrained optimization*, pages 303–326, 2014.

[3] L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders. Large-scale pde-constrained optimization: an introduction. In *Large-Scale PDE-Constrained Optimization*, pages 3–13. Springer, 2003.

[4] A. Borzì and V. Schulz. *Computational optimization of systems governed by partial differential equations*. SIAM, 2011.

[5] A. Bueno-Orovio, E. M. Cherry, and F. H. Fenton. Minimal model for human ventricular action potentials in tissue. *Journal of theoretical biology*, 253(3):544–560, 2008.

[6] F. Chollet. Keras. https://github.com/fchollet/keras, 2015.

[7] L. Dede. Reduced basis method and a posteriori error estimation for parametrized linear-quadratic optimal control problems. *SIAM Journal on Scientific Computing*, 32(2):997–1019, 2010.

[8] M. A. Dihlmann and B. Haasdonk. Certified pde-constrained parameter optimization using reduced basis surrogate models for evolution problems. *Computational Optimization and Applications*, 60(3):753–787, 2015.

[9] R. Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[10] A. Forrester, A. Sobester, and A. Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[11] A. I. Forrester, A. Sóbester, and A. J. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, 463(2088):3251–3269, 2007.

[12] P. C. Franzone, L. F. Pavarino, and S. Scacchi. *Mathematical cardiac electrophysiology*, volume 13. Springer, 2014.

[13] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[14] J. S. Hesthaven, G. Rozza, B. Stamm, et al. *Certified reduced basis methods for parametrized partial differential equations*, volume 590. Springer, 2016.

[15] M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[18] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[19] K. O. Lye, S. Mishra, and R. Molinaro. A multi-level procedure for enhancing accuracy of machine learning algorithms. *European Journal of Applied Mathematics*, 32(3):436–469, 2021.

[20] K. O. Lye, S. Mishra, D. Ray, and P. Chandrashekar. Iterative surrogate model optimization (ismo): An active learning algorithm for pde constrained optimization with deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 374:113575, 2021.

[21] A. Manzoni and S. Pagani. A certified rb method for pde-constrained parametric optimization problems. *Communications in Applied and Industrial Mathematics*, 10(1):123–152, 2019.

[22] X. Meng and G. E. Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *Journal of Computational Physics*, 401:109020, 2020.

[23] S. Mishra and T. K. Rusch. Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. *arXiv preprint arXiv:2005.12564*, 2020.

[24] B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review*, 60(3):550–591, 2018.

[25] D. C. Psichogios and L. H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.

[26] A. Quarteroni. *Numerical models for differential problems*. Springer, 2017.

[27] A. Quarteroni, L. Dede', A. Manzoni, and C. Vergara. *Mathematical modelling of the human cardiovascular system: data, numerical approximation, clinical applications*, volume 33. Cambridge University Press, 2019.

[28] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced basis methods for partial differential equations: an introduction*, volume 92. Springer, 2015.

[29] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*, volume 37. Springer Science & Business Media, 2010.

[30] M. Raissi and G. E. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[31] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[32] M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[33] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020.

[34] F. Tröltzsch. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. American Mathematical Soc., 2010.

[35] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[36] S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536*, 2020.

[37] S. Wang, X. Yu, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *arXiv preprint arXiv:2007.14527*, 2020.

[38] M. J. Zahr and C. Farhat. Progressive construction of a parametric reduced-order model for pde-constrained optimization. *International Journal for Numerical Methods in Engineering*, 102(5):1111–1135, 2015.